



Proceedings of the 13th International Conference on Applied Informatics

Eger, Hungary
February 19–21, 2026

2026

Proceedings of the 13th International Conference on Applied Informatics

Eger, Hungary
February 19–21, 2026

Organized by



Edited by

Gergely Kovásznai, Eszterházy Károly Catholic University
Tibor Tómacs, Eszterházy Károly Catholic University

ISBN 978-963-496-327-1 (PDF)

Responsible for publication
Rector of Eszterházy Károly Catholic University
Published by Líceum Kiadó
Publishing director: Patrik Dinnyés
Technical editor: Tibor Tómacs
Published: August 2026

Contents

Preface	1
A. AGÁRDI, Combination of Ant Algorithms and Simulated Annealing in solving Flow Shop Scheduling Problem	3
F. AZIZ, N. OLÁH, LLM-Augmented Machine Learning for Phishing Email Detection: Enhancing Classification, Explainability, and Multilingual Support	21
T. BALLA, Practical Prediction of Query Execution Time in PostgreSQL Using Lightweight Machine Learning Models	34
J. D. BALOGH, A. ADAMKÓ, Accuracy of Wearable Heart Rate Monitors under Static and High-Intensity Activity Conditions	48
N. BANYIK, B. NAGY, Creating Images on Non-Traditional Grids for Digital Image Processing	55
P. BORBÁS, E. TROLL, Topology-Aware Neural Vectorization of Orienteering Maps: Contour Segmentation as a Demanding Test Case for Editable Reconstruction	66
P. J. CSÁSZÁR, M. CSERÉP, Assessing Software Metrics and Complexity in Student and Open-Source Projects Using Static Analysis	81
T. T. FÜRÉSZ, K. UMANN, Z. PORKOLÁB, Static Analysis Possibilities for Regular Expressions in C++	96
T. GIRÁSZI, N. PAPP, N. OLÁH, A. HUSZTI, LLM-Powered Automated Attacks	107
L. HAORYUAN, M. BOBÁLY, J. T. BATTA, A. FAZEKAS, A Proposed Method for Measuring the Aesthetic Outcome of Head Surgery	122
M. HERBÁK, G. KOVÁSZNAI, A. A. ADIL, Optimizing Collaborative Learning: A Hard-Constraint Reinforcement Learning Approach to Fair Team Composition	133
D. G. HONI, A. A. MOHAMMED, L. SZATHMARY, An Interpretable Machine Learning Framework for Early Ectopic Pregnancy Prediction Using Routine Clinical Variables	146
F. KOCZKA, Comparative Analysis of Vulnerability Dynamics in Operating Systems and Containerized Environments	161
ZS. LACZKÓ, M. TEJFEL, Program Patterns in P4 Programs	169
T. LUKOVSKI, G. PUSZTAI, Experimental Study of Building a Nest by an Automaton	182
P. Á. MAKÁR, G. KOCIS, An Improved Measurement and Visualization System for High Precision Positioning with GPS-RTK	192
B. NAGY, GY. HORVÁTH, AI-Based Estimation of the Zone of Proximal Development in Programming Education	202
I. PINTÉR, L. KOVÁCS, Acoustic Traffic Monitoring with MEMS Microphones, Concatenated log-Bark Spectrograms and CNN	216
P. SÁGHELYI, P. BEREZKY, Formal Verification of Questionnaire Logic Using SMT Solvers	227

Sz. SZILÁGYI, An Empirical Study of HomePlug AV2 PowerLine Performance
in Real-World Residential Electrical Networks 244

Zs. VALISKÓ, T. BALLA, Cs. BIRÓ, Structural Node Importance Beyond
Classical Centrality: A Comparative Study of Entropy-Based k -Hop
Measures 257

I. VARGA, Local Scientific Collaboration Networks: A Case Study 271

Preface

In today's dynamic and ever-evolving digital landscape, applied informatics plays a pivotal role in shaping the future of technology. From advancing artificial intelligence and data science to strengthening cybersecurity, optimizing cloud networks, and supporting educational processes, the realm of applied informatics continues to drive innovation and transformation across industries. These current proceedings feature contributions from the 13th International Conference on Applied Informatics (ICAI 2026), which was held in Eger, Hungary on February 19–21, 2026. These research papers explore novel insights, innovative methodologies, and practical applications within the field of computer science and informatics. Each of them represents a valuable contribution to the applied informatics field and offers insights that bridge the gap between theory and practical application. They are a testament to the diversity and dynamism of our field, showcasing a wide range of research topics and applications.

The ICAI Conference series is traditionally held in Eger in every 3 years, and is jointly organized by the Eszterházy Károly Catholic University and the University of Debrecen. The conference is oriented towards professional exchange of ideas in the field of Applied Informatics. Reflecting the rapidly expanding horizons of the discipline, the conference covers a rich spectrum of domain areas, including:

- Artificial Intelligence including Machine Learning
- Computer Aided Geometric Modeling and Computer Graphics
- Computer Networks and Cloud Services
- Theoretical Computer Science
- Data Science including Database Systems
- Education
- Applications of Informatics
- Functional Programming
- Operational Research
- Security and Cryptography
- Smart Solutions including Embedded systems
- Software Design and Object-Oriented Programming
- Testing and Verification including Formal Methods
- Data Visualization
- Quantum Informatics
- Simulation and Modeling
- Applied Mathematics

The goal of the conference is to provide a forum for the discussion of academic research and industry, fostering interdisciplinary collaboration across these diverse topics.

ICAI 2026 Committees

Conference Chairmen

- József Sütő
- Gábor Kusper

Secretary

- Erzsébet Tóth
- Ede Troll

Program Committee

- **Chair:** Gergely Kovásznai
- Erika Abraham
- Anikó Apró
- Imre Baják
- Erika Baksáné Varga
- Tamás Balla
- Nándor Banyik
- Ágnes Baran
- Sándor Baran
- Csaba Biró
- İsmail Bögrekci
- János Botzheim
- Adrian Crăciun
- Lehel Csató
- Isabela Drămnesc
- Attila Egri-Nagy
- Attila Fazekas
- Katalin Fazekas
- Zoltán Gál
- Balázs Harangi
- Attila Házy
- Tamás Herendi
- Miklós Hoffmann
- Olivér Hornyák
- Attila Horváth
- Ildikó Horváth
- Zoltán Istenes
- Roland Király
- Sándor Király
- Gergely Kocsis

- Ádám Kovács
- Roland Imre Kunkli
- Gábor Kusper
- Tünde Lengyelne Molnár
- Mircea Marin
- Péter Mileff
- Bálint Molnár
- Benedek Nagy
- Károly Nehéz
- István Oniga
- Norbert Pataki
- Nikolaj Popov
- Al-Shamarti Mohammed
Ridha Nsaif
- Wolfgang Schreiner
- Stefan Schupp
- Martina Seidl
- József Sütő
- Sorin Stratulat
- Csaba Szabó
- Szabolcs Szilágyi
- Tibor Tajti
- Ede Troll
- József Udvaros
- Imre Varga
- György Vaszil
- Győző Yang
- Viktória Zsók

Organizing Committee

- **Chair:** Csaba Biró
- Tamás Balla
- Tamara Ipacs
- Sándor Király
- Ádám Kovács
- Ágnes Nagyné Bertalan
- Viktória Szabó
- Péter Szigetváry
- Péter Takács
- Erzsébet Tóth
- Ede Troll



Combination of Ant Algorithms and Simulated Annealing in solving Flow Shop Scheduling Problem*

Anita Agárdi

University of Miskolc
anita.agardi@uni-miskolc.hu

Abstract. In this article, a common production scheduling task is presented and solved. This problem is the Flow Shop Scheduling. During the problem, given n jobs and m machines. All jobs must be processed in the same machine order. A job can only be started on a given machine if the processing on the previous machine has already been completed and the machine is free. The objective function of the problem is the minimization of the makespan. In this paper, two Hybrid Ant Algorithms were applied to the problem and solved the Taillard benchmark dataset. These two algorithms are the Hybrid Ant Colony System with Simulated Annealing and the Hybrid Rank Based Version of Ant System with Simulated Annealing.

Keywords: Ant Colony System, Rank Based Version of Ant System, Flow Shop Scheduling

2020 *Mathematics Subject Classification:* Primary 90B35; Secondary 90C59, 68W40, 90C27

1. Introduction

The efficient use of available resources is a key issue in the operation of modern industrial and service systems. During production, many tasks have to be performed on different machines, either in a specific order and possibly under a time limit. This falls under the task of production scheduling. The goal is usually to optimize

*Supported by the University Research Scholarship Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

some performance indicators. The performance indicator can be, for example, minimizing lead time, increasing machine utilization, reducing delays, minimizing the makespan.

Flow Shop Scheduling [19] is a common problem in production scheduling tasks, where n jobs and m machines are given. During the problem, all jobs must be processed in the same machine order. A job can only be started on a given machine if the processing on the previous machine has already been completed and the machine is free. The common objective function of the problem is to minimize the makespan. Over the years, many variations of the problem have been developed to suit real industry needs. Some of these specific constraints, types, and objective functions are as follows: In the case of Permutation Flow Shop Scheduling Problem (PFSP) [29] the jobs are processed in the same order on each machine. Non-permutation Flow Shop Scheduling Problem [26] is the case, when the order of jobs can vary from machine to machine. Blocking Flow Shop Scheduling Problem [30] is a task, when there is no buffer capacity between machines in the system. If the next machine is busy, the job on the previous machine will occupy that machine. No-wait Flow Shop Scheduling Problem [25] means that there can be no waiting between two consecutive operations. No-idle Flow Shop Scheduling Problem [24] is a case, when machines cannot stop between two consecutive jobs. In the case of Flexible Flow Shop Scheduling Problem [11], multiple parallel machines are available in one or more operation stages. Distributed Flow Shop Scheduling Problem [16] means, that the system contains multiple factories (production sites). The jobs have to be distributed among them. In the case of Stochastic Flow Shop Scheduling Problem [14] some parameters of the job are represented as random variables. Multi-objective Flow Shop Scheduling Problem [20] is a task, when multiple objective functions are optimized simultaneously.

In the following, some important publications are presented where the authors used the Ant Colony System algorithm to solve certain Flow Shop tasks.

Li Xiangyong, Baki Md Fazle and Aneja Yash P [13] investigated a Flow Shop Scheduling Problem with batching and the assumption of item availability with m machines. The objective of the problem is to minimize the total completion time. A single operator is required to process the jobs and set up the machines. At most one machine can be active in the system at a time. Each job needs to be processed only once. Data sets of different difficulty were used during testing. The sizes were as follows, where m is the number of machines and n is the number of jobs: $(m = 2, n = 5)$, $(m = 2, n = 10)$, $(m = 3, n = 10)$, $(m = 5, n = 10)$ and $(m = 3, n = 20)$.

Chaouch Imen, Driss Olfa Belkahla and Ghedira Khaled [9] used a modified Ant Colony algorithm. They investigated a special version of the Flow Shop scheduling problem (Distributed Job Shop Scheduling Problem). In this model, jobs are distributed among different, geographically separated factories. The goal is to minimize the makespan. The following Ant Colony algorithms were used to solve the problem: ACO, Ant System (AS) and Modified Ant Colony Optimization (MACO). The model assumes that the tasks are independent of each other. Once a task is

assigned to a factory, it cannot be moved to another factory later. Interruption of operations is not allowed. A machine can only execute one operation at a time. The tests were performed on the well-known Taillard benchmark datasets.

Komaki GM, Sheikh Shaya and Malakooti Behnam [12] present a Flow Shop model that also includes assembly operations. The assembly Flow Shop system consists of a manufacturing or machining phase and then an assembly phase. The final products are assembled in a hierarchical structure. So, each component must first be machined separately. Then, the components must be assembled according to the predefined structure. The paper describes several different optimization methods, such as Artificial Bee Colony, Ant Colony Optimization, Genetic Algorithm, Particle Swarm Optimization, Simulated Annealing, Tabu Search, Variable Neighborhood Search. The objective function under study has several components: minimizing the makespan, reducing the total completion time, minimizing delays, and reducing inventory holding costs.

Chen Zhen, Zheng Xu, Zhou Shengchao, Liu Chuang and Chen Huaping [10] apply a Quantum-Inspired Ant Colony Optimization Algorithm (QIACO). The model uses two-phase permutation scheduling and batch processing machines in the Flow Shop task. During the operation of the algorithm, the ants are divided into two groups: one group prefers larger jobs, while the other selects smaller ones.

Ying Kuo-Ching and Liao Ching-Jong [28] use the Ant Colony System (ACS) algorithm to solve the Flow Shop Sequencing Problem. The results are also compared with other metaheuristic methods. The performance of the algorithm is tested on problems of different sizes during the experiments.

We can find many hybrid metaheuristic approaches in the literature, where the authors solve the classic Flow Shop problem or some variant of it. The article now highlights some publications, where the Taillard benchmark dataset is solved by the individual authors.

Zhou, Y., Chen, H., and Zhou, G. [31] solved the Taillard dataset with Invasive weed optimization algorithm. Although the results were better than the NEH algorithm, PSOns and HDPSO, they were far from the best known value.

Wei, H., Li, S., Jiang, H., Hu, J., and Hu, J. [27] investigated the Hybrid Simulated Genetic Algorithm and tested on the Taillard benchmark dataset. The authors generated the initial values of the population not randomly, but with the MinMax (MM) and Nawaz–Enscore–Ham (NEH) algorithms. The results were also considerably distant from the best known value.

Tseng, L. Y., and Lin, Y. T. [23] developed and tested a hybrid Genetic Algorithm on the Taillard benchmark dataset. The GA algorithm was hybridized with two Local Search algorithms. The results were also not close to the best-known value.

Qu, C., Fu, Y., Yi, Z., and Tan, J. [17] tested the efficiency of the Flower Pollination Algorithm on the Taillard dataset. The authors also presented the results of hybridizing the Ant Colony Optimization algorithm with the Simulated Annealing algorithm, which were also far from the best known value.

The importance of Simulated Annealing-based improvement

In the previous work of the author, several metaheuristic methods have been implemented to solve Taillard benchmark problems. These algorithms were not applied in a hybrid form, but as standalone approaches.

Based on the results, it can be observed that these purely metaheuristic methods generally show significant deviations from the best known solutions of the benchmark datasets:

- Hill Climbing [3]
- Elitist Strategy of Ant System [3]
- Ant System [2]
- Rank Based Version of Ant System [4]
- MAX-MIN Ant System [1]

In this paper, a Simulated Annealing-based improvement approach is investigated, which was tested on several benchmark datasets. The algorithm achieved the best known value in many cases during each run. In those cases where deviations from the best known value were observed, the deviations remained small. Based on the relative standard deviation (RSD) values, it can be observed that the algorithm showed stable behavior, since the fitness values of the solutions obtained during different runs produced low standard deviations and, in many cases, nearly identical results. This suggests that the method is robust.

The paper is structured as follows. Section 2 presents the applied algorithms. Then Section 3 presents the test results. Section 4 provides a conclusion and future research direction.

2. Applied algorithms

This section presents the applied algorithms: the Ant Colony System algorithm, the Rank Based Version of Ant System algorithm, and Simulated Annealing.

2.1. Ant Colony System

The Ant Colony System [21] algorithm is a member of the Ant Colony Optimization (ACO) algorithm family. ACO algorithms are nature-inspired metaheuristic algorithms that maintain a population of solutions. They model the way ants search for food.

Constructing a route [21]: Initially, each ant starts from a randomly selected permutation element. At each step, the ants decide which permutation segment to follow with a given probability.

The probability that ant k , which is lastly selected permutation element i , will select element j in iteration t is given by:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} \quad \text{when } j \in N_i^k$$

where $\eta_{ij} = \frac{1}{d_{ij}}$ is the reciprocal of the distance between two elements. $\tau_{ij}(t)$ is the current pheromone level of the pair (i, j) . α and β are pheromone and distance control parameters. N_i^k denotes the set of elements not yet selected by ant k .

Global Pheromone Update [21]: During the algorithm, only the ant – which represents the globally best solution – places pheromone.

The update is done using the following formula:

$$\tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \rho \Delta \tau_{ij}^{gb}(t)$$

where $\Delta \tau_{ij}^{gb}(t) = \frac{1}{L^{gb}}$, L^{gb} is the fitness of the globally best ant, ρ is the evaporation coefficient ($0 < \rho \leq 1$) and $\tau_{ij}(t)$ is the current pheromone level of the pair (i, j) . In this procedure, only the solution of best ant is updated.

Local pheromone update [21]: The algorithm also uses local updating:

$$\tau_{ij} = (1 - \xi) \tau_{ij} + \xi \tau_{ij}^0$$

where $0 < \xi < 1$, τ_{ij}^0 is the initial pheromone level and τ_{ij} is the current pheromone value of the permutation pair (i, j) .

This step reduces the pheromone level of the given edge, thus facilitating a wider search space.

Steps of the algorithm

The algorithm consists of the following steps [21]:

1. **Initialization of pheromone values.** This can be done based on an initial solution, or each edge can be given the same pheromone value.
2. **Route construction.**
 - Random selection of initial permutation element (solution element).
 - Calculating the selection probabilities of elements that have not yet been selected.
 - Select the next element.
3. **Local pheromone update.**

$$\tau_{ij} = (1 - \xi) \tau_{ij} + \xi \tau_{ij}^0$$

4. Repeat the above steps until the ant has completed its route, i.e. the algorithm has produced a solution.
5. **Global pheromone update.** After all ants have created their solution, we perform an update on the best solution:

$$\tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \rho \Delta \tau_{ij}^{gb}(t)$$

6. Steps 2–3 are repeated until the stopping condition is met. The stopping condition can be reaching a given number of iterations, convergence, or running time.

2.2. Rank Based Version of Ant System

The Rank Based Version of Ant System (RBVAS) [7] is an improved version of the Ant System algorithm. This algorithm is also a member of the Ant Colony Optimization family. Only the best ants contribute to increase in pheromone values. The algorithm ranks the ants in each iteration based on their fitness values:

$$L^1(t) \leq L^2(t) \leq \dots \leq L^m(t)$$

The amount of deposited pheromone depends on the rank (r) of the ant. Only the best ($w - 1$) ant of the current iteration deposits pheromone on the visited road sections. The ant that provides the globally best solution will deposit pheromone with weight w . The ant r deposits pheromone with the following weight:

$$\max\{0, w - r\}$$

The modified pheromone update is done with the following formula:

$$\tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \sum_{r=1}^{w-1} (w - r) \Delta \tau_{ij}^r(t) + w \Delta \tau_{ij}^{gb}(t)$$

where $\Delta \tau_{ij}^r(t) = \frac{1}{L^r(t)}$, $\Delta \tau_{ij}^{gb}(t) = \frac{1}{L^{gb}}$, $\tau_{ij}(t)$ is the pheromone level of edge (i, j) in iteration t , ρ is the evaporation parameter.

Steps of the algorithm

The algorithm consists of the following steps [21]:

1. **Initialization of pheromone values.**
2. **Route construction.** Each ant constructs a solution in the following way:
 - The ant randomly selects a starting solution (permutation) element.
 - Then, it calculates the probability for each element (i, j) , where i is the last element of the partial solution and j is not yet included.

The probability is given by:

$$p_{ij}^k = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} \quad \text{if } j \in N_i^k$$

where:

- $\eta_{ij} = \frac{1}{d_{ij}}$,
- $\tau_{ij}(t)$ is the current pheromone level of pair (i, j) ,
- t is the iteration number,
- N_i^k is the set of elements not yet visited.

The next permutation element is selected based on the calculated probabilities. The steps are repeated until a complete permutation (solution) is obtained.

3. **Pheromone update.** After all ants have created their solutions, the pheromone values are updated. The modification is based on the globally best ant and the best $(w - 1)$ ants of the given iteration.

The update formula is:

$$\tau_{ij}(t+1) = (1 - \rho) \tau_{ij}(t) + \sum_{r=1}^{w-1} (w - r) \Delta \tau_{ij}^r(t) + w \Delta \tau_{ij}^{gb}(t)$$

where:

- $\Delta \tau_{ij}^r(t) = \frac{1}{L^r(t)}$,
- $\Delta \tau_{ij}^{gb}(t) = \frac{1}{L^{gb}}$,
- ρ is the evaporation rate.

4. **Iteration.** Steps 2–3 are repeated until a stopping condition is met. This can be reaching a given number of iterations, convergence, or a time limit.

2.3. Simulated Annealing

Simulated Annealing [6] is inspired by the metallurgical heat treatment process. In the hardening of metals, the material is first heated to a high temperature and then slowly cooled. As the temperature is gradually reduced, the material adopts a lower-energy crystal structure.

During the search, random small modifications are taken to the current solution. That is, generating neighbors of the current solution. If the new solution is better than the current solution, it is accepted. If the new solution is worse, it can only be accepted with a certain probability. This acceptance probability depends on the energy change (ΔE) and decreases when decreasing temperature (T). In the initial stages of the algorithm, worse solutions are more likely to be accepted. Accepting worse solutions helps to escape from local optima.

Steps of the algorithm

The algorithm consists of the following steps [6]:

1. **Create an initial solution.** Select a possible solution, which will be the current solution: S_{current} .
2. **Create a neighboring solution.** Modify the current solution to produce a neighboring solution: S_{neighbor} . In this study, the 2-opt [15] operator was used to create the neighbor solution.
3. **Calculate the energy change.**

$$\Delta E = E(S_{\text{neighbor}}) - E(S_{\text{current}})$$

4. **Acceptance decision.** If $\Delta E < 0$, then the new solution is better, therefore:

$$S_{\text{current}} = S_{\text{neighbor}}$$

Otherwise, the new solution is accepted only with the following probability:

$$P(e^{-\Delta E/T} > \text{rand}[0, 1])$$

5. **Internal iteration.** Steps 2–4 are repeated L times (process length).
6. **Temperature reduction.** This is done using the following formula:

$$T = \alpha \cdot T$$

7. **Outer loop.** Steps 2–6 are repeated until a stopping condition is met. The stopping condition can be a given number of iterations, running time, or convergence.

2.4. Improvement with Simulated Annealing

The presented algorithms were improved using the Simulated Annealing method. The improvement was created in a similar way as in the publication [5] with the difference that here the Simulated Annealing improvement was not performed as the last step of each iteration, but as an intermediate step of the algorithm-specific operators. In addition, an Iterative Simulated Annealing is also examined, when the algorithm simply re-runs several Simulated Annealings.

The improvement with Simulated Annealing (similar to the publication [5]) was performed in the following way for each iteration of the algorithm:

1. Generate and evaluate the initial population.
2. Iteratively perform the following operations until the stopping condition is met:

- (a) Iterate over the elements of each population and perform the given algorithm-specific operators on each element.
- (b) Iterate over the elements of each population again and improve each element using the Simulated Annealing method.

2.5. Problem representation

A permutation representation method is used for the task. The solution is described as a sequence of jobs (a permutation) $\pi = (\pi_1, \pi_2, \dots, \pi_n)$. The permutation defines a common processing order of jobs on all machines.

Let $p_{k,g}$ denote the processing time of job k on machine g , where n is the number of jobs and m is the number of machines.

The completion time of the job at position t in the permutation on machine g , denoted by $C_{g,t}$, is computed recursively as follows:

$$C_{g,t} = \max(C_{g-1,t}, C_{g,t-1}) + p_{\pi_t,g}$$

with boundary conditions:

$$C_{0,t} = 0 \quad \forall t = 1, \dots, n, \quad C_{g,0} = 0 \quad \forall g = 1, \dots, m.$$

The term $C_{g-1,t}$ ensures that a job cannot start on machine g before it has completed processing on the previous machine. $C_{g,t-1}$ ensures that machine g processes jobs sequentially according to the permutation.

The objective function is makespan minimization (completion time of the last job on the last machine):

$$C_{\max} = C_{m,n}.$$

3. Test results

This section presents the test results of the Taillard [22] dataset, which is a Flow Shop Scheduling benchmark dataset. It is one of the most well-known test datasets for production scheduling problems. It was published by Éric Taillard in 1993. The Taillard examples contain problems of different sizes, consisting of different numbers of jobs and machines. For the Taillard benchmark dataset, the objective function is the makespan minimization. The tables consist of the following columns: the Taillard dataset number, the minimum, maximum and average fitness GAP values of the runs. The RSD values are also presented in the table.

The GAP value is calculated in the following way:

$$\text{GAP}(\%) = \frac{C_{\text{alg}} - C_{\text{best}}}{C_{\text{best}}} \cdot 100$$

where C_{alg} is the objective function value obtained by the algorithm, C_{best} is the value of the best known solution, $\text{GAP}(\%)$ is the relative deviation in percentage.

The RSD (Relative Standard Deviation) is calculated in the following manner:

$$RSD = \frac{\sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}}{\bar{x}} \cdot 100$$

3.1. Parameter settings and experimental environment

The algorithms were applied with the following parameter settings during the experiments.

Hybrid Ant Colony System (HACS):

- number of ants: 35,
- evaporation rate (ρ): 0.8,
- pheromone effect (α): 1,
- heuristic information effect (β): 2,
- local update parameter (κ): 0.8.

Hybrid Rank Based Version of Ant System (HRBVAS):

- number of ants: 35,
- evaporation rate (ρ): 0.8,
- pheromone effect (α): 1,
- heuristic information effect (β): 2,
- rank-based weighting parameter (w): 6.

Simulated Annealing (SA):

- number of iterations: 4000,
- initial temperature: 1000,
- cooling rate (α): 0.1.

In the case of Ant-based algorithms, the stopping condition was the convergence.

During the evaluation of the experimental results, 10 independent runs were performed for each test case. The algorithms were implemented in Java and run on a personal computer. For the smallest problems, the running time was 20-30 seconds, while for the largest instances, the running time was significantly longer, about 3-4 minutes.

3.2. Hybrid Ant Colony System test results

The Table 1 shows the performance of the Hybrid Ant Colony System (HACS) algorithm on the Taillard benchmark problem. The table includes the minimum, maximum, and average GAP values from the best known (BK) solutions in percentage form. It also contains the RSD values.

Table 1. Statistical results of Ta for Hybrid ACS.

Ta	Min	Max	Avg	RSD	Ta	Min	Max	Avg	RSD
1	matches best known				26	0.9883	1.4825	1.3208	0.1998
2	0.0000	0.0736	0.0074	0.0233	27	1.8038	2.3757	2.0326	0.2077
3	0.0000	0.3700	0.0370	0.1170	28	0.6818	2.0455	1.5636	0.5167
4	0.0000	0.3094	0.1160	0.1422	29	1.6987	2.4587	2.0563	0.3136
5	matches best known				30	1.2397	2.7089	2.2406	0.5936
6	matches best known				31	matches best known			
7	0.4052	0.9724	0.4619	0.1786	32	0.1411	0.4587	0.2893	0.1438
8	matches best known				33	0.0000	0.1145	0.0305	0.0497
9	matches best known				34	0.0727	0.0727	0.0727	0.0000
10	matches best known				35	matches best known			
11	0.7585	2.0228	1.5044	0.3280	36	0.0000	0.0707	0.0424	0.0387
12	0.7836	1.6275	1.1995	0.2685	37	0.0000	0.1101	0.0220	0.0492
13	1.4037	2.5401	1.7781	0.3423	38	0.0000	0.4100	0.2684	0.1974
14	0.5084	2.4691	1.4379	0.5675	39	0.1176	0.3527	0.2508	0.0941
15	0.4228	2.0437	1.0500	0.4803	40	matches best known			
16	0.3579	1.2885	0.7731	0.2818	41	2.8753	3.6108	3.3768	0.2801
17	0.3369	1.4151	0.7884	0.3057	42	2.6160	3.5926	3.2368	0.3642
18	0.7802	2.1456	1.6450	0.3932	43	2.8179	4.0859	3.6210	0.4620
19	0.5022	1.0672	0.8035	0.2232	44	0.6856	1.5344	1.1688	0.3073
20	0.0000	1.6970	0.9742	0.4991	45	2.4530	3.3266	3.0712	0.3501
21	1.6108	2.4380	1.9417	0.3656	46	2.0625	2.5615	2.4152	0.2087
22	1.7151	2.0962	1.9438	0.1852	47	1.6166	2.2308	2.0045	0.2435
23	1.9776	2.3646	2.1152	0.1440	48	0.5927	1.9098	1.3434	0.5424
24	1.1696	2.1592	1.6104	0.3602	49	1.6569	2.5889	1.9468	0.3790
25	1.2658	2.0079	1.6150	0.2665	50	2.4796	3.1974	2.8516	0.2887

Small-scale problems (Ta1–Ta10) For small problems, HACS performs very well. In several cases (Ta1, Ta5, Ta6, Ta8, Ta9, Ta10) the minimum, maximum and average values are the same. Here, the GAP is 0% in all cases. For the Ta2–Ta4 instances, the maximum GAP is also low. For the Ta7 dataset, the average GAP is 0.46%. This is still low. Here, Hybrid ACS is practically optimal or close to optimal, and the variance is also minimal.

Medium-sized problems (Ta11–Ta30) Larger GAP values can be observed in the case of medium-sized problems. The maximum GAP values are between 1.0-3.0%. For the Ta11–Ta20 instances, the average GAP is around 1%. For the Ta21–Ta30 problems, however, in the most cases the average GAP exceeds 1.5%. The maximum GAP values are below 2.0% in several cases.

Larger problems (Ta31–Ta50) For Ta31–Ta40 data, the GAP is again 0% or very low in most of the test runs. For Ta31, Ta35 and Ta40 the test results matches the best known values. For Ta41–Ta50 data, performance degradation is

observed. The average GAP values are mostly between 2–3.6%. E.g. Ta43: avg GAP $\approx$ 3.62%, Ta41: avg GAP $\approx$ 3.38%, Ta42: avg GAP $\approx$ 3.24%, Ta50: avg GAP $\approx$ 2.85%.

Stability test (min–max difference) For smaller problems, the min and max values are often the same, this means that the standard deviation is very low. For larger problems, the min–max difference increases (e.g. Ta41–Ta50).

3.3. Hybrid Rank Based Version of Ant System test results

The Table 2 indicates the performance of the Hybrid Rank Based Version of Ant System (HRBVAS) algorithm on the Taillard benchmark dataset. The table shows the minimum, maximum and average fitness values for each problem, as well as the deviations from the best known solution (GAP).

Table 2. Statistical results of Ta for Hybrid RBVAS.

Ta	Min	Max	Avg	RSD	Ta	Min	Max	Avg	RSD
1	matches best known				26	1.1231	1.4376	1.2579	0.1331
2	matches best known				27	1.5398	2.1557	1.9534	0.2414
3	0.0000	0.3700	0.0370	0.1170	28	0.9545	2.7727	1.9182	0.6919
4	0.0000	0.3094	0.1237	0.1596	29	1.9222	2.3692	2.1368	0.1943
5	0.0000	0.1619	0.0162	0.0512	30	1.6988	2.7089	2.1212	0.3950
6	matches best known				31	matches best known			
7	0.4052	0.4052	0.4052	0.0000	32	0.1411	0.4587	0.3035	0.1129
8	matches best known				33	0.0382	0.1145	0.0687	0.0418
9	matches best known				34	0.0727	0.0727	0.0727	0.0000
10	matches best known				35	matches best known			
11	0.6321	1.6435	1.1631	0.3400	36	0.0000	0.0707	0.0424	0.0387
12	0.3014	1.6878	1.1875	0.3820	37	0.0000	0.2569	0.1908	0.1080
13	0.8690	2.2727	1.5508	0.4043	38	0.0000	0.4100	0.2013	0.2048
14	0.6536	1.5251	1.1983	0.2954	39	0.0784	0.3527	0.2665	0.1153
15	0.4933	1.6913	1.0500	0.4190	40	matches best known			
16	0.0716	1.2885	0.7373	0.3593	41	3.0759	3.6108	3.3567	0.2225
17	0.4717	1.0108	0.7682	0.1761	42	2.8253	3.7321	3.3136	0.3500
18	0.7802	2.2757	1.5735	0.4255	43	3.3815	4.1916	3.6703	0.2956
19	0.5650	1.1299	0.9102	0.2180	44	1.1753	2.1221	1.6259	0.3334
20	0.8171	1.5085	1.1754	0.2192	45	2.9570	3.5282	3.1855	0.2225
21	1.4802	2.1332	1.8459	0.2971	46	1.9295	2.7944	2.3353	0.3752
22	1.2863	2.8109	1.8866	0.5808	47	1.8429	2.1662	2.0498	0.1374
23	1.2898	2.2786	1.7713	0.3783	48	1.5805	2.0086	1.8110	0.1696
24	1.4845	1.8444	1.6464	0.1583	49	1.8985	3.0376	2.4094	0.4898
25	1.4404	2.3134	1.8856	0.3512	50	2.5449	3.0669	2.8711	0.2139

Small-scale problems (Ta1–Ta10) In the case of Ta1–Ta10, the performance of the algorithm gave extremely favorable results. In several cases, the minimum, maximum and average values are the same. For most individuals, the algorithm resulted in a GAP of 0%. There was a smaller difference only in the case of Ta3, Ta4, Ta5 and Ta7. Here, the average GAP is between 0.016–0.4%.

Medium-sized problems (Ta11–Ta20) In this range, the GAP values are already higher. The average GAP is typically between 1–2%. The largest average

difference is observed in the case of Ta13 and Ta18. Here the difference is approximately 1.5%. The minimum GAP in most cases is between 0.3–0.8%.

Large-scale problems (Ta21–Ta30) For these data sets, the average GAP values are around 2% in most cases. In the case of Ta29 and Ta30, the average difference already exceeds 2%. The minimum GAP is already above 1% for several instances. In a larger search space, the result of the Hybrid RBVAS algorithm is therefore further away from the best known result.

Large-scale problems (Ta31–Ta40) Here again an improving trend can be observed. In several instances, the algorithm resulted in a minimum GAP value of 0% (e.g. Ta36, Ta37, Ta38). The average GAP values typically ranged between 0–0.4%. The stability of the algorithm is good, the standard deviation is moderate.

Largest size problems (Ta41–Ta50) Here again, a significant performance degradation is observed. The average GAP values were between 2–3.6% in the most cases. In the case of Ta43, the average deviation reached 3.67%. The maximum GAP exceeds 3% in several run cases regarding Ta41–Ta50 test instances.

Summary evaluation The performance of the Hybrid Rank Based Version of Ant System gave optimal or near-optimal results for small-scale tasks. Here, the GAP values were between 0–0.4% in most of the cases. For medium-sized tasks, the runs had a GAP value of around 1–2%. For Ta21–Ta30 individuals, the GAP was already 1.5–2.8% in most of the test runs, while for Ta31–Ta40 the algorithm is effective (0–0.45% GAP results).

3.4. Iterated Simulated Annealing algorithm results

The Table 3 presents the performance of the Iterated Simulated Annealing (ISA) algorithm on the Taillard benchmark problems. The table includes the minimum, maximum, and average fitness values, as well as their GAP values compared to the best known solutions.

Small-scale problems (Ta1–Ta10) For most instances (Ta1–Ta3, Ta5–Ta6, Ta8–Ta10) all runs reached the best known solution. In the case of Ta4 and Ta5, minimal deviations are observed. Here the average GAP is 0.10% and 0.4%. For small problems, the algorithm provides near-optimal solutions. The standard deviation here was low.

Medium-sized problems (Ta11–Ta30) In the case of medium-sized problems, some deterioration can already be observed. However, the difference was mostly around 1–2%. In the case of Ta11–Ta20, the average GAP values were between 0.7–1.7%. The maximum GAP in several cases exceeds 2%. The min–max differences are smaller, which means a relatively stable search. In the case of Ta21–Ta30, the

Table 3. Statistical results of Ta for ISA.

Ta	Min	Max	Avg	RSD	Ta	Min	Max	Avg	RSD
1	matches best known				26	0.8985	1.4376	1.2129	0.1935
2	matches best known				27	1.4078	2.3757	1.9358	0.3713
3	matches best known				28	1.3182	2.2727	1.9545	0.3622
4	0.0000	0.3867	0.1005	0.1631	29	1.0729	2.2798	1.8328	0.5240
5	matches best known				30	1.5611	3.3517	2.4793	0.7583
6	matches best known				31	matches best known			
7	0.4052	0.4052	0.4052	0.0000	32	0.1411	0.4940	0.2541	0.1438
8	matches best known				33	0.0000	0.1145	0.0458	0.0418
9	matches best known				34	0.0727	0.0727	0.0727	0.0000
10	matches best known				35	matches best known			
11	0.4425	2.0228	1.4349	0.4513	36	0.0000	0.0707	0.0566	0.0316
12	1.0247	1.7480	1.3804	0.2179	37	0.0000	0.2569	0.0514	0.1148
13	0.8690	2.2727	1.7380	0.3321	38	0.1118	0.4100	0.2758	0.1330
14	0.5084	1.5977	1.2128	0.3945	39	0.1176	0.3527	0.2351	0.1173
15	0.3524	2.0437	1.1557	0.5246	40	matches best known			
16	0.2863	1.5032	0.8661	0.3850	41	2.6747	3.3768	3.0425	0.2482
17	0.1348	1.3477	0.7480	0.4776	42	2.3369	3.7670	3.0206	0.5431
18	1.2354	1.8856	1.5670	0.2327	43	2.7827	4.0155	3.5717	0.4825
19	0.1255	1.1927	0.8537	0.3374	44	1.2733	1.6650	1.4757	0.1481
20	0.4400	1.6342	1.1691	0.4444	45	2.8562	3.5618	3.1519	0.3236
21	1.2625	2.2638	1.6282	0.3815	46	2.2289	2.7611	2.5815	0.2059
22	1.2863	2.3821	1.7723	0.4019	47	1.7459	2.1662	1.9657	0.1868
23	1.0318	2.4076	1.7885	0.5012	48	1.0207	1.8110	1.5081	0.2914
24	1.5744	2.4741	1.8893	0.3532	49	2.2437	3.0031	2.6372	0.3292
25	1.7896	2.0952	1.9380	0.1502	50	2.4144	2.9690	2.6884	0.2042

average GAP values were typically between 1.1–2.4%. The largest difference was in the case of Ta30, here the average GAP was 2.4%, the maximum GAP was 3.3%.

Large-scale problems (Ta31–Ta50) For the Ta31–Ta40 datasets, the algorithm gave 0% or very low deviations from the best known solution in several cases. The average GAP values were typically between 0–0.28%. Stability is favorable in this range. The min–max differences are small. A significant performance decrease can be observed for the Ta41–Ta50 datasets. The average GAP values typically resulted in values between 1.4–3.5%. The largest average deviation was in the case of Ta43, where the average GAP was 3.57%. For several instances, such as Ta41, Ta42, Ta43, Ta45 the average deviation exceeds 3%. For larger problems, ISA shows a significant difference compared to the best known solutions. Here, the standard deviation also increases significantly.

Stability testing For small data sets, the minimum and maximum GAP values are often the same or the difference between them is very small. Here, the standard deviation is very low. For medium problems, moderate standard deviation is observed. For large problems, the difference between the minimum and maximum GAP increases.

Summary evaluation The algorithm provided stable, near-optimal solutions on small-scale problems. For medium-scale problems, the average deviation was

between 1–2%. For large-scale problems, the deviation from the best solution was between 2–3.6%, and the standard deviation was much larger.

3.5. J. Carlier and C.R. Reeves benchmarks

Test runs were performed not only on the Taillard data set, but also on the J. Carlier (car) [8] and C.R. Reeve (rec) [18] data sets. This can be seen in Table 4.

Table 4. Car and Rec instances (HACS, HRBVAS, ISA comparison).

Alg	Inst	Min	Max	Avg	RSD	Alg	Inst	Min	Max	Avg	RSD
HACS	car1	matches best known				ISA	car1	matches best known			
HACS	car2	matches best known				ISA	car2	matches best known			
HACS	car3	matches best known				ISA	car3	matches best known			
HACS	car4	matches best known				ISA	car4	matches best known			
HACS	rec1	0.1604	0.1604	0.1604	0.0000	ISA	rec1	0.0000	0.1604	0.1283	0.0676
HACS	rec3	matches best known				ISA	rec3	matches best known			
HACS	rec13	1.2435	2.0725	1.5492	0.2779	ISA	rec13	1.0363	2.0725	1.5492	0.3222
HACS	rec19	1.3856	2.2458	1.9546	0.2736	ISA	rec19	1.3850	2.2940	1.9160	0.2746
HRBVAS	car1	matches best known									
HRBVAS	car2	matches best known									
HRBVAS	car3	matches best known									
HRBVAS	car4	matches best known									
HRBVAS	rec1	0.1604	0.1604	0.1604	0.0000						
HRBVAS	rec3	matches best known									
HRBVAS	rec13	0.8808	1.8135	1.3980	0.3196						
HRBVAS	rec19	1.4335	2.2937	1.8968	0.2720						

The table also shows that in many case the algorithms reached the best known value during each run. Although they did not reach the best known value during the most complex test data, but they were close to it.

3.6. Summary of the test results

This section presents a comparison of the Hybrid Ant Colony System, the Hybrid Rank Based Version of Ant System and the Iterated Simulated Annealing algorithms. First, the test results are summarized in a table. The Table 5 contains the following columns: Taillard benchmark dataset identifier, the best algorithm and the minimum fitness value of the best algorithm.

When examining the performance of the three metaheuristic algorithms on the Taillard benchmark dataset, it can be observed that the algorithms were able to achieve the best known solution in several cases. On the first ten datasets (except Ta07), all three methods achieved the best known value. For larger problems, the Hybrid Ant Colony System (HACS) gave better results on several datasets. The HACS algorithm gave the best result in 13 times, the Hybrid Rank Based Version of Ant System in 7 times, and the Iterated Simulated Annealing in 14 times. In addition, the algorithms provided identical results in 17 times.

Table 5. Summary of best algorithm and minimum values for Ta instances.

Ta	Best	Min	Ta	Best	Min
Ta1	same	1278	Ta26	ISA	2246
Ta2	same	1359	Ta27	ISA	2305
Ta3	same	1081	Ta28	HACS	2215
Ta4	same	1293	Ta29	ISA	2261
Ta5	same	1235	Ta30	HACS	2205
Ta6	same	1195	Ta31	same	2724
Ta7	same	1239	Ta32	same	2838
Ta8	same	1206	Ta33	HACS=ISA	2621
Ta9	same	1230	Ta34	same	2753
Ta10	same	1108	Ta35	same	2863
Ta11	ISA	1589	Ta36	same	2829
Ta12	HRBVAS	1664	Ta37	same	2725
Ta13	HRBVAS = ISA	1509	Ta38	HACS=HRBVAS	2683
Ta14	HACS=ISA	1384	Ta39	HRBVAS	2554
Ta15	ISA	1424	Ta40	same	2782
Ta16	HRBVAS	1398	Ta41	ISA	3071
Ta17	ISA	1486	Ta42	ISA	2934
Ta18	HACS=HRBVAS	1550	Ta43	ISA	2918
Ta19	ISA	1595	Ta44	HACS	3084
Ta20	HACS	1591	Ta45	HACS	3049
Ta21	ISA	2326	Ta46	HRBVAS	3064
Ta22	HRBVAS=ISA	2126	Ta47	HACS	3143
Ta23	ISA	2350	Ta48	HACS	3055
Ta24	HACS	2249	Ta49	HACS	2945
Ta25	HACS	2320	Ta50	ISA	3139

4. Summary

In this article, a well-known production scheduling problem, the Flow Shop Scheduling, was investigated. The problem was solved with three algorithms, namely Hybrid Ant Colony System, Hybrid Rank Based Version of Ant System and Iterated Simulated Annealing. Hybrid ACS and Hybrid RBVAS mean that the algorithm also applies an improvement with the Simulated Annealing algorithm. After the introduction of the Flow Shop task and the most important publications on the topic, the article presented the applied algorithms. This was followed by the presentation and evaluation of the test results. The summary showed that the Hybrid Ant Colony System algorithm gave the best result in 13 times on the Taillard dataset, Hybrid Rank Based Version of Ant System in 7 times, and Iterated Simulated Annealing in 14 times. In addition, the algorithms gave the same results in 17 times. The effective hybridization of other metaheuristic algorithms with the help of Simulated Annealing could be a further research topic. In addition, testing other optimization problems, such as the Traveling Salesman Problem or the Vehicle Routing Problem, as well-known logistics discrete optimization problems, may also become a research focus.

References

- [1] A. AGÁRDI: *Application of the Max-Min Ant System on Flow Shop Scheduling Problems*, Production Systems and Information Engineering 13.1 (2025), pp. 1–13.

- [2] A. AGÁRDI: *Efficiency analysis of the Ant System algorithm on the Flow Shop Scheduling Problem*, Production Systems and Information Engineering 12.1 (2024), pp. 29–40.
- [3] A. AGÁRDI: *Efficiency analysis of the Hill Climbing and Elitist Strategy of Ant System in the application of Flow Shop Scheduling Problems*, Production Systems and Information Engineering 13.2 (2025), pp. 1–17.
- [4] A. AGÁRDI: *Rank-Based Version of Ant System in production scheduling*, Production Systems and Information Engineering 13.1 (2025), pp. 14–25.
- [5] A. AGÁRDI, K. NEHÉZ, O. HORNYÁK, L. T. KÓCZY: *A hybrid discrete bacterial memetic algorithm with simulated annealing for optimization of the flow shop scheduling problem*, Symmetry 13.7 (2021), p. 1131.
- [6] D. BERTSIMAS, J. TSITSIKLIS: *Simulated annealing*, Statistical science 8.1 (1993), pp. 10–15.
- [7] B. BULLNHEIMER, R. HARTL, C. STRAUSS: *A new rank based version of the ant system: A computational study* (1997).
- [8] J. CARLIER: *Ordonnancements a contraintes disjonctives*, RAIRO-Operations Research 12.4 (1978), pp. 333–350.
- [9] I. CHAOUCH, O. B. DRISS, K. GHEDIRA: *A modified ant colony optimization algorithm for the distributed job shop scheduling problem*, Procedia Computer Science 112 (2017), pp. 296–305.
- [10] Z. CHEN, X. ZHENG, S. ZHOU, C. LIU, H. CHEN: *Quantum-inspired ant colony optimisation algorithm for a two-stage permutation flow shop with batch processing machines*, International Journal of Production Research 58.19 (2020), pp. 5945–5963.
- [11] S. DAUZÈRE-PÉRÈS, J. DING, L. SHEN, K. TAMSSAOUET: *The flexible job shop scheduling problem: A review*, European Journal of Operational Research 314.2 (2024), pp. 409–432.
- [12] G. KOMAKI, S. SHEIKH, B. MALAKOOTI: *Flow shop scheduling problems with assembly operations: a review and new trends*, International Journal of Production Research 57.10 (2019), pp. 2926–2955.
- [13] X. LI, M. F. BAKI, Y. P. ANEJA: *Flow shop scheduling to minimize the total completion time with a permanently present operator: Models and ant colony optimization metaheuristic*, Computers & Operations Research 38.1 (2011), pp. 152–164.
- [14] C.-L. LIU, T.-H. HUANG: *Dynamic job-shop scheduling problems using graph neural network and deep reinforcement learning*, IEEE transactions on systems, man, and cybernetics: systems 53.11 (2023), pp. 6836–6848.
- [15] H. OKANO, S. MISONO, K. IWANO: *New TSP construction heuristics and their relationships to the 2-opt*, Journal of Heuristics 5.1 (1999), pp. 71–88.
- [16] Y. PAN, K. GAO, Z. LI, N. WU: *Solving biobjective distributed flow-shop scheduling problems with lot-streaming using an improved Jaya algorithm*, IEEE transactions on cybernetics 53.6 (2022), pp. 3818–3828.
- [17] C. QU, Y. FU, Z. YI, J. TAN: *Solutions to No-Wait Flow Shop Scheduling Problem Using the Flower Pollination Algorithm Based on the Hormone Modulation Mechanism*, Complexity 2018.1 (2018), p. 1973604.
- [18] C. R. REEVES: *A genetic algorithm for flowshop sequencing*, Computers & operations research 22.1 (1995), pp. 5–13.
- [19] D. A. ROSSIT, F. TOHMÉ, M. FRUTOS: *The non-permutation flow-shop scheduling problem: a literature review*, Omega 77 (2018), pp. 143–153.
- [20] S. SHI, H. XIONG: *Solving the multi-objective job shop scheduling problems with overtime consideration by an enhanced NSGA-II*, Computers & Industrial Engineering 190 (2024), p. 110001.
- [21] T. STÜTZLE, M. DORIGO, ET AL.: *ACO algorithms for the traveling salesman problem*, Evolutionary algorithms in engineering and computer science 4 (1999), pp. 163–183.

- [22] E. TAILLARD: *Benchmarks for basic scheduling problems*, european journal of operational research 64.2 (1993), pp. 278–285.
- [23] L.-Y. TSENG, Y.-T. LIN: *A hybrid genetic algorithm for no-wait flowshop scheduling problem*, International journal of production economics 128.1 (2010), pp. 144–152.
- [24] D. M. UTAMA, C. N. AL IMRON: *A systematic literature review on no-idle flow shop scheduling problem*, in: Operations Research Forum, vol. 5, 1, Springer, 2024, p. 18.
- [25] D. M. UTAMA, S. Z. UMAMY, C. N. AL-IMRON: *No-Wait Flow Shop scheduling problem: a systematic literature review and bibliometric analysis*, RAIRO-Operations Research 58.2 (2024), pp. 1281–1313.
- [26] Z. WANG, B. CAI, J. LI, D. YANG, Y. ZHAO, H. XIE: *Solving non-permutation flow-shop scheduling problem via a novel deep reinforcement learning approach*, Computers & Operations Research 151 (2023), p. 106095.
- [27] H. WEI, S. LI, H. JIANG, J. HU, J. HU: *Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion*, Applied Sciences 8.12 (2018), p. 2621.
- [28] K.-C. YING, C.-J. LIAO: *An ant colony system for permutation flow-shop sequencing*, Computers & Operations Research 31.5 (2004), pp. 791–801.
- [29] A. N. H. ZAIED, M. M. ISMAIL, S. S. MOHAMED: *Permutation flow shop scheduling problem with makespan criterion: literature review*, J. Theor. Appl. Inf. Technol 99.4 (2021), pp. 830–848.
- [30] F. ZHAO, S. DI, L. WANG: *A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem*, IEEE Transactions on Cybernetics 53.5 (2022), pp. 3337–3350.
- [31] Y. ZHOU, H. CHEN, G. ZHOU: *Invasive weed optimization algorithm for optimization no-idle flow shop scheduling problem*, Neurocomputing 137 (2014), pp. 285–292.



LLM-Augmented Machine Learning for Phishing Email Detection: Enhancing Classification, Explainability, and Multilingual Support*

Farouk Aziz[†], Norbert Oláh

University of Debrecen, Faculty of Informatics, Debrecen, Hungary
farouk.aziz@mailbox.unideb.hu, olah.norbert@inf.unideb.hu

Abstract. Nowadays, email is the world’s primary communication channel, making it a dominant vector for phishing attacks that exploit urgency, authority, and trust. Advances in Large Language Models (LLMs) have intensified this threat by enabling attackers to generate persuasive, context-aware, and multilingual phishing emails at low cost. Traditional machine learning (ML) approaches are no longer sufficient, as phishing techniques have evolved with the widespread public use of AI. These systems also face the scarcity of datasets for non-English content and the lack of human-friendly explanations of model decisions. Large Language Models can address these limitations, but are costly and impractical for large-scale deployment given that billions of emails are sent daily. This paper proposes a selective hybrid framework combining ML with LLMs in a cost-aware architecture. LLMs are invoked selectively to correct ML mistakes, translate non-English emails for a single English-trained classifier, and generate on-demand human-readable explanations of model decisions. GPT-4, Claude, and DeepSeek were independently evaluated across these three tasks to identify which performs best and fastest. The ML mistakes were forwarded to each LLM using task-specific prompts, translation was assessed across 18 languages, and explainability was evaluated by converting Local Interpretable Model-agnostic Explanations (LIME) feature attributions into natural language and measuring readability. A Term Frequency–Inverse Document Frequency (TF-IDF) based Linear Support Vector Machine (SVM) trained on approximately 14,000 English

*This research was conducted for the 13th International Conference on Applied Informatics.

[†]Corresponding authors: farouk.aziz@mailbox.unideb.hu, olah.norbert@inf.unideb.hu

emails serves as the baseline classifier. The results confirm improvements in classification robustness, multilingual coverage, and explainability, while keeping computational overhead well below universal LLM pipelines. The three LLMs showed distinct strengths across tasks, highlighting that model selection should be tailored to each role within the architecture.

Keywords: phishing detection, large language models, machine learning, selective hybrid architecture, explainability, multilingual classification

2020 Mathematics Subject Classification: Primary 68T50; Secondary 68T05, 94A60

1. Introduction

Email is the dominant attack vector for cybercriminals. Phishing accounts for approximately 90% of data breaches, with an average loss of \$3.86 million per incident [5]. Attacks take several distinct forms. Bulk phishing sends identical lure messages to millions of recipients simultaneously, relying on volume rather than personalisation – a single convincing template can yield thousands of victims. Spear phishing targets specific individuals by incorporating personalised details such as the recipient’s name, role, organisation, or recent activity, making the message appear to come from a trusted colleague or institution. Whaling is a high-value variant of spear phishing directed exclusively at senior executives, board members, or regulators, typically impersonating legal counsel, auditors, or government authorities to authorise large transfers or disclose confidential data. Business Email Compromise (BEC) involves hijacking or spoofing a legitimate corporate email account to redirect payments, alter supplier banking details, or extract sensitive information, causing average losses exceeding \$125,000 per incident [1, 16]. Beyond these established categories, AI-generated phishing has emerged as a new threat dimension: attackers leveraging LLMs can now produce grammatically flawless, contextually convincing, and personalised messages in any language at negligible cost, effectively eliminating the surface errors – misspellings, awkward phrasing, generic salutations – that rule-based and classical ML detectors have historically relied upon [15, 16].

Existing defensive systems either apply LLMs uniformly to all inputs [7] or use preprocessing filters that still route every non-trivial email through expensive inference [20]. While these systems are described as hybrid, they rely solely on LLMs for the actual classification decision, making the “hybrid” label a description of their pipeline structure rather than a reflection of combining fundamentally different classification approaches.

2. Scientific Literature

Phishing detection has progressed from static blacklists and rule-based filters through classical ML, deep learning, and transformer-based systems to current

adaptive multimodal approaches, as illustrated in Figure 1 [18]. These solutions are discussed in the following sections.

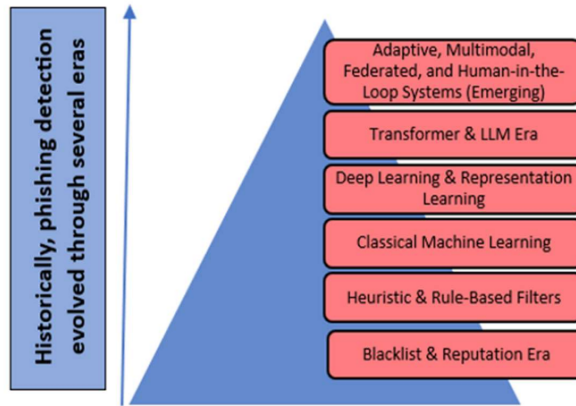


Figure 1. Evolution of phishing detection techniques, from blacklist and rule-based systems to adaptive multimodal approaches [18].

2.1. Classical Machine Learning Approaches

Supervised ML with Term Frequency–Inverse Document Frequency (TF-IDF) feature extraction and classifiers such as SVMs, Random Forests, and Logistic Regression established the first generation of scalable phishing detectors [12, 18]. Machine learning algorithms facilitate adaptive classification rules that outperform blacklisting or rule-based filtering, which rely on hand-coded rules susceptible to the changing nature of phishing attacks [12]. TAWIL ET AL. [26] found TF-IDF with Random Forest and Logistic Regression achieved F1 of 0.98 at high throughput. RASTENIS ET AL. [23] demonstrated Linear SVM with TF-IDF scales to multilingual datasets, noting its linear complexity as a deployment advantage. Despite their speed, classical models struggle with noisy or dynamic datasets [24] and fail entirely on non-English content [20].

2.2. Large Language Model-Based and Hybrid Approaches

LLM-based systems address the semantic limitations of classical ML but introduce prohibitive inference costs. LLMs require significant GPU resources, with training times ranging from hours to days, and API costs for processing 8 million tokens reaching \$36–\$128 depending on the model [13, 25]. DESOLDA, GRECO, VIGANÒ [7] proposed APOLLO, which integrates GPT-4o with VirusTotal and BigData-Cloud APIs, achieving 97–99% detection accuracy and generating user-readable explanations through prompt chaining. LIM ET AL. [21] introduced EXPLICATE, combining a classical classifier with LIME/SHAP attributions and DeepSeek v3 to

translate feature weights into natural-language warnings, reaching 94.2% explanation accuracy. LEE [20] evaluated four open-source LLMs combining whitelisting and HTML parsing before LLM inference, improving throughput but still routing every non-whitelisted email through costly API calls. TARAPIAH ET AL. [25] fine-tuned three transformer variants – DistilBERT (a lightweight, distilled version of BERT that retains 97% of its language understanding at 60% of the size), ALBERT (a parameter-efficient BERT variant using cross-layer parameter sharing to reduce memory footprint), and RoBERTa (a robustly optimised BERT with longer training and dynamic masking) – confirming high accuracy but at GPU-level inference cost. KUMAR, KUSHWAHA, RAJPOOT [18] confirm that hybrid ensemble approaches dominate production deployments and transformer-era models are the first to achieve meaningful multilingual generalisation. The critical gap is that LLMs are invoked uniformly rather than selectively across all prior systems.

Two further limitations remain unresolved: English-trained models drop below 60% accuracy on non-English emails [18, 23] with no general multilingual phishing dataset available [2]; and explainability tools such as LIME and SHAP require technical expertise to interpret, while LLM-generated explanations have only been evaluated universally rather than on demand [7, 21]. No prior system simultaneously addresses selective deployment, multilingual translation, and on-demand explainability within a single cost-aware architecture. DESOLDA, GRECO, VIGANÒ [7] achieve 97–99% accuracy with APOLLO but process all emails through GPT-4o, making real-time high-volume deployment economically challenging. LEE’s system [20] reduces overhead through preprocessing but still applies LLM inference to every non-whitelisted email. LIM ET AL. [21] demonstrate comparable explanation quality with EXPLICATE but generate explanations universally rather than on demand, incurring multi-second overhead per email. The selective hybrid architecture proposed in this work goes further by using calibrated uncertainty as a routing signal, restricting LLM invocation to the cases where classical ML demonstrably fails.

2.3. Our Contribution

This paper proposes a selective hybrid architecture that exploits the heterogeneous difficulty of email classification, restricting LLM invocation to three well-defined failure points. Table 1 summarises how the proposed system compares to prior approaches across key evaluation dimensions.

Table 1. Comparison of system capabilities across key dimensions.

System	Efficiency	Accuracy	Multilingual	Explainability
ML only (SVM)	High	Moderate	No	Partial
LLM only (APOLLO [7])	Low	High	Yes	Yes
EXPLICATE [21]	Moderate	High	No	Yes
Proposed (Selective)	High	High	Yes	Yes

The main contributions are: (1) a selective architecture invoking LLMs only for the most uncertain misclassifications (calibrated confidence near 0.5); (2) on-demand LLM translation of non-English emails, improving accuracy from 50% to 75% across 18 languages; (3) an on-demand explainability module converting Local Interpretable Model-agnostic Explanations (LIME) attributions into plain-language warnings; and (4) a head-to-head evaluation of GPT-4, Claude 3.5 Sonnet, and DeepSeek across all three tasks.

3. Methodology

3.1. Machine Learning Classification Pipeline

The pipeline requires a classifier with high accuracy on English email text, calibrated probability estimates for uncertainty routing, and feature-level attributions for explainability. Raw emails are preprocessed via tokenization, stop-word removal, lowercasing, stemming, and URL removal [12, 19]. A Linear SVM with TF-IDF vectorization was selected after evaluating neural, tree-based, and linear alternatives [18, 26].

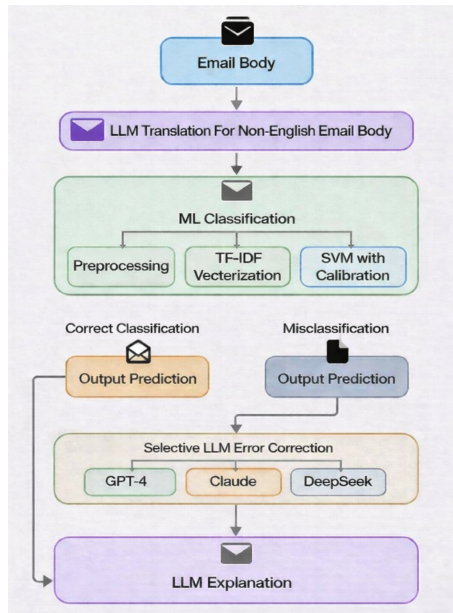


Figure 2. Selective hybrid architecture: ML processes all emails; LLMs are invoked only for uncertain misclassifications, non-English content, and on-demand explanations.

Figure 2 illustrates the overall architecture. TF-IDF (up to 10,000 features, n-gram range 1–3, min/max document frequency 2/0.95) was chosen for its sub-

millisecond CPU inference on sparse matrix representations, directly interpretable feature weights compatible with LIME, and natural emphasis on rare discriminative phishing terms [14, 26]. N-gram features capture contextual phrasal cues such as “verify your account” that distinguish phishing from legitimate messages [17]. SVM was selected for its strong generalisation in high-dimensional sparse spaces [18, 22, 26] and, critically, its calibration via Platt scaling (CalibratedClassifierCV, 5-fold) which transforms the decision function into posterior probabilities [10] – the mechanism enabling selective routing:

$$P(y = 1 \mid x) = \sigma(A \cdot f(x) + B)$$

where σ is the sigmoid function, $f(x)$ the SVM decision function, and A, B are learned calibration parameters. A well-calibrated classifier correctly quantifies the level of uncertainty associated with its predictions, which is essential for cost-sensitive classification and optimal decision making [10].

3.2. Selective Large Language Model Augmentation

The three LLM components share a common principle. LLM invocation occurs only when and where it provides measurable value over the ML baseline.

Error correction. The SVM assigns a calibrated posterior probability $P(y = 1 \mid x)$ to every email. An email is considered uncertain when this probability falls within the band $[0.5 - \delta, 0.5 + \delta]$ for a configurable threshold δ , meaning the model has low confidence in either class. The k most uncertain ML misclassifications – those with calibrated confidence closest to 0.5 – are routed to LLM re-classification, where k is a budget parameter controlling the accuracy–cost trade-off:

$$\text{Confidence}(x) = \max_c P(y = c \mid x)$$

Ambiguous cases benefit from the LLM’s ability to reason about intent, tone, and deceptive context that lexical features miss [20, 25].

Multilingual translation. Non-English emails are routed to an LLM using a phishing-aware prompt that explicitly instructs the model to preserve urgency markers, grammatical errors, and suspicious phrasing rather than produce fluent translations. This is deliberate: these surface anomalies are the diagnostic signals the English-trained SVM relies on for classification [23]. Translation is performed on demand rather than universally, avoiding the per-email latency and cost of preprocessing all incoming email [27].

On-demand explainability. LIME feature attributions are generated for the ML prediction and passed to an LLM only when a user requests an explanation. Local Interpretable Model-agnostic Explanations (LIME) is an explainability technique that approximates any classifier locally with a simple interpretable model, identifying which input features most influenced a specific prediction [21]. SHapley Additive exPlanations (SHAP) is an alternative that assigns each feature a contribution score derived from cooperative game theory, measuring its average impact across all possible feature combinations [21]. LIME was chosen over SHAP for

its local operation on individual predictions and direct compatibility with the TF-IDF vocabulary, enabling the LLM to map numerical weights to concrete phishing tactics in accessible natural language [7, 21].

3.3. Large Language Model Provider Selection

GPT-4 (openai/gpt-4.1), Claude 3.5 Sonnet (anthropic/claude-3.5-sonnet), and DeepSeek (deepseek/deepseek-chat-v3.1) were selected as leading commercial models with strong multilingual pretraining and documented reasoning performance [13, 16]. Independent benchmarking by Artificial Analysis [3] confirms that these three providers represent distinct positions on the intelligence–speed–cost frontier, making them well-suited for comparative evaluation across latency-sensitive and accuracy-critical tasks. All API calls were made via OpenRouter at temperature 0.3 [13, 25].

4. Experimental Design

ML Training Dataset: 17.496 English emails sourced from the Phishing Email Collection on Kaggle [9] (after removing 1.154 duplicates from 18.650 originals), split 80/20 into training (13.996) and held-out test (3.500) sets using stratified sampling.

Hybrid Test Dataset: 16.478 emails from the Phishing Emails dataset on HuggingFace [8], completely separate from training. Used to assess real-world generalization and evaluate selective LLM error correction.

Multilingual Dataset: 36 manually curated emails across 18 languages (Hungarian, Spanish, French, German, Italian, Russian, Japanese, Portuguese, Polish, Dutch, Swedish, Turkish, Arabic, Korean, Czech, Greek, Norwegian, Finnish), one phishing and one legitimate per language, each reflecting culturally specific phishing tactics.

Model performance was assessed using 10-fold stratified cross-validation [11]. Cross-validation is a resampling technique that partitions the dataset into k equally sized folds; the model is trained on $k - 1$ folds and evaluated on the remaining fold, repeating this process k times so that every sample is used for evaluation exactly once. Using $k = 10$ is a standard choice that balances bias and variance while preserving class distribution across folds. Stability was confirmed across ten random data splits.

Explainability quality was quantified using the Flesch Reading Ease score [6], a standardised readability metric scored on a 0–100 scale. Scores in the range 70–80 correspond to text readily understood by a 7th-grade student, scores of 60–70 indicate standard readability accessible to an 8th–9th grade reader, scores of 50–60 reflect content suited to a high school student, scores of 30–50 indicate college-level difficulty, and scores below 30 describe highly technical or professional text typically requiring advanced education to comprehend. This metric was used to measure the accessibility of LLM-generated explanations for non-expert users.

5. Results and Analysis

5.1. ML Baseline

The SVM model achieved 98.37% accuracy on the held-out test set (F1: 0.9781, ROC-AUC: 0.9986) with training time of 3.13 seconds and inference at 0.34 ms per email. On the independent hybrid test dataset, the model processed all 16.478 emails in 5.55 seconds, achieving 90.30% accuracy (1.598 misclassifications), reflecting realistic domain shift between training and deployment data.

5.2. Selective Error Correction

The 1.598 misclassified emails were ranked by uncertainty and the 250 most uncertain cases were selected for LLM review. In this evaluation, ground truth labels were available to identify confirmed misclassifications; in production deployment, the system would instead route all predictions below a configurable confidence threshold to LLM review, with the optimal threshold determined empirically per deployment context. Table 2 presents the correction results across all three providers, and Figure 3 visualises the accuracy and number of mistakes corrected.

Table 2. LLM Error Correction on 250 Most Uncertain ML Misclassifications.

Model	Accuracy	Fixed	Avg Time/Email
Claude 3.5 Sonnet	93.60%	234/250	2.06 s
GPT-4	68.00%	170/250	0.67 s
DeepSeek	57.94%	135/250	14.46 s

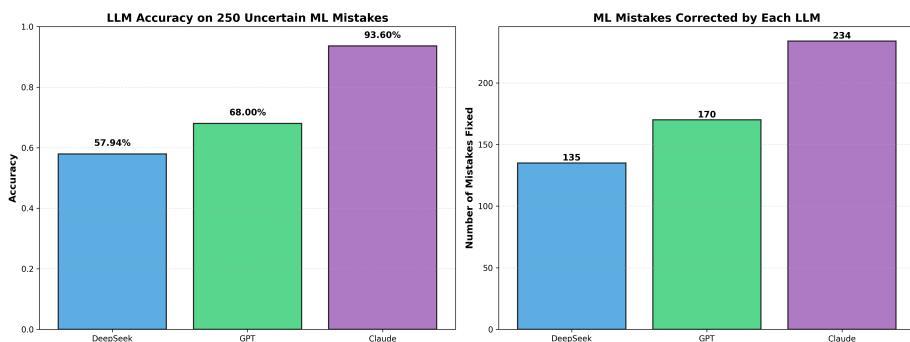


Figure 3. LLM accuracy and number of mistakes corrected on 250 most uncertain ML misclassifications.

Claude 3.5 Sonnet achieved the highest correction accuracy (93.60%, 234/250), while GPT-4’s lower accuracy (68%) is offset by its 3× faster response time, making

it preferable for latency-sensitive deployments. The wide performance gap (57.94–93.60%) confirms that semantic reasoning about deceptive intent varies substantially across LLM architectures [16].

5.3. Multilingual Translation

The multilingual dataset was designed for representative coverage across diverse language families rather than volume, balancing evaluation breadth against the per-call API costs inherent to LLM-based processing. Table 3 reports classification performance across all translation configurations, and Figure 4 compares F1 scores across providers.

Table 3. Multilingual Classification Performance (36 Emails, 18 Languages).

Approach	Accuracy	Recall	F1	Time/Email
ML Only (Raw)	50.0%	66.7%	0.571	6.83 ms
ML + GPT-4	63.9%	83.3%	0.698	847 ms
ML + Claude 3.5	58.3%	88.9%	0.681	2.033 ms
ML + DeepSeek	75.0%	94.4%	0.791	2.611 ms

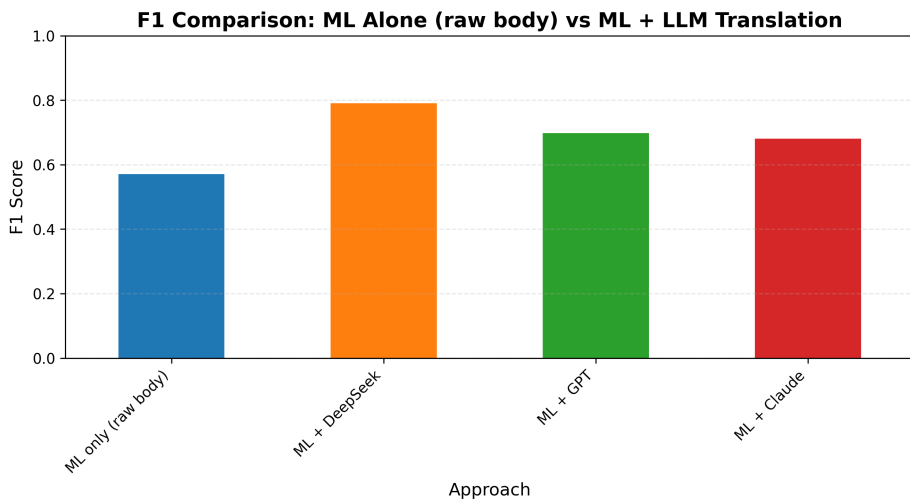


Figure 4. F1 score comparison: ML alone vs ML with LLM translation across three providers.

Without translation, the ML baseline scored 50% – equivalent to random guessing. DeepSeek achieved the best results (75% accuracy, 94.4% recall, F1 0.791), a 38% relative F1 improvement over baseline, while GPT-4 offered a favorable speed–accuracy trade-off at 847 ms per email. High recall is critical: missed phishing emails reach users [27].

5.4. Explainability Enhancement

This evaluation prioritised qualitative depth and comparison of explanation quality across providers over large-scale quantitative evaluation, reflecting practical API cost constraints for a three-provider comparison. Table 4 presents the readability scores and latency of LLM-generated explanations across all three providers, and Figure 5 visualises the mean Flesch scores.

Table 4. Readability of LLM-Generated Explanations (5 Emails).

Model	Mean Flesch	Std Dev	Time/Explanation
DeepSeek	65.59	6.65	6.72 s
GPT-4	62.70	14.67	4.15 s
Claude 3.5 Sonnet	32.13	16.05	7.04 s

DeepSeek achieved the highest mean Flesch Reading Ease score (65.59, 8th–9th grade) with the lowest variance (6.65), indicating consistent accessibility across email types. GPT-4 produced comparable readability (62.70) at the lowest latency (4.15 s). Claude’s low score (32.13, college level) suggests it prioritizes thoroughness over simplicity. All three providers successfully transformed LIME feature-weight pairs into plain-language warnings [21].

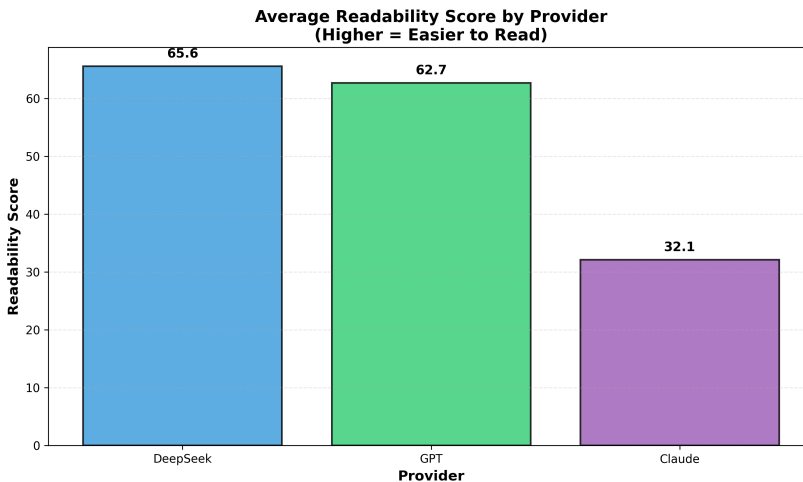


Figure 5. Average Flesch Reading Ease scores by LLM provider. Higher scores indicate more accessible explanations for non-expert users.

6. Discussion and Conclusion

Architectural Effectiveness

Routing only 1.5% of emails (250 of 16.478) to LLM review preserves ML-level throughput while recovering the most difficult misclassifications. The results confirm the core hypothesis: email classification exhibits heterogeneous difficulty, and restricting LLM invocation to calibrated failure points delivers LLM-level accuracy at a fraction of the cost.

Task-Specific LLM Recommendations

No single provider dominates all tasks: Claude 3.5 Sonnet leads on error correction (93.60%), DeepSeek on translation (75%, F1 0.791) and readability (65.59 Flesch), and GPT-4 on speed across all tasks. The modular architecture allows independent provider selection per task and naturally extends to a multi-agent collaboration paradigm, where specialized agents (one optimized for classification correction, one for translation, one for explanation generation) operate in parallel under a lightweight routing controller [13].

Future Work

Future work should validate thresholding in production, compare LLM translation against language-specific ML models, run user studies on explanation comprehension, and assess prompt injection resilience. Although the current system does not explicitly analyse URLs or attachments, LLMs may still recognise suspicious URL patterns through their broad pretraining knowledge, even without task-specific training on phishing URLs [16]; incorporating raw URL strings and attachment metadata into the LLM prompt, and potentially fine-tuning on phishing-specific datasets, would further strengthen this capability. As selective hybrid architectures of this kind are novel, the present work prioritised breadth of exploration across three distinct tasks and three LLM providers rather than depth of scale within any single task; larger-scale multilingual and explainability evaluations represent a natural next step as API costs continue to decrease and access to richer phishing datasets improves.

Conclusion

The proposed architecture demonstrates that targeting LLMs only at ML failure points delivers LLM-level accuracy at a fraction of the cost. This provides a practical path to robust, explainable, and multilingual phishing detection. The implementation is publicly available [4].

Acknowledgements

This work has been supported by the TalentUD Talent Management Programme of the University of Debrecen, Hungary.

References

- [1] A. A. ALSUWAYLIMI: *Enhancing Arabic phishing email detection: A hybrid machine learning based on genetic algorithm feature selection*, International Journal of Advanced Computer Science & Applications 15.8 (2024), DOI: [10.14569/IJACSA.2024.0150826](https://doi.org/10.14569/IJACSA.2024.0150826).
- [2] P. AN, R. SHAFI, T. MUGHOGHO, O. A. ONYANGO: *Multilingual email phishing attacks detection using OSINT and machine learning*, arXiv preprint (2025), arXiv: [2501.08723](https://arxiv.org/abs/2501.08723).
- [3] ARTIFICIAL ANALYSIS: *Artificial Analysis: Independent Analysis of AI Models and API Providers*, Online platform, Accessed: 2025, 2024, URL: <https://artificialanalysis.ai/>.
- [4] F. AZIZ: *Hybrid email classifier research: LLM-augmented machine learning for phishing detection*, GitHub repository, Accessed: 2025, 2025, URL: <https://github.com/FaroukAziz2020/Hybrid-Email-Classifier-Research>.
- [5] P. BOUNTAKAS, K. KOUTROUMPOUCHOS, C. XENAKIS: *A comparison of natural language processing and machine learning methods for phishing email detection*, in: Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES 2021), New York, NY, USA: ACM, 2021, pp. 1–12, DOI: [10.1145/3465481.3469205](https://doi.org/10.1145/3465481.3469205).
- [6] I. CHANIS, A. ARAMPATZIS: *Enhancing phishing email detection with stylometric features and classifier stacking*, International Journal of Information Security 24.1 (2025), p. 15, DOI: [10.1007/s10207-024-00928-7](https://doi.org/10.1007/s10207-024-00928-7).
- [7] G. DESOLDA, F. GRECO, L. VIGANÒ: *APOLLO: A GPT-based tool to detect phishing emails and generate explanations that warn users*, Proceedings of the ACM on Human-Computer Interaction 9.4 (2025), pp. 1–33, DOI: [10.1145/3733049](https://doi.org/10.1145/3733049).
- [8] DUONG TIN: *Phishing Emails*, Hugging Face dataset, Accessed: 2025, 2024, URL: <https://huggingface.co/datasets/duongtino/phishing-email>.
- [9] ETHAN NGUYEN: *Phishing Email Collection*, Kaggle dataset, Accessed: 2025, 2024, URL: <https://www.kaggle.com/datasets/ethanhnguyen/phishing-email-collection>.
- [10] T. S. FILHO, H. SONG, M. PERELLO-NIETO, R. SANTOS-RODRIGUEZ, M. KULL, P. FLACH: *Classifier calibration: A survey on how to assess and improve predicted class probabilities*, Machine Learning 112.9 (2023), pp. 3211–3260, DOI: [10.1007/s10994-023-06336-7](https://doi.org/10.1007/s10994-023-06336-7).
- [11] T. FONTANARI, T. C. FRÓES, M. RECAMONDE-MENDOZA: *Cross-validation strategies for balanced and imbalanced datasets*, in: Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS 2022), Cham: Springer, 2022, pp. 626–640, DOI: [10.1007/978-3-031-21686-2_43](https://doi.org/10.1007/978-3-031-21686-2_43).
- [12] T. GANGAVARAPU, C. D. JAIDHAR, B. CHANDUKA: *Applicability of machine learning in spam and phishing email filtering: Review and approaches*, Artificial Intelligence Review 53.7 (2020), pp. 5019–5081, DOI: [10.1007/s10462-020-09814-9](https://doi.org/10.1007/s10462-020-09814-9).
- [13] G. GOLDENITS, P. KOENIG, S. RAUBITZEK, A. EKEHART: *Small language models for phishing website detection: Cost, performance, and privacy trade-offs*, arXiv preprint (2025), arXiv: [2511.15434](https://arxiv.org/abs/2511.15434).
- [14] E. S. GUALBERTO, R. T. D. SOUSA, T. P. D. B. VIEIRA, J. P. C. L. D. COSTA, C. G. DUQUE: *The answer is in the text: Multi-stage methods for phishing detection based on feature engineering*, IEEE Access 8 (2020), pp. 223529–223547, DOI: [10.1109/ACCESS.2020.3044229](https://doi.org/10.1109/ACCESS.2020.3044229).

- [15] J. HAZELL: *Large language models can be used to effectively scale spear phishing campaigns*, arXiv preprint (2023), arXiv: [2305.06972](#).
- [16] F. HEIDING, S. LERMEN, A. KAO, B. SCHNEIER, A. VISHWANATH: *Evaluating large language models' capability to launch fully automated spear phishing campaigns: Validated on human subjects*, arXiv preprint (2024), arXiv: [2412.00586](#).
- [17] I. KANARIS, K. KANARIS, I. HOUVARIDAS, E. STAMATATOS: *Words versus character n-grams for anti-spam filtering*, International Journal on Artificial Intelligence Tools 16.6 (2007), pp. 1047–1067, DOI: [10.1142/S0218213007003692](#).
- [18] P. KUMAR, C. KUSHWAHA, A. K. RAJPOOT: *Advances in Phishing Detection: A Comprehensive Review of Machine Learning, Deep Learning, and Transformer-based Approaches*, SSRN preprint (2025), Available at SSRN, URL: <https://ssrn.com/abstract=6079506>.
- [19] B. A. KUMARA, M. M. KODABAGI, T. CHOUDHURY, J.-S. UM: *Improved email classification through enhanced data preprocessing approach*, Spatial Information Research 29.2 (2021), pp. 247–255, DOI: [10.1007/s41324-020-00349-5](#).
- [20] C. LEE: *Enhancing phishing email identification with large language models*, arXiv preprint (2025), arXiv: [2502.04759](#).
- [21] B. LIM, R. HUERTA, A. SOTELO, A. QUINTELA, P. KUMAR: *EXPLICATE: Enhancing phishing detection through explainable AI and LLM-powered interpretability*, arXiv preprint (2025), arXiv: [2503.20796](#).
- [22] M. J. NABET, L. E. GEORGE: *Phishing attacks detection by using support vector machine*, Journal of Al-Qadisiyah for Computer Science and Mathematics 15.2 (2023), pp. 180–191, DOI: [10.29304/jqcs.2023.15.21378](#).
- [23] J. RASTENIS, S. RAMANAUSKAITĖ, I. SUZDALEV, K. TUNAITYTĖ, J. JANULEVIČIUS, A. ČENYS: *Multi-language spam/phishing classification by email body text: Toward automated security incident investigation*, Electronics 10.6 (2021), p. 668, DOI: [10.3390/electronics10060668](#).
- [24] K. SHAUKAT, S. LUO, V. VARADHARAJAN, I. A. HAMEED, S. CHEN, D. LIU, J. LI: *Performance comparison and current challenges of using machine learning techniques in cybersecurity*, Energies 13.10 (2020), p. 2509, DOI: [10.3390/en13102509](#).
- [25] S. TARAPIAH, L. ABBAS, O. MARDAWI, S. ATALLA, Y. HIMEUR, W. MANSOOR: *Evaluating the effectiveness of large language models (LLMs) versus machine learning (ML) in identifying and detecting phishing email attempts*, Algorithms 18.10 (2025), p. 599, DOI: [10.3390/a18100599](#).
- [26] A. A. TAWIL, L. ALMAZAYDEH, D. QAWASMEH, B. QAWASMEH, M. ALSHINWAN, K. ELLEITHY: *Comparative analysis of machine learning algorithms for email phishing detection using TF-IDF, Word2Vec, and BERT*, Computers, Materials & Continua 81 (2024), pp. 3395–3416, DOI: [10.32604/cmc.2024.058217](#).
- [27] W. ZHU, H. LIU, Q. DONG, J. XU, S. HUANG, L. KONG, J. CHEN, L. LI: *Multilingual machine translation with large language models: Empirical results and analysis*, in: Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico: Association for Computational Linguistics, 2024, pp. 2765–2781, DOI: [10.18653/v1/2024.findings-naacl.176](#).



Practical Prediction of Query Execution Time in PostgreSQL Using Lightweight Machine Learning Models

Tamás Balla^{ab}

^aDepartment of Computational Science, Eszterházy Károly Catholic University, Hungary
balla.tamas@uni-eszterhazy.hu

^bDoctoral School of Informatics, Eötvös Loránd University, Hungary

Abstract. Accurate prediction of query execution time is important for workload scheduling, admission control, and service-level management in modern database systems. In PostgreSQL, the optimizer selects execution plans using internal cost estimates that reflect relative expected work rather than actual wall-clock runtime. This paper investigates whether these optimizer cost estimates, combined with a small set of plan-structure features extracted from machine-readable EXPLAIN output, can be calibrated into practical runtime predictions for analytical SQL queries. The study uses the TPC-H benchmark at scale factor 1 in a controlled single-user PostgreSQL environment. Features are extracted from EXPLAIN (FORMAT JSON), while runtime labels are obtained from EXPLAIN (ANALYZE, BUFFERS, FORMAT JSON). A linear baseline is compared with a Gradient Boosting Decision Tree (GBDT) regressor. The results show that optimizer cost and measured runtime are strongly correlated (Pearson's $r = 0.8625$), but a direct linear model fails to capture their relationship, yielding very poor predictive performance (MAE = 97,616.04 ms, $R^2 = -8,869.94$). In contrast, the GBDT model achieves substantially better accuracy on the random evaluation split (MAE = 74.07 ms, $R^2 = 0.9924$), indicating that the cost-to-runtime relationship is strongly nonlinear. These findings support the view that PostgreSQL's optimizer cost contains useful predictive signal, but practical runtime prediction requires nonlinear calibration. The study is intentionally modest in scope and should be interpreted as a proof of concept.

Keywords: query performance prediction, cost model, machine learning

2020 Mathematics Subject Classification: Primary 68P15, 68T05

1. Introduction

SQL is a declarative language: users specify what result they want, while the database management system decides how to produce it. In relational database systems, this decision is typically made by a cost-based optimizer, which evaluates alternative execution strategies and selects the plan with the lowest estimated cost. Since the classical System R work, cost-based optimization has remained a central principle of relational query processing, and PostgreSQL follows the same general architecture of parsing, rewriting, planning, and execution.

A fundamental difficulty, however, is that optimizer cost is not the same as elapsed execution time. In PostgreSQL, planner costs are internal relative values derived from estimated I/O and CPU work, and they are intended for comparing alternative plans rather than predicting wall-clock latency directly. By contrast, actual execution time is an observed runtime quantity that depends not only on plan structure and estimated cardinalities, but also on hardware, cache state, statistics quality, and measurement conditions. As a result, even when optimizer cost contains useful signal, it cannot be treated as a direct time estimate.

This distinction matters in practice. Accurate runtime prediction is relevant to workload scheduling, admission control, service-level management, and general query-performance diagnostics. In such scenarios, it is often not enough to know which plan PostgreSQL prefers internally; practitioners also need a realistic estimate of how long a query is likely to run. This motivates the central question of the present study: can PostgreSQL’s internal optimizer cost be calibrated into a practically useful predictor of execution time?

The problem is closely related to the broader literature on query performance prediction and learned database components. Earlier studies have shown that machine learning models can predict query latency and resource consumption from optimizer- and plan-derived features, while more recent work has explored deep plan representations, learned cost models, and learned query optimization. However, many of these approaches either require richer plan encodings, more invasive integration, or substantially larger training settings. In contrast, the present work intentionally adopts a lightweight and practical perspective.

Rather than replacing PostgreSQL’s optimizer or modifying the database engine, this paper studies runtime prediction as a cost-to-runtime calibration problem. The approach uses only standard PostgreSQL interfaces: pre-execution features are extracted from `EXPLAIN (FORMAT JSON)`, while runtime labels are obtained from `EXPLAIN (ANALYZE, BUFFERS, FORMAT JSON)`. The empirical setting is deliberately modest: the TPC-H analytical benchmark at scale factor 1, executed in a controlled single-user environment. A simple linear baseline is compared against a Gradient Boosting Decision Tree (GBDT) regressor trained on optimizer cost and a small set of interpretable plan-structure features.

The contribution of the paper is therefore practical rather than architectural. First, it frames PostgreSQL runtime prediction as a nonlinear calibration problem rather than as a learned-optimizer problem. Second, it demonstrates a compact

and reproducible workflow for extracting predictive features from PostgreSQL’s machine-readable plan output without modifying the engine. Third, it provides empirical evidence that optimizer cost contains strong predictive signal, but that effective runtime estimation requires a nonlinear model. The study is intentionally limited in scope and is best interpreted as a proof of concept; nevertheless, it shows that useful runtime prediction can be achieved with standard PostgreSQL tooling and lightweight machine learning.

2. Related Work

The present study is situated at the intersection of cost-based query optimization, query performance prediction, and learned database components.

The problem builds on classical cost-based optimization in relational database systems. Since the System R optimizer, query execution has been formulated as a search over alternative access paths and join strategies, where the optimizer selects the plan with the lowest estimated cost [15]. Modern systems, including PostgreSQL, continue to follow this principle, relying on statistics-driven cardinality estimates and analytical cost models to compare candidate plans [10, 12, 13]. At the same time, prior work has shown that optimizer quality is limited by estimation errors, so estimated cost and actual runtime may diverge substantially [4].

A closely related line of work is query performance prediction (QPP), where runtime or resource estimation is treated as a supervised learning problem. Ganapathi et al. demonstrated that machine learning can predict multiple execution-related metrics from query and plan characteristics [2]. Akdere et al. and Li et al. showed that optimizer-derived features can support accurate prediction of SQL query performance and resource consumption, even when direct reliance on analytical optimizer estimates is insufficient [1, 5]. Wu et al. further argued that optimizer cost is not unusable for runtime prediction, but rather requires a more appropriate modeling strategy than naive direct scaling [17, 18].

More recent work has extended this direction toward learned query optimization and learned cost estimation. Plan-structured neural models have been proposed for latency prediction [8], while systems such as Neo and Bao explore learned components for plan selection and optimizer guidance [6, 7]. On the cost-estimation side, learned models have also been proposed as alternatives or complements to traditional analytical cost formulas [3, 16, 20]. Industrial-scale evidence likewise confirms the practical relevance of runtime prediction for scheduling and resource-management tasks [19]. In this design space, the present work adopts a deliberately lightweight position: rather than replacing PostgreSQL’s optimizer, it investigates whether PostgreSQL’s native cost estimate and a small set of interpretable plan-structure features can be calibrated into useful runtime predictions by a nonlinear regressor.

3. Methodology

The study formulates PostgreSQL query runtime estimation as a supervised regression task. For each query instance q , a feature vector $x(q)$ is extracted from the pre-execution plan returned by `EXPLAIN (FORMAT JSON)`, while the target value $y(q)$ is the measured execution time in milliseconds obtained from the top-level `Execution Time` field of `EXPLAIN (ANALYZE, BUFFERS, FORMAT JSON)`. The learning objective is to estimate a function $f: x(q) \mapsto \hat{y}(q)$, where $\hat{y}(q)$ denotes the predicted execution time.

The feature design is intentionally lightweight and interpretable. The final feature set consists of PostgreSQL’s root-level estimated cost together with a small number of aggregated structural descriptors derived from the query plan: `total_cost`, `plan_rows`, `node_count`, `join_count`, and `scan_count`. The structural descriptors were computed by recursively traversing the JSON plan tree returned by `EXPLAIN (FORMAT JSON)`. The variable `node_count` denotes the total number of plan nodes in the tree, including the root node and all nested child plans. The variable `join_count` counts plan nodes whose type corresponds to a join operator, including Hash Join, Merge Join, and Nested Loop. The variable `scan_count` counts scan-related operators, including sequential scans, index scans, index-only scans, and bitmap scan variants. The variables `total_cost` and `plan_rows` were taken from the root-level plan node and therefore represent PostgreSQL’s estimated total plan cost and estimated output cardinality for the complete query plan, respectively. All features are available before query execution and do not rely on post-execution statistics. They were selected because they are easy to extract from machine-readable plan output and meaningful from the perspective of relational query processing. In particular, `total_cost` represents PostgreSQL’s internal estimate of expected work, while the structural counts provide additional information about plan complexity.

The target variable reflects instrumented server-side execution time rather than end-to-end application latency. This distinction matters because `EXPLAIN ANALYZE` introduces measurement overhead, and the reported execution time does not include all possible client-side or network-related delays. Nevertheless, it provides a consistent and practically meaningful label for controlled runtime prediction experiments. To avoid information leakage, only pre-execution plan attributes are used as predictors; post-execution values such as `Actual Rows` or runtime buffer statistics are not included in the feature matrix.

Two predictive models were considered. The baseline is a simple linear regression using `total_cost` as the only predictor, intended to test whether PostgreSQL’s internal cost estimate is already approximately proportional to measured runtime. The nonlinear model is a Gradient Boosting Decision Tree (GBDT) regressor trained on the full feature set. GBDT was chosen because it can capture nonlinear relationships and feature interactions while remaining well suited to small tabular datasets.

Model performance was evaluated using Mean Absolute Error (MAE), Root

Mean Squared Error (RMSE), and the coefficient of determination (R^2). In addition, Pearson correlation between `total_cost` and measured execution time was computed in order to assess whether optimizer cost contains predictive signal independently of any particular regression model. Beyond scalar metrics, the analysis also used prediction-versus-actual scatter plots, residual diagnostics, and an ablation experiment in which `total_cost` was removed from the feature set. The purpose of the ablation was to determine whether the nonlinear predictor primarily acts as a calibration layer over PostgreSQL’s internal cost estimate, or whether structural plan features alone are sufficient for accurate runtime prediction.

4. Experimental Setup

4.1. Workload

The experiments were conducted using the TPC-H decision-support benchmark at scale factor 1. TPC-H was selected because it provides a standardized analytical workload consisting of 22 business-oriented SQL query templates with substantial variation in joins, aggregations, filtering conditions, and intermediate result sizes. This makes it an appropriate benchmark for studying the relationship between optimizer-derived plan characteristics and observed execution time.

The study used the 22 standard TPC-H query templates as the workload basis. Since the goal of the paper is runtime prediction rather than benchmark certification, TPC-H was used here solely as a controlled source of analytical SQL queries. No official audited TPC-H performance metric, such as QphH@Size, is claimed.

4.2. Database System and Execution Environment

All experiments were performed using PostgreSQL 16.11 in a controlled single-user environment. The measurements were carried out on a Lenovo ThinkPad L15 laptop upgraded to 32 GB RAM, with PostgreSQL running on Ubuntu 24.04 under Windows Subsystem for Linux (WSL).

The single-user setting was chosen deliberately in order to reduce interference from concurrent sessions and to isolate the relationship between query-plan properties and runtime as clearly as possible. The reported runtimes should therefore be interpreted as measurements obtained under controlled analytical conditions rather than as estimates for highly concurrent production workloads.

4.3. Plan and Runtime Collection

For each query, two types of PostgreSQL output were collected.

First, `EXPLAIN (FORMAT JSON)` was used to obtain the pre-execution query plan in machine-readable form. This output served as the source of predictive features, including the optimizer’s estimated total cost and aggregated structural descriptors of the plan.

Second, **EXPLAIN ANALYZE** with **BUFFERS** and **JSON** output was used to collect measured execution statistics after query execution. From this output, the top-level **Execution Time** field was extracted as the regression target.

This measurement protocol creates a clean temporal separation between features and labels: the predictive model is trained only on information available before execution, while actual runtime values are obtained from separate analyzed executions.

4.4. Measurement Protocol

The workflow followed a per-query measurement strategy. For each TPC-H query template, a **JSON** plan representation was first collected for feature extraction. Query execution was then measured using one warm-up run and three retained measured runs per query in order to reduce the influence of first-run effects such as cold-cache behavior.

The SQL query instances were generated automatically by the benchmark tooling. The final regression dataset therefore contained 66 measured query executions, corresponding to 22 query templates with three retained measured runs each. Each row of the final dataset represents one retained measured execution paired with its pre-execution plan-derived features.

4.5. Train-Test Split and Feature Set

The predictive models were evaluated using a random train-test split generated with Scikit-learn's `train_test_split` function. The split used `test_size=0.3` and `random_state=42`. Accordingly, 70% of the data were used for training and 30% for testing, with a fixed random seed to ensure reproducibility.

The final feature set used in the experiments consisted of the following variables: `total_cost`, `plan_rows`, `node_count`, `join_count` and `scan_count`.

These features were extracted from PostgreSQL's machine-readable plan output and were all available before query execution.

4.6. Experimental Scope

The experimental design intentionally favors methodological clarity over scale. The benchmark size, workload scope, and controlled execution setting make the study suitable as a proof-of-concept evaluation of cost-to-runtime calibration, but they also limit the strength of generalization claims.

In particular, the study does not attempt to model:

- concurrent multi-user workloads,
- end-to-end client-observed latency,
- workload drift across different database states, or

- optimizer behavior under substantially different hardware or PostgreSQL configurations.

Instead, the focus is on a narrower and more defensible question: whether PostgreSQL’s optimizer cost and a small set of plan-structure features can be transformed into useful runtime predictions in a controlled analytical environment.

5. Results

The empirical results support four main observations. First, PostgreSQL’s internal optimizer cost contains substantial predictive signal with respect to measured execution time. The Pearson correlation coefficient between `total_cost` and observed runtime was $r = 0.8625$, indicating a strong positive relationship between optimizer cost and query latency. However, this association alone is not sufficient to justify a direct linear calibration. As shown in Figure 1, optimizer cost and runtime are related, but the relationship is not captured adequately by a simple linear model.

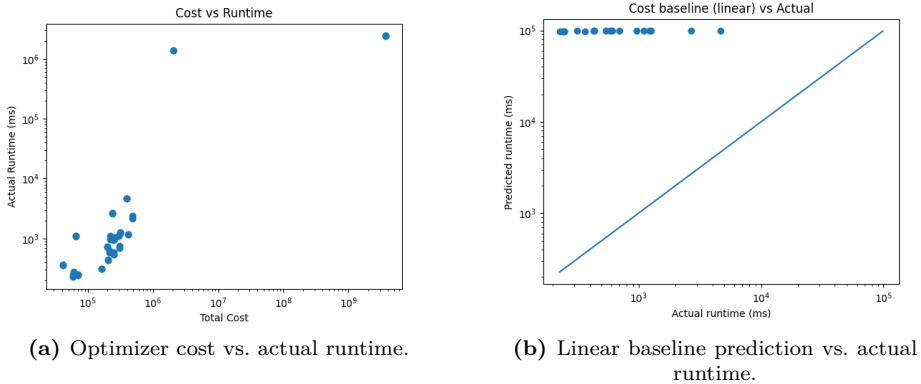


Figure 1. Relationship between PostgreSQL optimizer cost and measured execution time. Although optimizer cost is strongly associated with runtime, a direct linear baseline fails to provide an adequate calibration.

Second, the linear baseline fails to model the relationship between optimizer cost and execution time in a useful way. Using `total_cost` as the only predictor, the linear model achieved a mean absolute error of 97,616.04 ms and a coefficient of determination of $R^2 = -8869.94$. The strongly negative R^2 indicates that the linear model performs substantially worse than a trivial mean predictor. This behavior is also visible in Figure 1(b), where the predictions collapse into a narrow range and deviate strongly from the identity line. In other words, optimizer cost is informative, but it is not linearly calibrated to observed execution time.

Third, the nonlinear Gradient Boosting Decision Tree (GBDT) model substantially improves predictive performance. The GBDT regressor achieved a mean

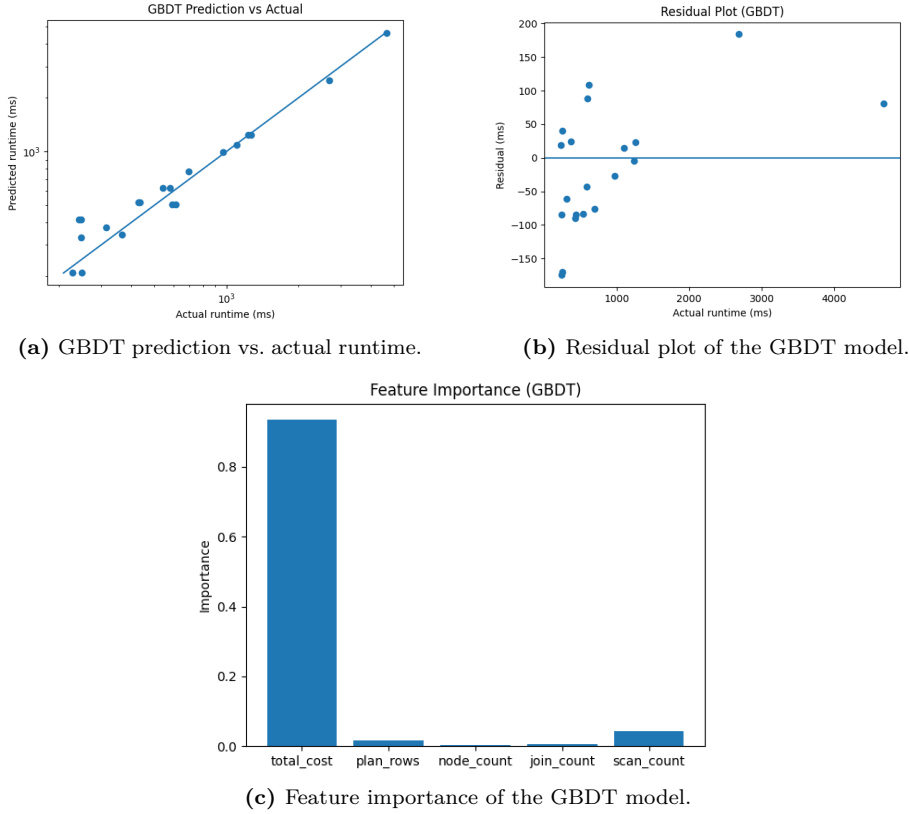


Figure 2. Evaluation of the nonlinear GBDT model. The model closely follows the identity line, residuals remain centered around zero, and feature importance indicates that `total_cost` is the dominant predictive signal.

absolute error of 74.07 ms and $R^2 = 0.9924$, indicating a very close fit between predicted and measured runtime on the evaluation subset. Although this result indicates strong predictive performance on the random test split, it should be interpreted cautiously. The dataset contains repeated measured executions of the same 22 TPC-H query templates, and the random split may place structurally identical or highly similar plans in both the training and test subsets. Consequently, the reported R^2 value should be understood primarily as evidence of interpolation performance under the chosen controlled workload and measurement protocol, rather than as evidence of generalization to unseen query templates or production workloads. More demanding evaluation protocols, such as grouped splits by query template or parameter instantiation, are left for future work. As shown in Figure 2(a), most predictions lie close to the identity line across the observed runtime range. This result strongly suggests that the relationship between PostgreSQL optimizer

cost and actual execution time is highly nonlinear, and that this nonlinearity can be captured effectively by a lightweight tree-based regressor.

Fourth, residual analysis supports the stability of the nonlinear model. Figure 2(b) shows residuals centered around zero with no clear systematic trend across the runtime range. This indicates that the GBDT model does not consistently overestimate or underestimate execution time. Although a small number of larger residuals remain for slower queries, the overall residual distribution does not indicate a strong structural bias.

Finally, the feature-importance analysis highlights the dominant role of PostgreSQL’s internal cost estimate. As shown in Figure 2(c), `total_cost` contributes most of the predictive power, while the remaining structural plan features provide smaller but non-negligible refinements. This interpretation is further supported by the ablation analysis. When `total_cost` was removed from the feature set, predictive performance deteriorated substantially, with MAE increasing to 1,339.60 ms and R^2 decreasing to -1.8071 . This result indicates that the structural plan descriptors alone were insufficient for accurate runtime prediction in the present setting. Instead, the learned model is best understood primarily as a nonlinear calibration layer over PostgreSQL’s internal optimizer signal, while the additional structural features provide complementary refinements.

Taken together, these findings support a clear interpretation. PostgreSQL’s optimizer cost is not a direct runtime estimate, and a linear cost-to-time mapping is ineffective. At the same time, the cost signal is highly informative, and a lightweight nonlinear model can transform it into a practically useful predictor of query execution time in a controlled analytical environment.

Table 1. Predictive performance of the evaluated models.

Model	MAE (ms)	R^2
Linear baseline (<code>total_cost</code> only)	97,616.04	$-8,869.94$
GBDT (full feature set)	74.07	0.9924
GBDT without <code>total_cost</code>	1,339.60	-1.8071

6. Discussion

The results suggest that PostgreSQL query runtime prediction can be interpreted as a calibration problem rather than as a replacement problem. PostgreSQL’s optimizer already encodes substantial information about expected query execution effort in its internal cost estimate, but that estimate is not intended to represent wall-clock latency directly [11, 14]. The empirical findings of the present study show that the optimizer’s cost signal is informative, but that its relationship to actual runtime is strongly nonlinear. This interpretation is consistent with earlier work showing that optimizer-derived signals can support runtime prediction, while direct use of analytical cost estimates is often insufficient [1, 18].

This interpretation is supported by the contrast between the linear baseline and the GBDT model. The strong Pearson correlation indicates that optimizer cost and runtime are meaningfully related, yet the linear baseline performs extremely poorly. This means that the problem is not the absence of signal, but the inadequacy of a simple linear mapping. In other words, PostgreSQL’s optimizer cost should not be interpreted as a direct runtime estimate, but it can serve as a useful latent signal that requires calibration.

From this perspective, the GBDT model does not replace the optimizer. Instead, it acts as a lightweight nonlinear correction layer built on top of PostgreSQL’s existing cost model. This distinction is important. The goal of the study is not to redesign plan search, cardinality estimation, or operator costing inside the database engine. Rather, it is to demonstrate that the information already exposed by PostgreSQL through standard `EXPLAIN` interfaces can be reused to obtain practically useful runtime predictions with minimal engineering overhead. In this sense, the present work is closer to query performance prediction than to full learned-optimizer systems such as Neo or Bao [6, 7].

The feature-importance analysis reinforces this interpretation. The dominant role of `total_cost` suggests that the optimizer’s own estimate remains the primary predictive signal, while the additional structural plan features provide complementary information that helps the model correct systematic distortions. This is also consistent with more recent arguments in the learned cost model literature that useful information already exists inside traditional optimization pipelines and should often be refined rather than discarded [3, 20]. The ablation result supports the same conclusion: once `total_cost` is removed, predictive quality degrades substantially. The structural descriptors alone are therefore insufficient, but they improve performance when used in combination with PostgreSQL’s internal estimate.

These findings position the proposed approach closer to the query performance prediction literature than to full learned-optimizer systems. The method is deliberately lightweight, engine-external, and based on interpretable tabular features rather than full plan-tree neural encodings [1, 2, 8]. This makes it attractive for applied settings where integration effort and reproducibility matter as much as predictive accuracy. In particular, the workflow relies only on machine-readable `EXPLAIN` output and does not require changes to PostgreSQL itself.

At the same time, the results should not be overgeneralized. The present study uses a small analytical workload in a controlled single-user environment, and the resulting model should be understood as workload- and environment-specific. The reported predictive success therefore demonstrates feasibility rather than universality. A model trained under one PostgreSQL version, hardware setting, and benchmark configuration should not automatically be assumed to transfer unchanged to other environments.

Nevertheless, the study has a clear practical implication. It shows that PostgreSQL’s optimizer cost, although not a runtime estimate in itself, can be transformed into an accurate predictor of execution time in the evaluated controlled setting when combined with a lightweight nonlinear regression model. This opens

a realistic path toward external runtime-estimation tools that can support scheduling, admission control, and performance analysis without requiring invasive modifications to the database engine. Similar application motivations have also been emphasized in both academic and industrial runtime prediction work [17, 19].

7. Limitations

The present study has several limitations that should be considered when interpreting the results.

First, the dataset is relatively small. Although the experiments cover all 22 standard TPC-H query templates at scale factor 1, the final regression dataset contains only 66 retained measured executions. This limits the statistical diversity of the workload and constrains the strength of generalization claims. In particular, repeated executions of the same benchmark templates provide less structural variety than a larger corpus of heterogeneous analytical queries. The study should therefore be interpreted as a proof-of-concept evaluation rather than as evidence of a universally robust runtime estimator.

Second, the experimental workload is restricted to a controlled single-user environment. This choice was deliberate, since it reduces interference and makes the relation between plan-derived features and observed runtime easier to analyze. However, it also means that the reported model does not capture effects caused by concurrency, workload interference, queueing delays, or resource contention. These factors are highly relevant in operational database settings and have been emphasized in prior work on runtime prediction for dynamic and concurrent workloads [17, 19]. As a result, the predictive performance reported here should not be assumed to transfer directly to multi-user production environments.

Third, the target variable is instrumented server-side execution time reported by `EXPLAIN ANALYZE`, not end-to-end application latency. PostgreSQL documentation explicitly notes that `EXPLAIN ANALYZE` adds profiling overhead to query execution and that the reported execution time does not include all client-side and network-related costs [9, 14]. Consequently, the learned model estimates PostgreSQL-reported execution time under controlled measurement conditions, which is a meaningful but narrower target than user-perceived response time.

Fourth, the evaluation uses a random train-test split with a fixed seed. While this supports reproducibility, it also introduces a possible optimism bias because retained executions of the same query template may appear in both training and test partitions. Under such conditions, the model may benefit from seeing structurally similar plans during training. A more demanding evaluation protocol could group samples by query template or by parameter instantiation in order to assess generalization more strictly. This issue is particularly relevant in light of the small dataset size.

Fifth, the feature set is intentionally lightweight and restricted to a small number of aggregated plan-level descriptors. This is an advantage from the perspective of interpretability and simplicity, but it also limits representational richness. More

expressive approaches, including operator-level models and plan-structured neural representations, may capture aspects of runtime behavior that are not visible in a compact tabular feature set [1, 8]. The present study does not attempt such comparisons and therefore cannot claim superiority over richer model families.

Finally, the learned calibration should be regarded as environment-specific. PostgreSQL planner costs depend on configuration parameters, database statistics, and the execution environment, and these factors can vary across installations [11, 12]. The mapping learned in this study is therefore tied to the particular PostgreSQL version, hardware setting, and benchmark configuration used during measurement. Applying the same model in a different environment would likely require retraining or recalibration.

These limitations do not invalidate the main result of the paper, but they define its scope clearly. The contribution is not a universal latency predictor for PostgreSQL, but rather a compact demonstration that optimizer cost can be transformed into an accurate runtime predictor through lightweight nonlinear calibration under controlled analytical conditions.

8. Conclusion

This paper investigated whether PostgreSQL’s internal optimizer cost can be calibrated into a practical predictor of query execution time using a lightweight machine learning approach. The evaluation was conducted on the TPC-H benchmark at scale factor 1 in a controlled single-user PostgreSQL environment, using features extracted from pre-execution query plans.

The results support a clear but deliberately limited conclusion. PostgreSQL’s optimizer cost contains substantial predictive signal, as shown by the strong positive correlation between `total_cost` and measured runtime. However, this signal is not linearly calibrated to wall-clock execution time: the linear baseline performed extremely poorly, whereas the Gradient Boosting Decision Tree model achieved high predictive accuracy on the random test split. Because the dataset contains repeated executions of the same TPC-H query templates, this result should be interpreted as proof-of-concept evidence under controlled conditions rather than as strong evidence of generalization to unseen query templates or production workloads. Residual analysis further indicated that the nonlinear model did not exhibit clear systematic bias, and feature-importance analysis showed that `total_cost` remained the dominant predictive variable, complemented by a small set of plan-structure descriptors.

Taken together, these findings suggest that runtime prediction in PostgreSQL can be understood as a nonlinear cost-calibration problem. The proposed approach does not attempt to replace PostgreSQL’s optimizer or modify the database engine. Instead, it demonstrates that useful runtime prediction can be achieved externally, using standard PostgreSQL interfaces and an interpretable lightweight model.

The contribution of the paper is therefore practical and methodological. It shows that even a small, controlled experimental setup can provide evidence that

optimizer-derived signals are valuable for runtime prediction, provided that they are modeled appropriately. At the same time, the study remains deliberately modest in scope. The reported results should be interpreted as a proof of concept tied to a specific benchmark, PostgreSQL configuration, and execution environment rather than as a universal latency estimator.

Several directions for future work follow naturally from this study. A larger dataset with more diverse query parameterizations and additional workloads would allow stronger generalization claims. More demanding evaluation protocols, such as grouped splits by query template or parameter instantiation, would provide a stricter assessment of robustness. It would also be valuable to extend the experiments to concurrent multi-user settings, where queueing and resource contention influence runtime behavior more strongly. Finally, future work could compare the present lightweight tabular approach with richer operator-level or plan-structured models in order to better understand the trade-off between simplicity, interpretability, and predictive performance.

The study demonstrates that PostgreSQL’s internal cost signal, although not designed as a runtime estimate, can be transformed into an accurate execution-time predictor through lightweight nonlinear calibration under controlled analytical conditions.

References

- [1] M. AKDERE, U. CETINTEMEL, M. RIONDATO, E. UPFAL, S. B. ZDONIK: *Learning-based Query Performance Modeling and Prediction*, in: 2012 IEEE 28th International Conference on Data Engineering, 2012, pp. 390–401, DOI: [10.1109/ICDE.2012.64](https://doi.org/10.1109/ICDE.2012.64).
- [2] A. GANAPATHI, H. A. KUNO, U. DAYAL, J. L. WIENER, A. FOX, M. I. JORDAN, D. A. PATTERSON: *Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning*, in: Proceedings of the 2009 IEEE International Conference on Data Engineering, 2009, pp. 592–603, DOI: [10.1109/ICDE.2009.130](https://doi.org/10.1109/ICDE.2009.130).
- [3] R. HEINRICH, X. LI, M. LUTHRA, Z. KAoudi: *Learned Cost Models for Query Optimization: From Batch to Streaming Systems*, Proceedings of the VLDB Endowment 18.12 (2025), pp. 5482–5487, DOI: [10.14778/3750601.3750699](https://doi.org/10.14778/3750601.3750699).
- [4] V. LEIS, A. GUBICHEV, A. MIRCHEV, P. BONCZ, A. KEMPER, T. NEUMANN: *How Good Are Query Optimizers, Really?*, Proceedings of the VLDB Endowment 9.3 (2015), pp. 204–215, DOI: [10.14778/2850583.2850594](https://doi.org/10.14778/2850583.2850594).
- [5] J. LI, A. C. KONIG, V. NARASAYYA, S. CHAUDHURI: *Robust Estimation of Resource Consumption for SQL Queries Using Statistical Techniques*, Proceedings of the VLDB Endowment 5.11 (2012), pp. 1555–1566, DOI: [10.14778/2350229.2350269](https://doi.org/10.14778/2350229.2350269).
- [6] R. MARCUS, P. NEGI, H. MAO, N. TATBUL, M. ALIZADEH, T. KRASKA: *Bao: Making Learned Query Optimization Practical*, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 1275–1288, DOI: [10.1145/3448016.3452838](https://doi.org/10.1145/3448016.3452838).
- [7] R. MARCUS, P. NEGI, H. MAO, C. ZHANG, M. ALIZADEH, T. KRASKA, O. PAPAEMMANOUIL, N. TATBUL: *Neo: A Learned Query Optimizer*, Proceedings of the VLDB Endowment 12.11 (2019), pp. 1705–1718, DOI: [10.14778/3342263.3342644](https://doi.org/10.14778/3342263.3342644).
- [8] R. MARCUS, O. PAPAEMMANOUIL: *Plan-Structured Deep Neural Network Models for Query Performance Prediction*, Proceedings of the VLDB Endowment 12.11 (2019), pp. 1733–1746, DOI: [10.14778/3342263.3342646](https://doi.org/10.14778/3342263.3342646).

- [9] POSTGRESQL GLOBAL DEVELOPMENT GROUP: *EXPLAIN*, Accessed: 2026-04-19, PostgreSQL Global Development Group, 2026, URL: <https://www.postgresql.org/docs/current/sql-explain.html>.
- [10] POSTGRESQL GLOBAL DEVELOPMENT GROUP: *Planner/Optimizer*, Accessed: 2026-04-19, PostgreSQL Global Development Group, 2026, URL: <https://www.postgresql.org/docs/current/planner-optimizer.html>.
- [11] POSTGRESQL GLOBAL DEVELOPMENT GROUP: *Query Planning*, Accessed: 2026-04-19, PostgreSQL Global Development Group, 2026, URL: <https://www.postgresql.org/docs/current/runtime-config-query.html>.
- [12] POSTGRESQL GLOBAL DEVELOPMENT GROUP: *Statistics Used by the Planner*, Accessed: 2026-04-19, PostgreSQL Global Development Group, 2026, URL: <https://www.postgresql.org/docs/current/planner-stats.html>.
- [13] POSTGRESQL GLOBAL DEVELOPMENT GROUP: *The Path of a Query*, Accessed: 2026-04-19, PostgreSQL Global Development Group, 2026, URL: <https://www.postgresql.org/docs/current/query-path.html>.
- [14] POSTGRESQL GLOBAL DEVELOPMENT GROUP: *Using EXPLAIN*, Accessed: 2026-04-19, PostgreSQL Global Development Group, 2026, URL: <https://www.postgresql.org/docs/current/using-explain.html>.
- [15] P. G. SELINGER, M. M. ASTRAHAN, D. D. CHAMBERLIN, R. A. LORIE, T. G. PRICE: *Access Path Selection in a Relational Database Management System*, in: Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, 1979, pp. 23–34, DOI: [10.1145/582095.582099](https://doi.org/10.1145/582095.582099).
- [16] J. SUN, G. LI: *An End-to-End Learning-based Cost Estimator*, Proceedings of the VLDB Endowment 13.3 (2019), pp. 307–319, DOI: [10.14778/3368289.3368296](https://doi.org/10.14778/3368289.3368296).
- [17] W. WU, Y. CHI, H. HACIGÜMÜŞ, J. F. NAUGHTON: *Towards Predicting Query Execution Time for Concurrent and Dynamic Database Workloads*, Proceedings of the VLDB Endowment 6.10 (2013), pp. 925–936, DOI: [10.14778/2536206.2536219](https://doi.org/10.14778/2536206.2536219).
- [18] W. WU, Y. CHI, S. ZHU, J. TATEMURA, H. HACIGÜMÜŞ, J. F. NAUGHTON: *Predicting Query Execution Time: Are Optimizer Cost Models Really Unusable?*, in: 2013 IEEE 29th International Conference on Data Engineering (ICDE), 2013, pp. 1081–1092, DOI: [10.1109/ICDE.2013.6544899](https://doi.org/10.1109/ICDE.2013.6544899).
- [19] Z. WU, R. MARCUS, Z. LIU, P. NEGI, V. NATHAN, P. PFEIL, G. SAXENA, M. RAHMAN, B. NARAYANASWAMY, T. KRASKA: *Stage: Query Execution Time Prediction in Amazon Redshift*, in: Companion of the 2024 International Conference on Management of Data, 2024, pp. 280–294, DOI: [10.1145/3626246.3653391](https://doi.org/10.1145/3626246.3653391).
- [20] J. YANG, S. WU, D. ZHANG, J. DAI, F. LI, G. CHEN: *Rethinking Learned Cost Models: Why Start from Scratch?*, Proceedings of the ACM on Management of Data 1.4 (2023), pp. 1–27, DOI: [10.1145/3626769](https://doi.org/10.1145/3626769).



Accuracy of Wearable Heart Rate Monitors under Static and High-Intensity Activity Conditions

János Dávid Balogh, Attila Adamkó

Faculty of Informatics, University of Debrecen
davidkm2@mailbox.unideb.hu
adamko.attila@inf.unideb.hu

Abstract. The widespread adoption of wearable devices has created new opportunities for continuous monitoring of physical activity, cardiovascular health, and sleep patterns. However, it remains an open question to what extent the physiological and kinematic data provided by commercial smartwatches can be considered accurate and clinically interpretable under different conditions. The aim of this study is to compare heart rate (HR), heart rate variability (HRV), step count, and sleep parameters measured by four commercial smartwatches against clinical-grade reference devices under both static/resting and dynamic activity conditions.

In a controlled study involving 30 participants, we evaluated four popular device models: Apple Watch Series 8, Garmin Forerunner 265, Samsung Galaxy Watch 6, and Xiaomi Smart Band 8. For heart rate in a static environment, the devices showed high correlation with the reference ($r > 0.92$) and low Mean Absolute Error (MAE < 3 bpm). During high-intensity physical activity, the HR error rate increased significantly across all models (MAE = 8–16 bpm). Step count accuracy was generally high during steady-state walking (error $< 3\%$), but variability increased during running. Regarding sleep parameters, the devices estimated Total Sleep Time (TST) with moderate accuracy, but sleep stage determination showed poor reliability compared to clinical polygraphy.

The results suggest that smartwatches are highly suitable for detecting long-term health trends and tracking general physical activity. However, their use for critical clinical decision-support requires careful interpretation, particularly regarding dynamic heart rate tracking and sleep architecture analysis. The study highlights the need for standardized multi-parameter validation protocols.

Keywords: Wearable devices, Heart rate monitoring, Photoplethysmography (PPG), Heart rate variability (HRV), Step count, Sleep tracking, Measurement accuracy, Motion artifacts, Clinical validation

1. Introduction

Wearable devices equipped with photoplethysmography (PPG) sensors and triaxial accelerometers have revolutionized the continuous monitoring of human physiology and daily activity. These consumer devices are increasingly utilized as preliminary screening tools for detecting cardiovascular anomalies, tracking physical exertion, and monitoring sleep quality [5, 13].

Despite their popularity, the clinical reliability of wrist-worn sensors remains controversial. Previous literature highlights that while PPG technology performs admirably under static or resting conditions, its accuracy degrades significantly during physical exertion due to motion artifacts, changes in skin contact pressure, and variations in peripheral blood perfusion [3, 6, 8]. Similarly, while accelerometer-based step counting is well-established, its accuracy can fluctuate depending on walking speed and arm kinematics. Furthermore, sleep tracking relies on a combination of actigraphy and HR/HRV algorithms, which often struggle to accurately differentiate between specific sleep stages when compared to clinical references [7].

While numerous studies have validated specific devices in isolated scenarios, there is a lack of comprehensive, multi-parameter evaluations focusing on HR, steps, and sleep across multiple devices in a unified methodological framework [12]. Therefore, the primary objective of this empirical study is to present a structured validation framework that quantitatively evaluates the reliability of four highly popular commercial smartwatches under strictly controlled conditions against standard clinical reference instruments.

2. Methods

2.1. Participants

A total of 30 healthy adult volunteers (15 male, 15 female; mean age 26 ± 4 years, mean BMI 23.5 ± 2.1) participated in the study. All participants provided informed consent prior to the measurements. The experimental protocol was approved by the institutional ethics committee.

2.2. Devices and Reference Standards

To ensure a comprehensive analysis, four commercially available smartwatches from leading manufacturers were evaluated simultaneously: Apple Watch Series 8, Garmin Forerunner 265, Samsung Galaxy Watch 6, and Xiaomi Smart Band 8.

To validate the smartwatch measurements, we utilized the following standard clinical reference devices:

- **Heart Rate & HRV:** A clinically validated ambulatory Holter ECG monitor, widely used in hospital settings for reliable beat-to-beat cardiovascular monitoring [11].
- **Step Count:** A clinically validated pedometer, supplemented by manual video-based step tallying to ensure ground-truth accuracy during the treadmill protocol.
- **Sleep Parameters:** A clinically validated polygraphy device (cardiorespiratory monitor), commonly used in clinical settings for sleep assessment, recording airflow, respiratory effort, and continuous oxygen saturation.

2.3. Experimental Protocol

The data collection was divided into three distinct phases:

1. **Static/Resting HR and HRV Phase:** Participants remained seated in a resting state for 15 minutes to establish a baseline with low temporal dynamics.
2. **Dynamic Activity and Step Count Phase:** Participants performed a treadmill protocol consisting of a 10-minute walk (5 km/h) followed by a 10-minute high-intensity run (10–12 km/h). This phase evaluated both the step-counting algorithms and the robustness of the HR sensors against motion artifacts.
3. **Sleep Monitoring Phase:** A subset of participants ($n = 15$) wore the smartwatches and the portable polygraphy system overnight in a controlled environment for a minimum of 7 hours.

2.4. Statistical Analysis

Data were synchronized using timestamps. Performance was evaluated using Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), and Pearson’s correlation coefficient (r). Statistical significance between measurement errors under different conditions was determined using paired t-tests, with $p < 0.05$ considered significant. Bland–Altman analysis was applied to assess the limits of agreement (LoA) [4].

3. Results

3.1. Heart Rate and HRV Accuracy

The evaluation yielded a clear dichotomy in HR measurement performance based on the physical activity level. Table 1 summarizes the quantitative metrics for each device compared to the ECG reference.

Table 1. Device-level HR Accuracy: Static vs. Dynamic Conditions.

Device	Static MAE	Static r	Dynamic MAE	Dynamic r
Apple Watch Series 8	1.5 bpm	0.98	8.5 bpm	0.85
Garmin Forerunner 265	1.8 bpm	0.96	9.2 bpm	0.82
Samsung Galaxy Watch 6	2.2 bpm	0.95	14.5 bpm	0.72
Xiaomi Smart Band 8	2.5 bpm	0.92	16.1 bpm	0.68
Average	2.0 bpm	0.95	12.1 bpm	0.77

In the static environment, all smartwatches demonstrated a very high correlation with the reference and an exceptionally low MAE. Conversely, high-intensity running resulted in a statistically significant increase in measurement error across all models ($p < 0.001$). The Apple Watch and Garmin managed motion artifacts slightly better than the Samsung and Xiaomi devices, but the degradation in accuracy was universal. Regarding HRV, dynamic activity profoundly distorted the pulse-to-pulse intervals (PPIs), making HRV data derived during intense exercise statistically unreliable compared to resting HRV [2, 10].

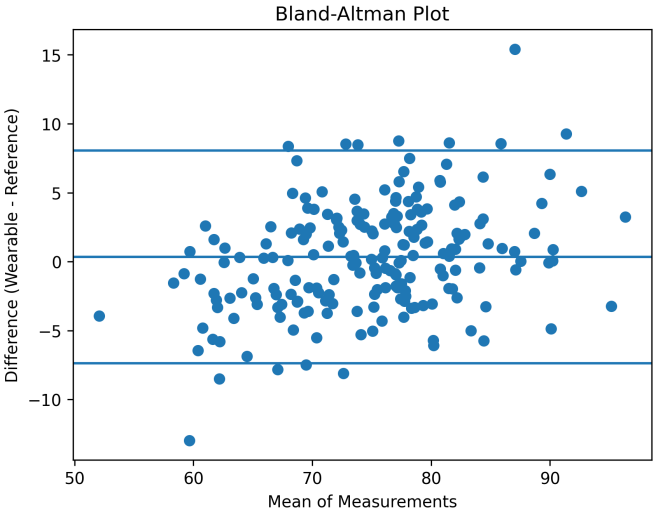


Figure 1. Bland–Altman plot for HR measurements. Wider limits of agreement during dynamic activity (dashed lines) illustrate the increased error margin and vulnerability to motion artifacts.

3.2. Step Count Accuracy

Step counting proved to be the most reliable metric across the evaluated devices. Table 2 shows the Mean Absolute Percentage Error (MAPE) against the clinically

validated pedometer and video reference.

Table 2. Step Count Accuracy (MAPE %).

Device	Walking (5 km/h)	Running (12 km/h)
Apple Watch Series 8	1.2%	2.5%
Garmin Forerunner 265	1.5%	2.8%
Samsung Galaxy Watch 6	2.1%	4.2%
Xiaomi Smart Band 8	2.5%	4.8%

During steady-state walking, the error was negligible (MAPE < 3% for all devices). During running, the error increased slightly due to changes in arm swing kinematics, but overall step detection remained highly robust and suitable for daily physical activity tracking.

3.3. Sleep Parameter Accuracy

Sleep tracking accuracy varied significantly depending on the metric evaluated. Table 3 compares the wearable estimates of Total Sleep Time (TST) and Sleep Efficiency (SE) against the clinical polygraphy data [7].

Table 3. Sleep Tracking Accuracy vs. Clinical Polygraphy.

Device	TST Error (Bias ± SD)	SE Error (Bias ± SD)
Apple Watch Series 8	+15 ± 20 min	+2.5 ± 4.0%
Garmin Forerunner 265	+12 ± 18 min	+1.8 ± 3.5%
Samsung Galaxy Watch 6	+22 ± 25 min	+3.2 ± 5.1%
Xiaomi Smart Band 8	+30 ± 28 min	+4.5 ± 6.2%

All devices systematically overestimated Total Sleep Time by 12 to 30 minutes, primarily due to misclassifying periods of quiet wakefulness as light sleep. Furthermore, while basic sleep duration was moderately accurate, sleep stage determination showed poor agreement with the clinical reference, with Cohen’s kappa values ranging from 0.35 to 0.48 across the devices.

4. Discussion

The results of this study confirm that commercial smartwatches offer a mixed profile of reliability depending on the measured parameter and the physical state of the user. Step counting remains highly accurate regardless of intensity, validating these devices as excellent tools for general activity promotion.

For cardiovascular metrics, our findings align with previous literature [6, 9]: in static environments, PPG sensors closely match clinical ECG references, supporting

their use for resting HR and baseline HRV monitoring. However, the MAE increase to 8–16 bpm during high-intensity running indicates that these devices should not be solely relied upon for real-time clinical decision-support or precise sports-health diagnostics during exertion.

Sleep tracking results present a similar duality. While smartwatches provide a reasonable estimation of overall sleep duration (TST), their reliance on wrist actigraphy and basic HR patterns limits their ability to accurately map complex sleep architecture [7]. Therefore, clinical sleep diagnostics still firmly require clinical-grade polygraphy or polysomnography.

Future improvements in the wearable industry must target robust motion-compensation algorithms for PPG and multi-sensor fusion for sleep staging. Furthermore, the validation of physiological data streams should be integrated into modular system architectures that can continuously assess data quality and flag anomalies caused by sensor noise before the data is fed into predictive health informatics models [1].

5. Conclusion

Wearable devices demonstrate high accuracy in tracking step counts and resting heart rates, making them valuable tools for general wellness tracking and long-term physiological trend detection.

However, during high-intensity physical activity, HR measurement error increases significantly due to motion artifacts. Similarly, while overnight sleep duration is estimated with acceptable accuracy, advanced sleep staging remains unreliable compared to clinical standards.

Future research should focus on:

- Developing advanced adaptive filtering to reduce PPG noise during vigorous activity.
- Enhancing sleep algorithms by fusing continuous HRV, temperature, and actigraphy data.
- Standardizing ambulatory validation protocols across diverse populations and physical activities.
- Enhancing algorithmic transparency, allowing independent scientific verification of proprietary signal processing methodologies.

In conclusion, while commercial smartwatches are excellent for non-critical health monitoring, integrating their dynamic HR and sleep measurements into high-stakes clinical workflows necessitates a cautious approach, robust validation, and awareness of their technological limitations.

References

- [1] J. D. BALOGH, A. ADAMKÓ: *eHealth and Smart Solutions Framework for Health Monitoring in the Course of the Pandemic*, Annales Mathematicae et Informaticae (2023), DOI: [10.33039/ami.2023.08.013](#).
- [2] C. BELLENGER, D. MILLER, S. HALSON, G. ROACH, C. SARGENT: *Wrist-Based Photoplethysmography Assessment of Heart Rate and Heart Rate Variability*, Sensors 21.10 (2021), p. 3571, DOI: [10.3390/s21103571](#).
- [3] B. BENT, B. GOLDSTEIN, W. KIBBE, J. DUNN: *Investigating Sources of Inaccuracy in Wearable Optical Heart Rate Sensors*, npj Digital Medicine 3 (2020), p. 18, DOI: [10.1038/s41746-020-0226-6](#).
- [4] J. M. BLAND, D. G. ALTMAN: *Statistical methods for assessing agreement between two methods of clinical measurement*, The Lancet 327.8476 (1986), pp. 307–310, DOI: [10.1016/S0140-6736\(86\)90837-8](#).
- [5] M. ETIWY, Z. AKRAS, S. GILLINOV, D. PHELAN, R. WANG, G. BLACKBURN, P. HOUGHTALING, H. JAVADIKASGARI, M. DESAI: *Erratum to accuracy of wearable heart rate monitors in cardiac rehabilitation*, Cardiovascular Diagnosis and Therapy 10.3 (2020), pp. 644–645, DOI: [10.21037/cdt.2019.10.05](#).
- [6] S. GILLINOV, M. ETIWY, ET AL.: *Variable accuracy of wearable heart rate monitors during aerobic exercise*, Medicine & Science in Sports & Exercise 49.8 (2017), pp. 1697–1703, DOI: [10.1249/MSS.0000000000001284](#).
- [7] S. HAGHAYEGH, S. KHOSHNEVIS, M. H. SMOLENSKY, K. R. DILLER: *Accuracy of PurePulse photoplethysmography technology of Fitbit Charge 2 for assessment of heart rate during sleep*, Chronobiology International 36.7 (2019), pp. 927–932, DOI: [10.1080/07420528.2019.1596947](#).
- [8] J. PARAK, I. KORHONEN: *Evaluation of wearable consumer heart rate monitors based on photoplethysmography*, in: 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, Chicago, IL, USA, 2014, pp. 3670–3673, DOI: [10.1109/EMBC.2014.6944419](#).
- [9] R. QUINN, N. LEADER, G. LEBOVIC, C.-M. CHOW, P. DORIAN: *Accuracy of Wearable Heart Rate Monitors During Exercise in Sinus Rhythm and Atrial Fibrillation*, Journal of the American College of Cardiology 83.12 (2024), pp. 1177–1179, DOI: [10.1016/j.jacc.2024.01.024](#).
- [10] F. SHAFFER, J. P. GINSBERG: *An Overview of Heart Rate Variability Metrics and Norms*, Frontiers in Public Health 5 (2017), p. 258, DOI: [10.3389/fpubh.2017.00258](#).
- [11] R. Z. UR REHMAN, M. CHATTERJEE, N. V. MANYAKOV, ET AL.: *Assessment of Physiological Signals from Photoplethysmography Sensors Compared to an Electrocardiogram Sensor: A Validation Study in Daily Life*, Sensors 24.21 (2024), p. 6826, DOI: [10.3390/s24216826](#).
- [12] C. N. VAN OOST, F. MASCI, A. MALISSE, ET AL.: *Accuracy of Heart Rate Measurement Under Transient States: A Validation Study of Wearables for Real-Life Monitoring*, Sensors 25.20 (2025), p. 6319, DOI: [10.3390/s25206319](#).
- [13] M. P. WALLEN, S. R. GOMERSALL, S. E. KEATING, U. WISLØFF, J. S. COOMBES: *Accuracy of heart rate watches: Implications for weight management*, PLoS ONE 11.5 (2016), e0154420, DOI: [10.1371/journal.pone.0154420](#).



Creating Images on Non-Traditional Grids for Digital Image Processing

Nándor Banyik^{a,b}, Benedek Nagy^{a,c}

^aEszterházy Károly Catholic University, Institute of Mathematics and Informatics
banyik.nandor@uni-eszterhazy.hu

^bUniversity of Debrecen, Doctoral School of Informatics

^cEastern Mediterranean University, Faculty of Arts and Sciences
Department of Mathematics
nbenedek.inf@gmail.com

Abstract. Digital image processing and computer graphics are based on the underlying grid (e.g., defined by the computer screen). However, the square grid has some disadvantages, e.g., topological paradox. Therefore, some non-traditional grids could effectively be used in these fields. On the other hand, there are wide-spread imaging devices (e.g., smart phones) that provide very high resolution images on the square grid. In this work, we propose resampling techniques to allow to save images on non-traditional grids. This gives the possibility that one can choose a grid for processing the image that has some better symmetrical and/or topological properties than the traditional grid has.

Keywords: digital image processing, computer graphics, geometric modeling, nontraditional grids, image resampling, symmetry

2020 Mathematics Subject Classification: Primary 68U10, 68U05, 52C20; Secondary 05B45

1. Introduction

In this paper we will address one of the most important preparation processes before using non-traditional grids in, e.g., imaging sciences, namely, how to create and have images with various shape of pixels, i.e., on various grids.

The usual question could be stated: why not the square grid? The square

grids has numerous issues such as the edge handling. While vertical and horizontal lines are continuous, completely diagonal edges experience information gaps due to pixel connections. This phenomenon results from the inhomogeneous neighborhood relations of the square grid, where the distance of immediate neighbors alternates between $d = 1$ (side neighbor) and $d = \sqrt{2}$ (diagonal neighbor). This inconsistency extends to the violation of the digital version of the Jordan Curve Theorem, creating a topological paradox for which one of the best known solutions is that a curve can only be considered ‘closed’ if the neighborhood definitions for the object and its background are inconsistently applied (e.g., 8-connectivity for the curve and 4-connectivity for the background, or *vice versa*). To address this and related problems without using special methods, it is suitable to use non-traditional grids. Some more details and arguments can be found, e.g. in [4, 19, 27]. Non-traditional grids are useful and can be used also for path planning, building maps with robots and pictorial representation of sensor data [1, 7].

In this paper, as an initial step, we address the problem how to have images on various grids.

2. Historical Notes

There were some resampling techniques proposed in the literature earlier, e.g., the authors of [8] recommended resampling the image by uniting two-two square pixels to make a triangular pixel (trixel). This technique can be seen in Figure 1, on the left.

In a very similar manner, based on a rectangular grid in a brick-wall like arrangement hexagonal resampling can be done, see, e.g., in [24] and in [11]. Here again, the resampling is done by uniting two neighbor pixels to one new pixel as it is shown in Figure 1, on the right.

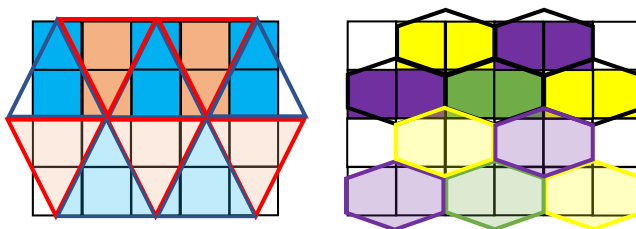


Figure 1. An idea for resampling images on the square grid for the triangular grid (left) and for the hexagonal grid (right) by uniting pixels.

One of the major problems of these low resolution resampling techniques that the resulting image is biased by the original image-grid. For example, the neighbor-

hood system of the triangle pixels will inherit the square-grid based neighborhood and do not reflect the symmetry and the structure of the triangular grid. There are similar methods used, e.g., to make, test and reconstruct binary phantom images on the square grid and on the Cairo pattern, one of the best known pentagonal tilings [13].

In this paper, we consider the problem in a more general way. Our aim is to have a program tool that has a high-quality traditional uncompressed digital image (with standard square pixels) as input together with the targeted grid and the resolution that the resulting image should have, and the program is able to generate the resampled image on the defined grid with the specified resolution as output. The next sections show the details.

3. Proposed Grids

Our aim is to make a general program that has also as the part of the input the target grid and the resolution. Although our aim is to have the system general, in this section, we recall three different grids with various properties. These grids can be seen as motivations, on the one hand, for our ‘preparation’ task. On the other hand, with them we can also highlight what type of properties a grid may have and what type of problems one needs to deal with when preparing a software tool for resampling the images.

3.1. Hexagonal grid

The hexagonal grid is one of the best known and probably the second most used grid. It is among the three regular grids. The image is built up by regular hexagons (hexels). This image grid is a point lattice, i.e., it is closed under translations of any grid vector (i.e., vector connecting the center of any two hexels). The hexagonal pixels are more round than the square pixels, and the six neighbors are more isometrically oriented than the four side neighbors in the square grid. Because of the only one type of neighbor relation, the grid is relatively simple. Her introduced a symmetric coordinate system where 0-sum integer triplets are used to address the hexels [9]. There are no topological paradoxes in this grid. The grid gives the most efficient packing (of same size disks) in the plane, and also used in various places in the nature, e.g., honeybees build their honeycombs in this structure.

As this is the second best used planar grid, several image processing algorithms are already developed for images represented in this grid. Geometric transformations are investigated in [6, 10, 23]. Distance transform for hexagonal images is computed in [3]. Tomography and image reconstruction for images on this grid are considered, e.g., in [14, 15, 20] with various approaches. Cellular topology is discussed in [18, 26], just to mention a few results already available on this grid. There is also a textbook on hexagonal image processing [16]. Some advantages of this grid over the classical square grid are also highlighted in [4, 27]. However, maybe because our screens, projectors and other wide-spread hardware devices use

mostly only the square grid, there are no well-known image storing formats and image databases (up to the knowledge of the authors) for such images. Since this grid is still very popular, we included it in our study.

3.2. Cairo pattern

The Cairo pattern is also well-known, it is named after paving some streets in Cairo. In fact, this is a pentagonal grid where the pentagons are identical, but not regular (see Figure 2). It is the dual tiling of the semi-regular tiling denoted by $(3, 3, 4, 3, 4)$ with the vertex configuration showing that at each vertex three triangles and two squares meet [5]. This grid was proposed to be used in image processing applications by G. Borgefors [2]. In the Cairo pattern there is a sole tile, but it appears in four different orientations. The Cairo pattern is involved in digital geometry and image processing: coordinate systems and various digital distance functions are defined and computed in [12, 21, 25]; binary tomography algorithm was presented in [13]. The symmetry of the grid includes both hexagonal type and rectangular type symmetries, as the two mentioned coordinate systems reflect these facts and this is one of the reasons why this grid could have various applications connected to imaging. This grid is an example for a grid when a pixel may have various orientations. This property is typical for the Catalan grids, i.e., for the dual grids of semi-regular grids [19, 22].

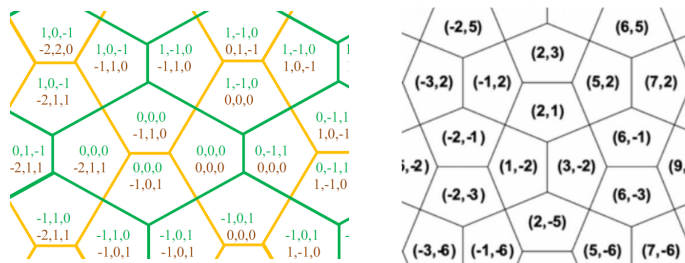


Figure 2. Cairo pattern with two coordinate systems from [12, 21], respectively.

3.3. Truncated nonagonal grid

The third grid we highlight here, contains non-regular enneagons (nonagons) and triangles. This grid was introduced in [17] as a possible solution to avoid topological paradoxes on a triangular grid. One may think about this grid that a type (one of the two orientations) of the triangles are pumped up in such a way that they can touch by edges the six closes same type pixels. Consequently the other types of triangles (the triangles with the opposite orientation) are squeezed not to have any extra corner neighbors, only the three side neighbor enneagons. Figure 3 shows a

part of the grid. This grid is a typical example for grids built up by more than one type of pixels. All semi-regular grids also have this property [19].

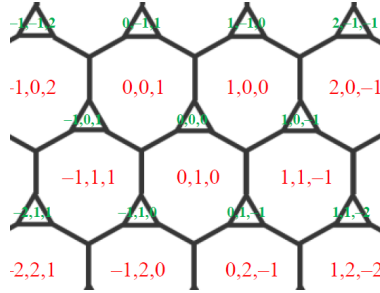


Figure 3. Truncated nonagonal grid with a coordinate system using integer triplets.

4. Resampling Process

Traditional resampling methods typically rely on a uniform, single-cell unit, such as a square or a simple hexagon. In contrast, we propose a methodology based on symmetric kernels composed of subdivided cells. This approach ensures that individual grid elements are not restricted to basic geometric primitives; instead, it allows for the construction of complex, heterogeneous grids, such as the truncated-nonagonal grid. By utilizing subdivided cells, we can maintain structural integrity even when multiple shape types are integrated into a single tiling system. Thus, for our method, the input is not only a high-quality uncompressed image on the traditional grid, but also a kernel describing the grid. This kernel must have a rectangular shape such that if one tiles the plane with its shifted copies in side-to-side manner, the kernel defines a periodic tiling of the plane. In Figure 4 we show the example kernels for our chosen grids. It is easy to observe how the hexagonal, the variously oriented pentagonal and the enneagon and triangle cells (pixels) tile the plane, respectively, based on the kernel.

The primary advantage of this kernel-based approach is the simplified workflow for establishing adjacency between current and preceding neighboring kernels. By defining the grid structure within a vertex matrix rather than as a singular repeating unit, we effectively eliminate spatial inconsistencies and the issue of unit shifting. This alignment is critical when mapping non-traditional grids onto the original image matrix, as it allows for the precise subdivision of kernel sizes while maintaining a perfect fit over the source data.

During the dynamic resampling process, grid units are synthesized from the divided components of both the current and the previous kernels, where the whole image contains $n \times m$ kernels and the current kernel is k_c , and the previous kernels are $k_c - 1, k_c - n$. Once the specific cell vertices are established, their exact

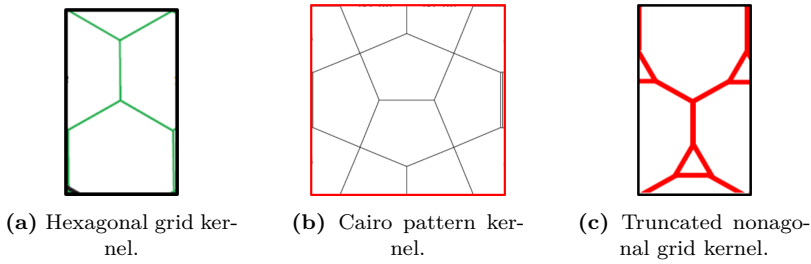


Figure 4. The kernels for the proposed grids.

coordinates are mapped onto the image matrix. The algorithm then identifies all pixels enclosed within the boundaries of a given cell and performs a color averaging operation. The resulting mean color value is assigned to the corresponding grid cell:

$$C_{cell} = \frac{1}{n} \sum_{i=1}^n P_i,$$

where P_i is a pixel inside a grid cell. While this averaging process inherently results in a loss of high-frequency spatial and edge information, it is not a significant drawback in the context of our research as it is assumed that the input is an extremely high resolution image (i.e., it may contain a hundred megapixel, as such devices that may produce such digital images become more and more common in our everyday life). As the usual image processing tasks are working with lower-resolution images, our aim to output lower-resolution images on the given non-traditional grids. Thus, the framework is optimized for lower-resolution image processing where the primary goal is to maintain topological stability and prevent the geometric distortions typical of traditional square lattices.

5. Preparation for Custom Grids

The manual creation of unique kernels for non-traditional grids is a mathematically intensive and time-consuming task. To address this, we introduce an automated generation system supported by a dedicated graphical user interface (GUI). In this environment, users can define the fundamental vertex points of a kernel interactively. While these raw points do not constitute a complete grid definition on their own, their spatial coordinates serve as the foundational data for the automated construction of the vertex matrix.

Once the vertices are defined, the user partitions the regions for each cell or subdivided cell. This step is essential for establishing the neighborhood relations and connectivity rules that define the grid's topology. Automating this process ensures that the resulting grid adheres to the necessary symmetry and distance relations required for effective data handling.

A key technical challenge lies in the development of a robust, automatic coordinate system for these custom structures. However, by enforcing a convexity constraint on the generated grid cells, we can streamline the rendering process. Convex cells allow for the implementation of automated triangle slicing (tessellation) algorithms, which are natively supported and highly optimized for the OpenGL API. This ensures that even high-resolution images can be resampled and rendered onto custom, non-traditional grids.

6. Some Experimental Results

In this section, to show how our method works in the practice, we give some examples. One may notice that our method can also be used to create low(er) resolution images on the square grid, and thus, we may have a real comparison of the images on various grids. Later on these comparisons will be very important to

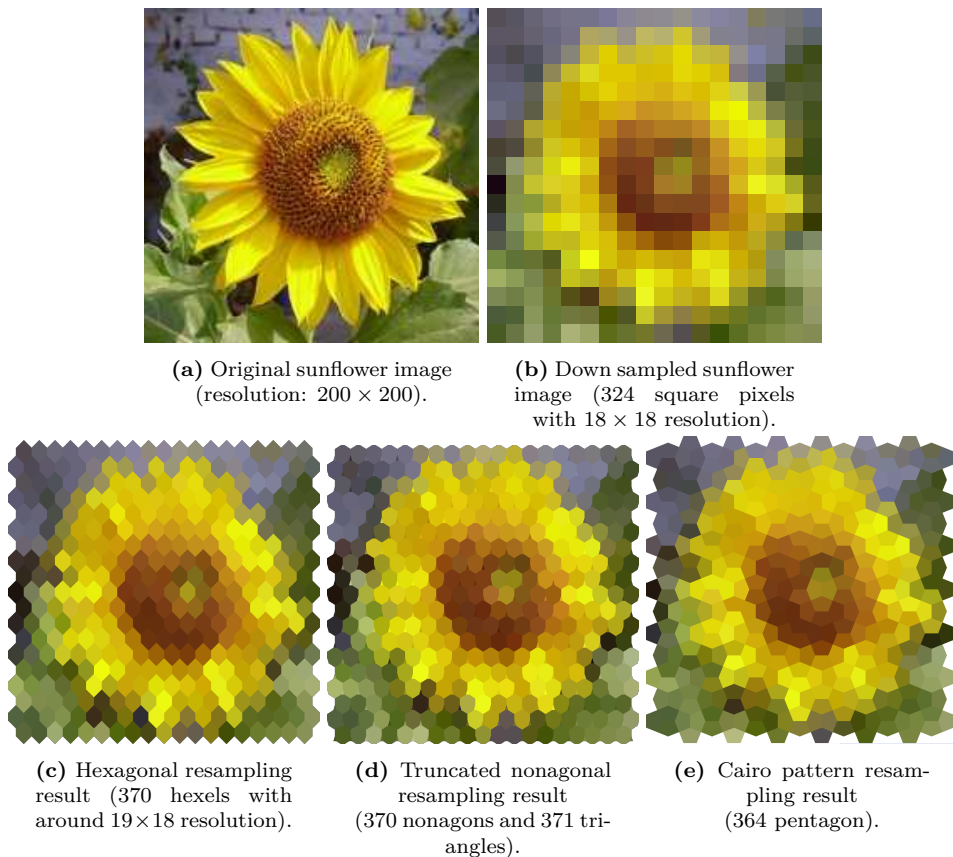


Figure 5. Results of various resampling processes: sunflower.

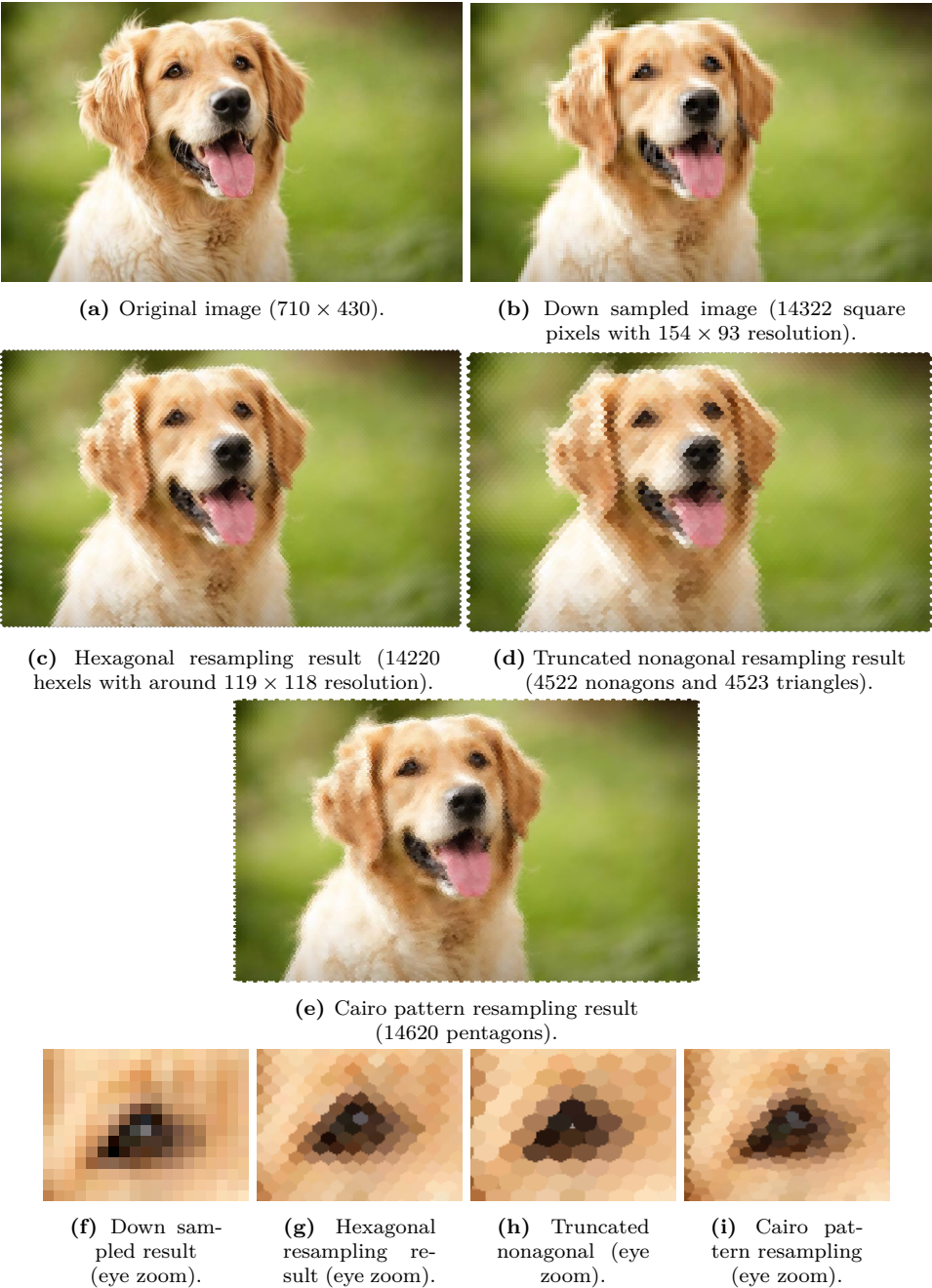


Figure 6. Results of various resampling processes: dog and left eye zoom.

show possible advantages of image processing algorithms on non-traditional grids, in the practice. At this stage of research, one can only ‘see’ the differences of the images, as we can generate those. Thus, here some images and their resampled variants are shown in various resolutions, see Figures 5 and 6.

The resampled images are stored by having both the geometric data and image data of cells. As each grid contains only of finitely many types (and orientations) of their cells (pixels): we need to store the coordinates of the cell, its type (not applicable for the hexagonal grid, as it has a sole type) and its color value.

7. Conclusions and Future Work

In this paper we give some details on our project to use images based on non-traditional grids in digital image processing. The first, ‘preparatory’ step of the project was to able to create test images on various grid. We have developed a software tool that is able to resample (high-quality) images on any given non-traditional grid (defined by its kernel).

While the current work is preparatory, the objective is to develop a framework capable of integrating existing image processing algorithms and supporting the design of novel ones. The system enables users to define custom grid structures, providing a platform for testing and validation. By maintaining consistent coordinate systems, the implemented processing algorithms remain applicable across different grid configurations.

Although there is a naive approach to store the resampled images, it is generally not efficient in terms of storage places. For the square grid various image compression strategies are also used. The next steps of the research line are more grid dependent, i.e., to find efficient way(s) to store the resampled images based on the structure and symmetry of the underlying grid. This can be based on a nice-structured coordinate system for the grid. For some grids, as we have mentioned, such coordinate system exists. However for some others, we may try to develop a coordinate system based on the kernel design (see also the related paper [22]).

References

- [1] H. M. E. I. M. ARNAOOT: *Hexagon path planning algorithm*, IET Radar, Sonar & Navigation 16.12 (2022), pp. 1895–1911, DOI: <https://doi.org/10.1049/rsn2.12250>.
- [2] G. Borgefors: *A semiregular image grid*, J. Vis. Commun. Image Represent. 1.2 (1990), pp. 127–136, DOI: [10.1016/1047-3203\(90\)90002-D](https://doi.org/10.1016/1047-3203(90)90002-D).
- [3] G. Borgefors: *Distance transformations on hexagonal grids*, Pattern Recognit. Lett. 9.2 (1989), pp. 97–105, DOI: [10.1016/0167-8655\(89\)90042-1](https://doi.org/10.1016/0167-8655(89)90042-1).
- [4] V. E. BRIMKOV, R. P. BARNEVA: *Honeycomb vs square and cubic models*, in: 8th International Workshop on Combinatorial Image Analysis, IWCIA 2001, Philadelphia, PA, USA, August 23-24, 2001, ed. by S. FOUREY, G. T. HERMAN, T. Y. KONG, Electronic Notes in Theoretical Computer Science, Elsevier, 2001, pp. 321–338, DOI: [10.1016/S1571-0661\(04\)80995-2](https://doi.org/10.1016/S1571-0661(04)80995-2).

- [5] J. H. CONWAY, H. BURGIEL, C. GOODMAN-STAUS: *The Symmetries of Things*, A K Peters/CRC Press, Apr. 2016, ISBN: 9781439864890, DOI: [10.1201/b21368](https://doi.org/10.1201/b21368), URL: <http://dx.doi.org/10.1201/b21368>.
- [6] M. J. E. GOLAY: *Hexagonal Parallel Pattern Transformations*, IEEE Trans. Computers 18.8 (1969), pp. 733–740, DOI: [10.1109/T-C.1969.222756](https://doi.org/10.1109/T-C.1969.222756).
- [7] T.-R. GRUMEZA, T.-A. LAZĂR, I. DRĂMNESC, G. KUSPER, K. PAPADOPOULOS, N. FACHANTIDIS, I. LEFKOS: *A Case Study in Building 2D Maps with Robots*, in: Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics - Volume 2: ICINCO, INSTICC, SciTePress, 2024, pp. 228–235, ISBN: 978-989-758-717-7, DOI: [10.5220/0013002200003822](https://doi.org/10.5220/0013002200003822).
- [8] N. P. HARTMAN, S. L. TANIMOTO: *A hexagonal pyramid data structure for image processing*, IEEE Trans. Syst. Man Cybern. 14.2 (1984), pp. 247–256, DOI: [10.1109/TSMC.1984.6313207](https://doi.org/10.1109/TSMC.1984.6313207).
- [9] I. HER: *A symmetrical coordinate frame on the hexagonal grid for computer graphics and vision*, ASME J. Mech. Design 115.3 (1993), pp. 447–449, DOI: [10.1115/1.2919210](https://doi.org/10.1115/1.2919210).
- [10] I. HER: *Geometric transformations on the hexagonal grid*, IEEE Trans. Image Process. 4.9 (1995), pp. 1213–1222, DOI: [10.1109/83.413166](https://doi.org/10.1109/83.413166).
- [11] I. HER, C. YUAN: *Resampling on a Pseudo-hexagonal Grid*, CVGIP Graph. Model. Image Process. 56.4 (1994), pp. 336–347, DOI: [10.1006/CGIP.1994.1030](https://doi.org/10.1006/CGIP.1994.1030).
- [12] G. KOVÁCS, B. NAGY, N. D. TURDAY: *Distance on the Cairo pattern*, Pattern Recognit. Lett. 145 (2021), pp. 141–146, DOI: [10.1016/J.PATREC.2021.02.002](https://doi.org/10.1016/J.PATREC.2021.02.002).
- [13] T. LUKIC, B. NAGY: *Binary tomography on the Cairo pattern*, in: Combinatorial Image Analysis - 23rd International Workshop, IWCIA 2025, Szeged, Hungary, September 24–26, 2025, Revised Selected Papers, ed. by P. BALÁZS, R. P. BARNEVA, V. E. BRIMKOV, A. NAGY, Lecture Notes in Computer Science 15985, Springer, 2026, to appear, DOI: [10.1007/978-3-032-19347-6_15](https://doi.org/10.1007/978-3-032-19347-6_15).
- [14] T. LUKIC, B. NAGY: *Regularized binary tomography on the hexagonal grid*, Physica Scripta 94 (2019), p. 025201, DOI: [10.1088/1402-4896/aafbc6](https://doi.org/10.1088/1402-4896/aafbc6).
- [15] S. MATEJ, G. T. HERMAN, A. VARDI: *Binary tomography on the hexagonal grid using Gibbs priors*, Int. J. Imaging Syst. Technol. 9.2-3 (1998), pp. 126–131, DOI: [10.1002/\(SICI\)1098-1098\(1998\)9:2/3<126::AID-IMA9>3.0.CO;2-D](https://doi.org/10.1002/(SICI)1098-1098(1998)9:2/3<126::AID-IMA9>3.0.CO;2-D).
- [16] L. MIDDLETON, J. SIVASWAMY: *Hexagonal Image Processing: A Practical Approach*, Advances in Pattern Recognition, Springer, 2005, ISBN: 978-1-85233-914-2, DOI: [10.1007/1-84628-203-9](https://doi.org/10.1007/1-84628-203-9).
- [17] B. NAGY: *A Khalimsky-Like Topology on the Triangular Grid*, in: Discrete Geometry and Mathematical Morphology - Third International Joint Conference, DGMM 2024, Florence, Italy, April 15–18, 2024, Proceedings, ed. by S. BRUNETTI, A. FROSINI, S. RINALDI, Lecture Notes in Computer Science, Springer, 2024, pp. 150–162, DOI: [10.1007/978-3-031-57793-2_12](https://doi.org/10.1007/978-3-031-57793-2_12).
- [18] B. NAGY: *Cellular topology and topological coordinate systems on the hexagonal and on the triangular grids*, Ann. Math. Artif. Intell. 75.1-2 (2015), pp. 117–134, DOI: [10.1007/S10472-014-9404-Z](https://doi.org/10.1007/S10472-014-9404-Z).
- [19] B. NAGY: *Non-traditional 2D Grids in Combinatorial Imaging – Advances and Challenges*, in: Combinatorial Image Analysis - 21st International Workshop, IWCIA 2022, Messina, Italy, July 13–15, 2022, Proceedings, ed. by R. P. BARNEVA, V. E. BRIMKOV, G. NORDO, Lecture Notes in Computer Science, Springer, 2022, pp. 3–27, DOI: [10.1007/978-3-031-23612-9_1](https://doi.org/10.1007/978-3-031-23612-9_1).
- [20] B. NAGY, T. LUKIC: *Binary Tomography on Triangular Grid Involving Hexagonal Grid Approach*, in: Combinatorial Image Analysis - 19th International Workshop, IWICIA 2018, Porto, Portugal, November 22–24, 2018, Proceedings, ed. by R. P. BARNEVA, V. E. BRIMKOV, J. M. R. S. TAVARES, Lecture Notes in Computer Science, Springer, 2018, pp. 68–81, DOI: [10.1007/978-3-030-05288-1_6](https://doi.org/10.1007/978-3-030-05288-1_6).

- [21] B. NAGY, M. SAADAT: *Digital distances on the 4-fold pentille tessellation*, Scientific Reports 15 (2025), p. 35601, DOI: [10.1038/s41598-025-17010-4](https://doi.org/10.1038/s41598-025-17010-4).
- [22] M. SAADAT, B. NAGY: *Digital Geometry on the Dual of Some Semi-regular Tessellations*, in: Discrete Geometry and Mathematical Morphology - First International Joint Conference, DGMM 2021, Uppsala, Sweden, May 24-27, 2021, Proceedings, ed. by J. LINDBLAD, F. MALMBERG, N. SLADOJE, Lecture Notes in Computer Science, Springer, 2021, pp. 283–295, DOI: [10.1007/978-3-030-76657-3_20](https://doi.org/10.1007/978-3-030-76657-3_20).
- [23] M. SAADETOGLU, A. AVKAN, B. NAGY, M. SAADAT: *On the Size of the Bijective Regions in Digital Rotations on the Hexagonal Grid*, in: International Symposium on Image and Signal Processing and Analysis, ISPA 2023, Rome, Italy, September 18-19, 2023, IEEE, 2023, pp. 1–6, DOI: [10.1109/ISPA58351.2023.10278914](https://doi.org/10.1109/ISPA58351.2023.10278914).
- [24] R. C. STAUNTON: *The design of hexagonal sampling structures for image digitization and their use with local operators*, Image Vis. Comput. 7.3 (1989), pp. 162–166, DOI: [10.1016/0262-8856\(89\)90040-1](https://doi.org/10.1016/0262-8856(89)90040-1).
- [25] N. D. TURGAY, B. NAGY, G. KOVÁCS, B. VIZVÁRI: *Weighted distances in the Cairo pattern*, Pattern Recognit. Lett. 166 (2023), pp. 105–111, DOI: [10.1016/j.patrec.2023.01.004](https://doi.org/10.1016/j.patrec.2023.01.004).
- [26] P. WIEDERHOLD, S. MORALES: *Thinning on Quadratic, Triangular, and Hexagonal Cell Complexes*, in: Combinatorial Image Analysis, 12th International Workshop, IWCIA 2008, Buffalo, NY, USA, April 7-9, 2008. Proceedings, ed. by V. E. BRIMKOV, R. P. BARNEVA, H. A. HAUPTMAN, Lecture Notes in Computer Science, Springer, 2008, pp. 13–25, DOI: [10.1007/978-3-540-78275-9_2](https://doi.org/10.1007/978-3-540-78275-9_2).
- [27] C. A. WÜTHRICH, P. STUCKI: *An algorithmic comparison between square- and hexagonal-based grids*, CVGIP Graph. Model. Image Process. 53.4 (1991), pp. 324–339, DOI: [10.1016/1049-9652\(91\)90036-J](https://doi.org/10.1016/1049-9652(91)90036-J).



Topology-Aware Neural Vectorization of Orienteering Maps: Contour Segmentation as a Demanding Test Case for Editable Reconstruction*

Péter Borbás, Ede Troll

Eszterházy Károly Catholic University, Eger, Hungary
borbas.peter@uni-eszterhazy.hu
troll.ede@uni-eszterhazy.hu

Abstract. We present a reproducible end-to-end pipeline for converting printed orienteering maps into editable vector layers. While the broader framework targets multiple ISOM symbol classes, this paper focuses on elevation contours as a particularly demanding test case because they are central to terrain interpretation, frequently occluded by overprinted symbols, and sensitive to topological degradation during raster-to-vector conversion.

To assess practical usefulness rather than raster similarity alone, we combine overlap metrics (Dice, IoU, Boundary-F1) with topology- and structure-aware measures, including Betti error, warping error, and graph-level error proxies. The experiments reveal a consistent raster–vector mismatch: models with nearly identical raster validation scores can produce materially different vector graphs after tracing. In particular, U-Net with a ResNet-18 encoder provided the best efficiency–quality trade-off despite near-tied raster scores with heavier variants, while topology-aware losses improved topology-related raster metrics without a comparably strong advantage in the post-vectorization evaluation. These results show that editable map reconstruction must be evaluated end-to-end in the vector domain, since raster overlap alone is not a reliable proxy for downstream structural fidelity or subsequent editing effort.

Keywords: orienteering maps, map vectorization, contour extraction, U-Net, topology-aware evaluation, raster-to-vector conversion

*This research was supported by the EKÖP-25-I-1 scholarship program.

1. Introduction

This paper examines whether elevation contours can be recovered from printed orienteering maps in a form suitable for editable vector reconstruction. In the broader TANVOM framework, contour extraction serves as a demanding test case for multi-layer recovery.

Printed orienteering maps are a difficult vectorization domain. They are standardized by the International Specification for Orienteering Maps (ISOM), yet visually dense and heavily layered [8]. Contours, paths, vegetation, and other symbols often overlap or partially occlude one another, while some interruptions are cartographically valid rather than erroneous. As a result, classical thresholding–thinning–tracing pipelines are often unstable on noisy or scanned inputs, especially when semantic context is needed to disambiguate overlapping structures.

Contours are particularly suitable for evaluation. They are central to terrain interpretation, but also topologically fragile: small raster defects can easily turn into breaks, spurs, or false junctions after skeletonization and graph tracing. This makes contour extraction a useful benchmark for testing whether raster predictions remain useful after deterministic raster-to-vector conversion.

A further challenge is evaluation. In practice, the final artifact is not a segmentation mask but an editable vector graph. Raster overlap alone is therefore insufficient for model selection. Inspired in part by prior contour-vectorization work that emphasized topology-aware and editing-oriented assessment [20], we ask whether raster-level metrics reliably predict downstream vector quality, or whether model rankings change after tracing.

The main contributions are threefold. First, we describe a reproducible end-to-end pipeline from OCAD-derived symbol masks to editable GeoJSON/OMAP outputs. Second, we define an evaluation protocol that combines raster overlap, topology-aware diagnostics, and post-vectorization benchmarking under a fixed deterministic backend. Third, through controlled architecture and loss comparisons, we show that raster-domain metrics alone are weak predictors of downstream editability, so model selection should be performed end-to-end in the vector domain.

2. Related Work

Map vectorization has a long tradition in document analysis and GIS. Classical systems typically decompose the task into colour or intensity separation, raster cleanup, thinning, and geometric tracing. In historical-map workflows, colour thresholding and statistical pixel classification have been used to isolate thematic layers, after which morphology, skeletonization, and line-following convert raster masks into vector primitives [1, 7]. Practical vectorization tools and contour-specific workflows follow similar assumptions: if the input is clean and the target colour is well separated, thinning-based conversion can produce useful editable curves [14, 21]. However, these methods are local and parameter-sensitive. Paper texture, scan noise, colour drift, compression, and overprinted symbols can

change the appearance of the same cartographic object, and small raster errors may become broken polylines, false junctions, or unnecessary manual edits.

Contour extraction from topographic paper maps has also been studied as a dedicated problem. Earlier work on USGS paper maps combined colour separation, line tracking, and geographic-feature extraction to recover contour and auxiliary information [9]. Oka et al. studied contour vectorization from scanned topographic maps using a sequence of segmentation, thinning, and geometric reconstruction steps [13]. These approaches demonstrate that traditional pipelines can be effective under controlled map design and scanning conditions, but they have limited semantic understanding of whether a line interruption is intentional, whether nearby fragments should be connected, or whether a crossing is cartographically meaningful. In printed orienteering maps this limitation is amplified by the ISOM symbol system, where contours, paths, vegetation screens, rock features, erosion gullies, and point symbols may overlap in dense areas, and where some breaks are intentional for legibility [8].

Deep learning changed the front end of this pipeline by replacing hand-tuned colour rules with learned segmentation models. Encoder-decoder architectures such as U-Net are widely used because they combine fine spatial localization with contextual features [17]. More broadly, CNN-based segmentation has become a standard tool in image segmentation and remote sensing, where robustness to texture, local clutter, and appearance variation is crucial [11, 12, 23]. In historical-map research, semantic segmentation has been used to transfer across heterogeneous map styles [16], while road-network extraction from historical maps has been addressed with deep-learning models designed for variable symbols, sheet quality, and class imbalance [5, 19, 22]. These studies show that learned raster prediction can reduce the brittleness of threshold-based systems, but they also highlight a key limitation for cartographic production: a high-quality segmentation mask is not automatically an editable vector map.

Recent map-vectorization research therefore increasingly evaluates the full reconstruction pipeline. Chen et al. proposed a deep edge filtering approach in which a neural model predicts a simplified edge map that is then converted into closed shapes [3]; the same research direction later led to a benchmark for automatic historical-map vectorization [4]. This is close in spirit to our downstream-aware setup, because the neural stage and the deterministic geometric stage are considered together. However, much of this work targets region boundaries and closed polygonal objects. Thin linear objects have different failure modes, including fragmentation, duplicated centerlines, dangling endpoints, false junctions, and unstable graph connectivity. These errors may be visually small but costly in a cartographic editor.

The closest recent work to the present contour setting is the study by Vynikal and Pacina, who investigate automatic elevation-contour vectorization with a deep-learning approach and emphasize evaluation after vectorization [20]. Their work moves contour extraction beyond pixel overlap and toward edit-oriented assessment. Our paper follows the same downstream motivation, but differs in domain

and task constraints. Orienteering maps are competition-oriented, highly generalized, multi-layer documents governed by ISOM. This creates additional ambiguity between the printed symbol and the intended editable geometry, especially around valid breaks, overprint, and symbol families sharing similar colours.

A parallel line of research aims to improve the topology of thin-structure segmentation directly through the loss function. Topology-preserving losses can penalize changes in connected components and holes [6]; cDice encourages centerline overlap for tubular structures [18]; and Skeleton Recall Loss was introduced to preserve connectivity more efficiently for thin structures [10]. These losses are relevant to contour recovery because the downstream vector graph depends heavily on connectivity. Nevertheless, recent reassessments suggest that topology-based losses do not always outperform well-tuned baselines and may be sensitive to implementation and evaluation choices [15]. This motivates the central question of the present paper: whether topology-oriented improvements visible in the raster domain survive thresholding, skeletonization, graph tracing, and export into editable vector objects.

3. Problem Setting and Data

3.1. Cartographic domain and task focus

Orienteering maps are specialized topographic maps designed for fast terrain navigation. They encode landforms, runnability, vegetation, paths, and other navigationally relevant features in a compact standardized symbol system. In printed ISOM maps, contours coexist with black path symbols, erosion-related line symbols, vegetation overlays, and other content that may overlap spatially or appear visually similar.

The broader TANVOM framework targets multi-layer ISOM recovery, including linework and region layers. The current system already covers several groups, such as contour geometry, black-road geometry, rare line symbols, multi-class region bases, and multi-label region overlays. In this paper, however, we isolate the contour component and formulate it as a binary segmentation task in which ISOM 101 and 102 are merged into a single target.

This design is deliberate. For editable reconstruction, preserving contour continuity is more important than distinguishing ordinary and index contours during raster prediction. The task is therefore defined as geometry-first contour recovery, making it an appropriate demanding test case for downstream vector reconstruction.

3.2. OCAD-derived ground truth and dataset construction

Ground truth is generated from symbol-specific OCAD exports rather than manual pixel annotation. This is a major practical advantage of the domain: because the original cartographic content already exists in a structured authoring environment,

the training targets can be derived from authoritative symbol layers. The contour target is built by merging the ISOM 101 and 102 layers into a single binary mask. Training data are constructed from full map sheets by tiled sampling.

Table 1. Summary of the contour dataset used in the experiments.

Tile size	512×512 px
Stride	128 px
Augmented variants per base tile	4
Training samples	18,880
Validation samples	2,664
Total samples	21,544
Ground truth source	OCAD symbol-layer exports
Primary contour target	ISOM 101+102 merged

The augmentation policy is designed to preserve geometric correctness while simulating print and scan variability. Geometric transforms are restricted to discrete D4-type operations such as flips and right-angle rotations, so masks remain pixel-aligned with downstream tasks that depend on exact correspondence. Appearance augmentations simulate realistic degradations such as contrast changes, blur, motion blur, noise, JPEG-like compression, and mild color jitter. This matters because the practical deployment target is not a perfectly clean vector-native export, but printed or scanned map images whose artifacts can later be amplified by thinning and tracing.



Figure 1. Examples of augmented training tiles used for contour segmentation. The augmentations increase appearance variability while preserving the geometric consistency of the contour labels.

The processed tile-level dataset used in this study is publicly available on Zenodo [2].

3.3. Broader pipeline context

Although contours are the only quantitatively analyzed symbol family in this paper, the broader pipeline context is worth stating explicitly. In the full system, roads are treated in a geometry-first manner by merging ISOM 503–508 into a single roads-black mask, followed by type attribution on the traced polylines. Rare linear symbols are represented as per-code binary masks. Region layers are handled by a two-head formulation: a base landcover head predicts exactly one class per pixel, while overlay heads predict per-code binary masks that may overlap the base layer. A separate road-typing dataset is created by sampling RGB patches along skeletonized road pixels and storing them in a compact chunked format. This broader context is not analyzed in detail here, but it motivates why contour editability is treated as a demanding test case rather than as an isolated end in itself.

4. Method

4.1. End-to-end reconstruction pipeline

The proposed workflow consists of five reproducible stages. First, symbol-specific layers are exported from OCAD to obtain authoritative raster masks. Second, these masks and the corresponding RGB map sheets are converted into a tiled dataset with deterministic splitting and augmentation. Third, a neural segmentation model predicts contour probabilities on tiles or stitched full maps. Fourth, the raster predictions are converted into vectors by a fixed deterministic backend. Finally, the reconstructed linework is exported to editable formats such as GeoJSON and OpenOrienteering Mapper project files.

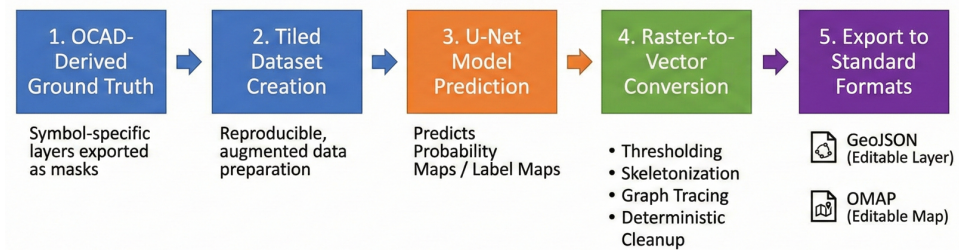


Figure 2. High-level overview of the TANVOM pipeline. Raster-to-vector conversion is treated as part of the evaluated method rather than as an informal post-processing step.

Because the practical output is an editable vector layer, evaluation is performed on the full reconstruction pipeline rather than on raster prediction alone.

4.2. Contour prediction models

For contour prediction, we use U-Net-family models with ResNet encoders. The main architecture comparison includes three variants: U-Net with a ResNet-18 encoder, U-Net with a ResNet-34 encoder, and U-Net++ with a ResNet-34 encoder. The goal is to determine which backbone yields the most useful contour vectors under a shared tracing policy, rather than merely which one maximizes a raster overlap metric.

In addition to standard BCE-style training, we also include a positive-weighted BCE setting that tends to produce thicker strokes. This serves as a controlled calibration test: if a metric is reliable for model selection, its ranking should remain meaningful after vectorization as well.

4.3. Deterministic raster-to-vector conversion

The contour model outputs a probability map, which is thresholded, skeletonized, converted into a graph, and traced into polylines. Simplification, artifact removal, endpoint handling, and graph cleanup are all governed by a fixed policy, after which the results are exported in standard editable formats.

Using a fixed backend is essential for fair comparison. It ensures that differences in final editability can be attributed to the learned raster predictions rather than to per-model tuning of the tracing stage.

4.4. Evaluation protocol

The evaluation is explicitly downstream-oriented. At the raster level, we compute Dice, IoU, and tolerance-aware Boundary-F1. While Dice and IoU summarize overlap, Boundary-F1 is more suitable for thin-line structures because it is less sensitive to small spatial shifts that may be acceptable in practice.

To assess structural correctness, we also compute warping error and Betti error on skeletonized representations. Warping error measures geometric mismatch between predicted and reference centerlines, whereas Betti error captures topological differences in connected components and holes. These metrics are evaluated both on the raster prediction and after vectorization by rasterizing the exported vectors back to the grid.

We further record graph-level proxies such as polyline counts, dangling endpoints, intersections, and generalized error sites obtained by clustering local anomalies. Since some contour interruptions are cartographically valid, topology is interpreted in a context-aware way and allowed breaks are excluded from editing-relevant error summaries.

The final stage is post-vectorization evaluation. After tracing the predicted linework into vectors, we rasterize the exported polylines back to the evaluation grid and recompute the same overlap and topology metrics. This provides a controlled measure of how much quality is preserved, or lost, during thresholding, thinning, tracing, and cleanup.

5. Experiments

5.1. Architecture screening and metric stress test

The first experimental block serves two purposes. The primary goal is to select the most suitable contour backbone among three U-Net-family variants: U-Net with a ResNet-18 encoder, U-Net with a ResNet-34 encoder, and U-Net++ with a ResNet-34 encoder. The secondary goal is to test the stability of the evaluation criteria under a controlled calibration shift induced by positive-class-weighted BCE.

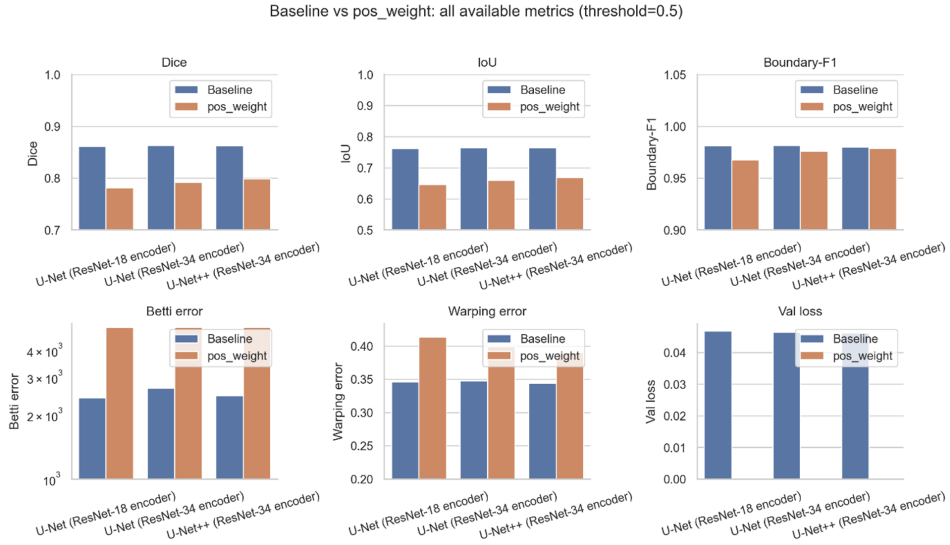
In the calibration-test setting, BCE is reweighted in favor of the sparse contour foreground using a positive-class weight. As expected for a thin-line task, this shifts the probability field toward thicker foreground responses. In raster space, such thickening typically lowers overlap quality and increases structural mismatch, since even small width changes strongly affect sparse masks. Under deterministic tracing, however, the effect is not monotonic: some post-vectorization metrics improve despite worse raster-domain scores, while the exported polyline count may increase because thicker responses can generate short spurs or locally duplicated centerlines after skeletonization. This calibration test is useful precisely because it reveals that metric rankings need not remain stable across the raster-to-vector transition.

For the architecture comparison itself, all three baselines were trained and evaluated under the same data split, augmentation policy, thresholding rule, and tracing backend. Their selected-epoch validation overlap metrics were nearly tied, so the final choice was driven mainly by downstream graph quality and training efficiency. On the proof-tile evaluation, the RN18 backbone produced the lowest generalized error count per tile while also requiring the least training time per epoch. Table 2 summarizes the screening result.

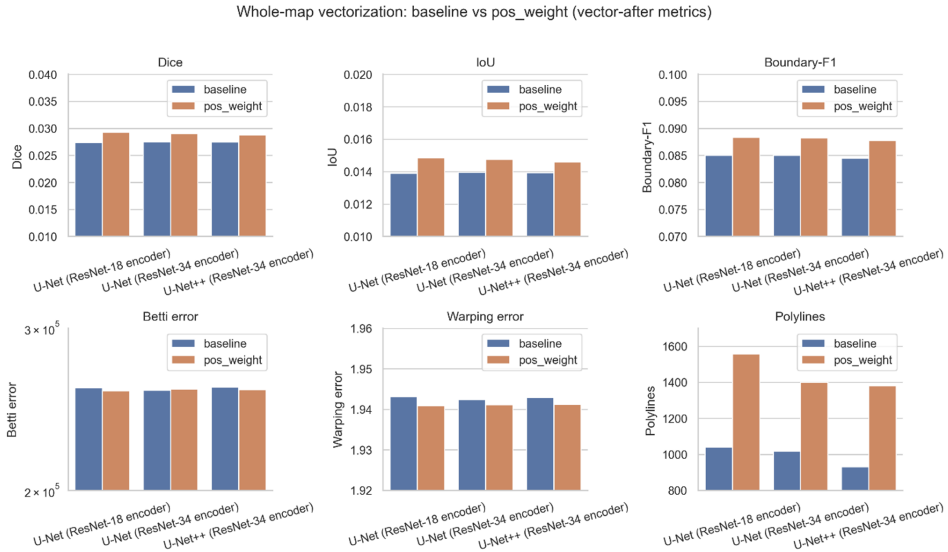
Table 2. Architecture screening for contour extraction. Dice, IoU, and BF1 are selected-epoch validation metrics; generalized errors per tile are measured after vectorization on the proof-tile subset. The RN18 model was selected because its raster scores are competitive while its downstream graph-error count and training cost are both lower.

Model	Dice	IoU	BF1	Gen. err./tile	Min./epoch
U-Net RN18	0.8616	0.7621	0.9814	22.50	12.50
U-Net RN34	0.8632	0.7649	0.9816	32.75	14.04
U-Net++ RN34	0.8629	0.7651	0.9787	24.50	29.75

The conclusion of this stage is not that architecture is irrelevant, but that raster overlap alone is insufficient for choosing among otherwise similar backbones. Under the shared tracing policy, RN18 provides the best efficiency–quality trade-off and is therefore used as the default contour backbone in the subsequent loss study.



(a) Raster-domain metric comparison between the baseline BCE model and the positive-class-weighted BCE variant, evaluated on the raw prediction outputs before vectorization.



(b) Whole-map metric comparison between the baseline BCE model and the positive-class-weighted BCE variant after deterministic vectorization. Unlike the raster-domain comparison, these results reflect the quality of the traced and re-rasterized contour vectors.

Figure 3. Effect of positive-class weighting before and after deterministic vectorization. The top panel shows raster-domain evaluation, while the bottom panel shows whole-map post-vectorization evaluation after tracing and re-rasterization.

5.2. Loss ablation under a fixed vectorization backend

The second experimental block evaluates whether topology-aware objectives improve the final editable contour vectors more reliably than overlap-oriented baselines. All runs use the RN18 backbone, the same train/validation split, the same augmentation protocol, the same inference threshold, and the same deterministic vectorization backend. The compared losses are BCE+Dice with Dice weights $\{0.0, 0.10, 0.25, 0.50\}$, BCE+soft-clDice, and BCE with an additional skeleton-recall term.

This ablation is designed to answer an end-to-end question rather than a purely raster one. If a topology-aware loss is genuinely useful for cartographic editing, then its benefit should remain visible after thresholding, skeletonization, graph tracing, cleanup, and export. A gain that appears only on raster masks but disappears after deterministic vectorization has limited practical value in the present workflow.

To keep the comparison interpretable, raster-domain and post-vectorization results are reported separately. Table 3 lists whole-map raster metrics, while Table 4 gives the corresponding post-vectorization results obtained by tracing the predictions into contour polylines and rasterizing the exported vectors back to the evaluation grid.

Table 3. Whole-map raster-domain results for the RN18 contour loss ablation. Topology-aware losses strongly reduce raster-domain Betti error, while BF1 remains within a narrow range across the competitive variants.

Loss	Dice (raster)	IoU (raster)	BF1 (raster)	Betti (raster)	Warp (raster)
BCE	1.640×10^{-2}	8.265×10^{-3}	0.977	1.802×10^5	1.920
BCE+Dice 0.10	1.677×10^{-2}	8.458×10^{-3}	0.978	1.380×10^5	1.923
BCE+Dice 0.25	1.733×10^{-2}	8.740×10^{-3}	0.976	1.178×10^5	1.922
BCE+Dice 0.50	1.710×10^{-2}	8.623×10^{-3}	0.978	1.011×10^5	1.924
BCE+soft-clDice	2.658×10^{-2}	1.347×10^{-2}	0.978	5.102×10^4	1.915
BCE+Skeleton-Recall	2.317×10^{-2}	1.172×10^{-2}	0.977	3.169×10^4	1.920

Table 4. Post-vectorization results after deterministic tracing and re-rasterization of exported contour polylines. In contrast to the raster-domain Betti reduction, vector-domain Betti and warping errors remain tightly clustered across the competitive variants.

Loss	Polylines	Dice (vector)	IoU (vector)	BF1 (vector)	Betti (vector)	Warp (vector)
BCE	929	6.395×10^{-3}	3.208×10^{-3}	0.916	2.411×10^5	1.902
BCE+Dice 0.10	915	3.291×10^{-3}	1.648×10^{-3}	0.953	2.417×10^5	1.916
BCE+Dice 0.25	909	2.706×10^{-3}	1.355×10^{-3}	0.958	2.416×10^5	1.915
BCE+Dice 0.50	900	2.689×10^{-3}	1.346×10^{-3}	0.959	2.416×10^5	1.913
BCE+soft-clDice	918	3.070×10^{-3}	1.537×10^{-3}	0.960	2.414×10^5	1.914
BCE+Skeleton-Recall	939	2.584×10^{-3}	1.293×10^{-3}	0.957	2.416×10^5	1.915

The raster-domain trend is clear. Relative to plain BCE, the topology-aware losses reduce raster Betti error substantially: soft-clDice lowers it from 1.802×10^5

to 5.102×10^4 , while Skeleton-Recall reduces it further to 3.169×10^4 . At the same time, raster BF1 remains tightly grouped in the 0.976–0.978 range, so the main separation appears in the topology-oriented metric rather than in boundary alignment.

The very low whole-map Dice and IoU values should be interpreted in the context of extremely sparse thin-line targets and strict re-rasterized comparison; in this setting, small centerline shifts or width differences can strongly depress overlap scores even when the reconstructed contour geometry remains practically usable.

After vectorization, this separation is much smaller. Post-vectorization Betti error remains close to 2.41×10^5 across the competitive losses, warping error changes only marginally, and post-vectorization BF1 and IoU form a narrow band. Polyline counts are also similar, ranging from 900 to 939, indicating that the fixed tracing backend largely determines the final contour-graph structure.

Figure 4 supports the same conclusion. The lowest generalized error count is obtained by BCE+Dice with Dice weight 0.50 (1259). Thus, topology-aware losses improve raster topology, but do not provide a clear advantage in the final exported contour graph under the current vectorization policy.

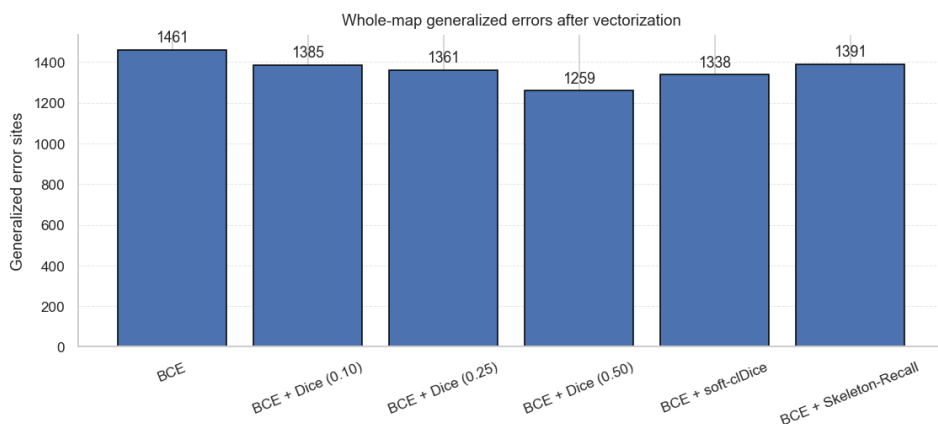


Figure 4. Whole-map generalized error sites after deterministic vectorization for the evaluated RN18 loss variants. Lower values indicate fewer clustered topological repair locations in the exported contour graph.

From a practical perspective, BCE+Dice with Dice weight 0.50 is the strongest default among the tested losses. It combines the best raster BF1, competitive post-vectorization BF1, the lowest polyline count among the strongest variants, and the lowest generalized error count after vectorization. Under the current fixed backend, the main limitation is therefore not the absence of topology-aware supervision, but the limited extent to which raster-domain improvements survive deterministic raster-to-vector conversion.

6. Results and Discussion

6.1. Raster–vector mismatch

The main finding of the paper is the mismatch between raster-domain quality and downstream vector quality. In both the architecture screening and the RN18 loss ablation, rankings changed after deterministic tracing: settings that improved raster-domain metrics did not necessarily yield better exported contour graphs. This shows that raster overlap and raster topology metrics are not sufficient as stand-alone criteria for model selection in editable contour reconstruction.

This is practically important because the target artifact is an editable vector layer rather than a raster mask. Subsequent editing effort is driven by fragmentation, duplicated centerlines, spurs, unstable junctions, and unexplained endpoints, all of which may emerge or worsen during thresholding, skeletonization, and tracing. For this reason, post-vectorization metrics and graph-level repair proxies are more informative than raster scores alone.

Whole-map correlations across the six RN18 loss variants support the same conclusion. Although the sample is small, raster-domain and post-vectorization rankings show weak agreement under the fixed tracing backend.

Table 5. Whole-map raster-to-vector correlation across the six RN18 loss variants. Low absolute correlations indicate only weak agreement between raster-domain and post-vectorization rankings.

Metric	Pearson r
Dice	-0.356
IoU	-0.356
Boundary-F1	-0.048
Betti error	-0.411
Warping error	0.127

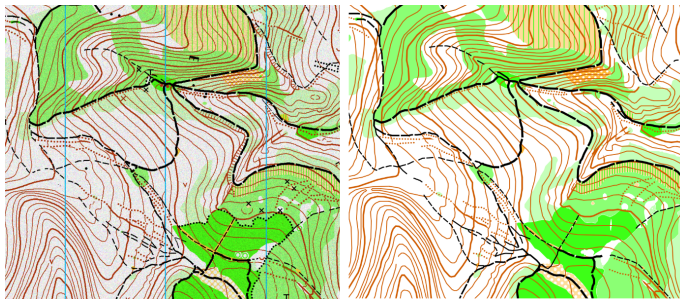


Figure 5. A qualitative end-to-end reconstruction example showing how contour extraction fits into the broader editable-layer pipeline.

The same pattern appears in the post-vectorization repair proxies. Under the current backend, BCE+Dice with Dice weight 0.50 produces the lowest generalized error count and the lowest polyline count among the strongest variants, while topology-aware losses do not provide a clear downstream advantage despite their stronger raster-domain Betti reductions.

6.2. Implications for broader recovery

Although the experiments are contour-centered, the implications extend to the broader TANVOM framework. As additional symbol families are added, evaluation must remain end-to-end, whole-map based, and sensitive to downstream vector quality. The surrounding experiments also suggest that different symbol groups require different calibration and that aggregate raster scores may hide class-specific failure modes. This makes head-specific postprocessing and class-aware checkpoint selection more realistic than a single global criterion.

6.3. Limitations and future work

The study intentionally focuses its quantitative analysis on contours. This makes the experiments controlled and highlights a difficult thin-line case, but it also limits how far the numerical conclusions can be generalized to other ISOM symbol families. Roads, rare linear symbols, landcover regions, and overlay patterns are already represented in the broader TANVOM pipeline, yet their full quantitative evaluation is left for future work. A second limitation is that editing effort is estimated through structural proxies such as unexplained endpoints, intersections, polyline counts, and generalized error clusters. These quantities are closer to the practical goal than pixel overlap, but they are still only approximations. A user-based editor study, measuring actual correction time and mapper judgement, would provide a stronger validation of the proposed error proxies.

Future work should therefore extend the benchmark to additional symbol classes, perform postprocessing and calibration sweeps under the same vector-after protocol, and compare structural proxy scores with real manual editing sessions. This would make it possible to separate three sources of error more clearly: neural recognition, probability calibration, and deterministic graph reconstruction.

7. Conclusion

This paper presented a contour-centered study of topology-aware neural vectorization for printed orienteering maps. Contours were used as a demanding test case because they are both cartographically central and highly sensitive to topological degradation during raster-to-vector conversion. The proposed workflow combines OCAD-derived ground truth, tiled training data, neural contour prediction, deterministic vectorization, and editable export in a single reproducible pipeline.

The experiments lead to a consistent conclusion: raster-domain metrics remain useful, but they are not sufficient for selecting models when the final output is an editable vector layer. Under the present fixed backend, U-Net with a ResNet-18 encoder provided the best efficiency–quality trade-off, while BCE+Dice with Dice weight 0.50 was the strongest practical default among the tested losses.

More broadly, the study shows that map-reconstruction benchmarks should be aligned with the final artifact. If the goal is editable vector recovery, then evaluation must be performed end-to-end in the vector domain.

References

- [1] A. G. AUFFRET, A. KIMBERLEY, J. PLUE, ET AL.: *HistMapR: Rapid Digitization of Historical Land-Use Maps in R*, *Methods in Ecology and Evolution* 8.11 (2017), pp. 1453–1457.
- [2] P. BORBÁS, E. TROLL: *TANVOM v1: Dataset for Orienteering Map Vectorization*, 2026, DOI: [10.5281/zenodo.19437536](https://doi.org/10.5281/zenodo.19437536), URL: <https://doi.org/10.5281/zenodo.19437536>.
- [3] Y. CHEN, E. CARLINET, J. CHAZALON, C. MALLET, B. DUMÉNIU, J. PERRET: *Vectorization of Historical Maps Using Deep Edge Filtering and Closed Shape Extraction*, in: *Lecture Notes in Computer Science*, Springer, 2021, pp. 510–525.
- [4] Y. CHEN, J. CHAZALON, E. CARLINET, M. Ô. V. NGOC, C. MALLET, J. PERRET: *Automatic Vectorization of Historical Maps: A Benchmark*, *PLOS ONE* 19 (2024), e0298217.
- [5] B. EKIM, E. SERTEL, M. E. KABADAYI: *Automatic Road Extraction from Historical Maps Using Deep Learning Techniques: A Regional Case Study of Turkey in a German World War II Map*, *ISPRS International Journal of Geo-Information* 10.8 (2021), p. 492.
- [6] X. HU, F. LI, D. SAMARAS, C. CHEN: *Topology-Preserving Deep Image Segmentation*, in: *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [7] I. IOSIFESCU, A. TSORLINI, L. HURNI: *Towards a Comprehensive Methodology for Automatic Vectorization of Raster Historical Maps*, *e-Perimetre* 11.2 (2016), pp. 57–76.
- [8] *ISOM 2017-2: International Specification for Orienteering Maps*, Revision 6, January 2024, International Orienteering Federation, 2024.
- [9] A. KHOTANZAD, E. ZINK: *Contour Line and Geographic Feature Extraction from USGS Color Topographical Paper Maps*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.1 (2003), pp. 18–31.
- [10] Y. KIRCHHOFF, M. R. ROKUSS, S. ROY, ET AL.: *Skeleton Recall Loss for Connectivity Conserving and Resource Efficient Segmentation of Thin Tubular Structures*, in: *European Conference on Computer Vision (ECCV 2024)*, Springer, 2024, pp. 218–234.
- [11] Y. LECUN, Y. BENGIO, G. HINTON: *Deep Learning*, *Nature* 521.7553 (2015), pp. 436–444.
- [12] S. MINAEI, Y. Y. BOYKOV, F. PORIKLI, A. J. PLAZA, N. KEHTARNAVAZ, D. TERZOPOULOS: *Image Segmentation Using Deep Learning: A Survey*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (2021), pp. 3523–3542.
- [13] S. OKA, A. GARG, K. VARGHESE: *Vectorization of Contour Lines from Scanned Topographic Maps*, *Automation in Construction* 22 (2012), pp. 192–202.
- [14] OPENORIENTEERING: *CoVe, the Contour Vectorizer (Mapper Manual)*, Accessed: 2026-04-17, 2020, URL: <https://www.openorienteering.org/mapper-manual/pages/cove.html>.
- [15] OPENREVIEW: *Reassessing the Utility of Topology Based Losses for Image Segmentation*, *ICLR 2025 Workshop submission*, Accessed: 2026-04-17, 2025, URL: <https://openreview.net/forum?id=bpBDx20prB>.

- [16] R. PETITPIERRE, F. KAPLAN, I. DI LENARDO: *Generic Semantic Segmentation of Historical Maps*, in: Proceedings of the Computational Humanities Research Conference, 2021, pp. 228–248.
- [17] O. RONNEBERGER, P. FISCHER, T. BROX: *U-Net: Convolutional Networks for Biomedical Image Segmentation*, in: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, vol. 9351, Springer, 2015, pp. 234–241.
- [18] S. SHIT, J. C. PAETZOLD, A. SEKUBOYINA, I. EZHOV, A. UNGER, K. PÖPPEL, G. TETTEH, B. H. MENZE: *clDice – A Topology-Preserving Loss Function for Tubular Structure Segmentation*, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16560–16569.
- [19] J. H. UHL, S. LEYK, Y.-Y. CHIANG, C. A. KNOBLOCK: *Towards the Automated Large-Scale Reconstruction of Past Road Networks from Historical Maps*, Computers, Environment and Urban Systems 94 (2022), p. 101794.
- [20] J. VYNIKAL, J. PACINA: *Automatic Elevation Contour Vectorization: A Case Study in a Deep Learning Approach*, ISPRS International Journal of Geo-Information 14.5 (2025), p. 201, DOI: [10.3390/ijgi14050201](https://doi.org/10.3390/ijgi14050201).
- [21] WINTOPO: *Vectorisation – Thinning (Help Documentation)*, Accessed: 2026-04-17, n.d. URL: <https://www.wintopo.com/help/html/vectorise.htm>.
- [22] Y. ZHAO, G. WANG, J. YANG, T. LI, Z. LI: *AU3-GAN: A Method for Extracting Roads from Historical Maps Based on an Attention Generative Adversarial Network*, Journal of Geovisualization and Spatial Analysis 8 (2024), p. 26.
- [23] X. X. ZHU, D. TUIA, L. MOU, G.-S. XIA, L. ZHANG, F. XU, F. FRAUNDORFER: *Deep Learning in Remote Sensing: A Comprehensive Review*, IEEE Geoscience and Remote Sensing Magazine 5.4 (2017), pp. 8–36.



Assessing Software Metrics and Complexity in Student and Open-Source Projects Using Static Analysis

Péter János Császár, Máté Cserép

Eötvös Loránd University,
Faculty of Informatics,
Hungary
csaszarpeter@inf.elte.hu
mcserep@inf.elte.hu

Abstract. Maintaining code quality remains a significant challenge in both educational and professional software development. This paper presents the development and evaluation of static code analysis tools aimed at identifying structural issues, such as complexity, cohesion, and maintainability in C# projects. The analyzers were applied to student projects from a Software Technology course. Code metrics were computed across multiple development milestones to track trends in code quality over time.

In addition to student projects, a diverse set of publicly available open-source C# projects was analyzed to evaluate the behavior of the implemented metrics in real-world software systems. These projects vary in size, domain, architectural style, and development practices, enabling a broader assessment of the robustness and general applicability of the approach. Due to the wide range of targeted .NET versions and dependency configurations across projects, the analysis was conducted using a Docker-based environment to provide controlled and reproducible execution contexts.

The results demonstrate that metric-based static analysis can effectively support the early detection of problematic code segments and provide actionable insights to improve software quality. The approach is adaptable and shows potential for integration into continuous development workflows in both academic and real-world environments.

Keywords: static code analysis, software metrics, code quality, maintainability, cohesion, complexity, C#, student projects, open-source software

1. Introduction

The increasing size and complexity of modern software systems place growing demands on both developers and educators to ensure long-term code quality. While conventional static analysis tools have proven effective in identifying syntax errors, type mismatches, and some well-known anti-patterns [9, 15], they often fall short in capturing deeper structural issues that affect software quality, such as complexity, maintainability, and readability. By quantifying specific properties such as cyclomatic complexity [19], cohesion, and parameter count in functions, software metrics can reveal patterns that may not be immediately obvious through traditional analysis methods [24, 28].

In the context of university assignment and submission management systems – such as TMS (Task Management System)¹ and Submitty² – previous research has already explored the application of automatic static code analysis to support the evaluation of student submissions [3, 14, 25]. Building on this foundation, our research aims to extend these tools to enable complexity and cohesion-based analysis of student projects written in C#, using various software metrics, and focusing on the automatic detection of problematic code segments and code smells. The metrics examined and implemented during the research include *Lack of Cohesion of Methods (LCOM)* [16], *Bumpy Road* [5], *Functional Parameter Count*, *Maintainability Index* [20], *Class coupling* [1], and *Cyclomatic Complexity* [19]. The implementation is based on the well-known .NET Compiler Platform SDK (Roslyn) [21]. These metrics can be used to determine the complexity and cohesion of the code. By continuously monitoring these metrics, it becomes possible to track their fluctuations throughout the development lifecycle and to evaluate the coherence of the system at the module and type levels.

The main contributions of this paper are the implementation of code metric analyzers based on the Roslyn platform and their application to both student and open-source C# projects. Furthermore, the study provides an empirical evaluation of these metrics, including threshold calibration using the elbow method and a comparative analysis across educational and real-world software systems.

2. Related work

Static analysis has been widely used to support the automated evaluation of student programming submissions. KASZAB, CSERÉP demonstrated that static analyzers can effectively identify common programming errors and coding standard violations in large collections of C++ and C# student solutions [15]. Their work led to the integration of static analysis into the Task Management System (TMS), showing that automated feedback can improve submission quality while reducing instructor workload.

¹<https://tms-elte.gitlab.io>

²<https://submitty.org/>

Beyond rule-based analysis, software metrics have long been used to assess code quality, complexity, and maintainability. A comprehensive review by NUÑEZ-VARELA ET AL. summarizes the use of code metrics across multiple programming paradigms, highlighting the prevalence of object-oriented metrics such as CK metrics, Lines of Code, and Cyclomatic Complexity in quality assessment and fault prediction [24]. Earlier work at NASA's Software Assurance Technology Center [28] emphasizes that software defects are often introduced early in development, motivating the use of metrics to support early detection and prevention.

Software metrics have also been applied to defect prediction using machine learning techniques. REBRO, CHREN, ROSSI evaluated multiple object-oriented metrics on large Java codebases and found that cohesion- and structure-related metrics outperform simple size-based measures such as Lines of Code [27]. Structural metrics have been used to study software stability over time, for example, through coupling-based instability measures [18, 29].

Cohesion metrics, particularly the Lack of Cohesion of Methods (LCOM), have been extensively studied. KIRĞIL, AYYILDIZ compared several LCOM variants and showed that LCOM5 provides a more sensitive measure of cohesion changes after refactoring [16]. However, applying cohesion metrics in practice requires careful threshold selection. Mishra [22] highlights that metric thresholds are highly context-dependent, while FILÓ, BIGONHA, FERREIRA provide empirical evidence that empirically tuned thresholds can reduce false positives in quality assessment [7].

Finally, prior work in software engineering education shows that students often struggle to refactor complex code [26], reinforcing the need for automated tools that can highlight problematic code segments early and guide targeted instructional intervention.

3. Methodology

The metrics analyzer must fulfill certain requirements: some of these are basic concepts for analyzers, others are special requirements for making a possible integration into assignment grading systems easier.

1. The tool must be able to perform code metrics calculation on a given C# solution.
2. The threshold values for these metrics should be adjustable from a configuration file.
3. The tool must be capable to produce a machine-interpretable output (e.g. a XML file), in addition to a summarized console output. This detailed output shall contain the metric results segmented by methods or classes, showing the corresponding metric values. The output path should also be configurable.
4. The analyzer should be applicable to arbitrary C# projects targeting the .NET platform without requiring project-specific modifications. Only a so-

lution or project file should be needed, while unsupported frameworks or custom build procedures may require additional configuration.

The analyzer relies on the Roslyn compiler platform to construct semantic models from C# projects. In most cases, providing a solution or project file is sufficient, and dependencies are restored automatically through the standard .NET build process. The approach has been successfully applied to WPF, Avalonia, Unity, and ASP.NET-based projects.

3.1. Implemented code metrics

In this section, the implemented code smells are presented.

3.1.1. Bumpy Road

This analyzer evaluates the nesting depth of statements within a method body. If the method body is empty or contains no statements, it is skipped. Otherwise, the nesting depth is computed recursively for each relevant statement. Relevant statements are *if*, *for*, *while*, *do*, *switch*, *block*, *local declaration statement* and *expression statement*. The result of the metrics is the average depth across all statements. The diagnostic is reported if this value exceeds the threshold defined in the configuration [5].

3.1.2. Functional Parameter Count

Methods with a high number of parameters are harder to read and understand, and they often indicate that the method does not have a single responsibility or that the required data is not properly encapsulated. A long parameter list also increases coupling to callers, makes call sites noisier, and complicates testing and reuse. Clean Code explicitly recommends keeping the number of function parameters low (ideally zero to two, and avoiding three or more when possible), since each additional parameter raises the cognitive load and the chance of incorrect usage [17].

This method-level analyzer counts the parameters for a given method and reports the result if it is higher than the configured threshold.

3.1.3. Lack of Cohesion of Methods

The LCOM4 metric [13] measures how well the methods in a class are connected by calling each other or accessing common fields. A cohesive class will have few independent method groups.

The LCOM5 is a stricter cohesion metric than LCOM4. It calculates cohesion based on the fields accessed by all methods of a class. If no fields are shared across methods, LCOM5 value approaches 1, indicating poor cohesion; if many fields are shared, the LCOM5 value approaches 0, indicating good cohesion. There are multiple approaches to compute the LCOM5 metric and studies have shown

[16] that the variant defined by Henderson and Sellers [11] is superior. If l is the number of attributes, k is the number of methods, and a is the sum of the number of distinct attributes, that are accessed by every method in a class, then LCOM5 is calculated as follows:

$$\text{LCOM5} = \frac{a - kl}{l - kl}$$

3.1.4. Maintainability index

Calculates an index value between 0 and 100 that represents the relative ease of maintaining the code. Higher values represent better maintainability. If HV is the Halstead Volume, CC is Cyclomatic Complexity and LC is the Lines of Code, then the maintainability index can be calculated as follows [20]:

$$\text{MI} = \max\left(0, \frac{100}{171} \times \left(171 - 5.2 \times \ln(HV) - 0.23 \times CC - 16.2 \times \ln(LC)\right)\right)$$

3.1.5. Cyclomatic complexity

Cyclomatic complexity measures the structural complexity of code by counting the number of linearly independent execution paths within a method. It is primarily influenced by the presence of control-flow constructs such as conditional branches and loops. While this metric is conceptually related to the implemented *Bumpy Road* code smell, the two can yield different results in practice.

The Bumpy Road metric focuses on sudden changes in complexity across consecutive code segments, capturing irregular growth patterns within a method. In contrast, cyclomatic complexity reflects the overall decision density of a method, regardless of how that complexity is distributed. As a result, a method with evenly distributed control flow may exhibit high cyclomatic complexity without triggering the Bumpy Road smell, whereas a method with localized spikes in complexity may be flagged by Bumpy Road despite having a moderate overall cyclomatic complexity.

3.1.6. Class coupling

Class Coupling measures the degree of coupling to unique classes through parameters, local variables, return types, method calls, generic or template instantiations, base classes, interface implementations, fields defined on external types, and attribute decorations. Good software design emphasizes high cohesion and low coupling. High coupling suggests a design that is difficult to reuse and maintain due to its numerous interdependencies with other types.

3.2. Determining threshold values for implemented analyzers

To minimize false positives and undetected faults, it is crucial to establish appropriate threshold values for software metrics, beyond which metric analyzers should

trigger warnings [7, 22]. Although the literature provides several recommended threshold values for commonly used software metrics [6, 17, 23], these recommendations are typically derived from observations on large-scale industrial software systems. As such, they may not be directly applicable to student projects, which differ significantly in size, development practices, and architectural complexity. For this reason, a dedicated threshold analysis was conducted on the implemented metrics based on the collected student project data.

The threshold analysis was performed for each metric by evaluating several threshold values on the student project dataset. To identify appropriate thresholds, the elbow method was applied to the cumulative number of warnings produced at each threshold value [2, 31]. The examined ranges were selected around commonly recommended values from the literature and adjusted according to the metric domain. For each metric, the total number of reported violations was plotted against the corresponding threshold values, and the knee point was automatically determined using the Kneedle algorithm [30], assuming a convex, monotonically decreasing curve. The threshold corresponding to the detected knee point was selected as the optimal value.

In contrast to other metrics, no robust threshold could be identified for LCOM5 and the Maintainability Index. For LCOM5, this is mainly due to its limited discriminative power: cohesion metrics in the LCOM family lack consensus, and prior studies have shown that they may provide inconsistent or misleading representations of class cohesion [4]. In our experiments, warning counts remained high across a wide range of thresholds, indicating that LCOM5 is overly sensitive for threshold-based classification.

For the Maintainability Index, the issue lies in its formulation and interpretation. The metric was originally derived from a limited set of procedural systems and does not explicitly capture object-oriented properties, while its threshold values are inconsistent across literature and tools. Recent work has also shown that different variants of the Maintainability Index can lead to differing assessments [10, 12].

Therefore, thresholds based on the literature were applied; however, both metrics should be interpreted with caution and are more suitable for comparative trend analysis than for strict threshold-based warning generation.

The resulting threshold values for each implemented metric are summarized in Table 1.

4. Analyzing student projects

This section presents the results of the code metrics analysis conducted on student projects from previous semesters of the Software Technology course at our university. Students implement a predefined game in groups of 3–4 members under the supervision of an instructor acting as product owner and, optionally, a senior student acting as scrum master.

Four milestones are defined, three of which involve implementation. Following

Table 1. Threshold values.

Metric	General thresholds	Applied on student projects
Bumpy Road code smell	4	2
Functional parameters	5	2
Lack of Cohesion (LCOM4)	1	4
Lack of Cohesion (LCOM5)	0.5	0.5
Maintainability index	65	65
Cyclomatic complexity	10	6
Class coupling	9	15

the design and requirements phase, students deliver two prototypes (approximately 30% and 90% of the use cases), followed by a final version conforming to clean-code principles [17], including testing and continuous integration. The last three milestones were considered in the analysis.

Although projects may be implemented in arbitrary languages, only C# projects were included. Common frameworks include *WPF*, *Avalonia*, *Unity*, and *Godot*.

The student projects are version-controlled and managed using GitLab. For each project, the last commit before each implementation milestone deadline on the default branch was retrieved using the GitLab API [8]. A total of 61 student projects were collected, of which 41 C# projects could be successfully analyzed after excluding repositories with build failures, missing dependencies, or other technical issues. Considering the three implementation milestones, up to 123 project versions were included in the analysis.

4.1. Visualization and discussion of the results

The goal of the analysis is to evaluate the effectiveness of the implemented code metrics in identifying potential code quality issues, such as high complexity, low cohesion, or maintainability concerns. Projects were analyzed at multiple milestones during their development, providing insights into how coding standards and structural requirements were followed throughout the project lifecycle. The reported values represent the number of warnings exceeding predefined thresholds, normalized by lines of code to ensure comparability across projects of different sizes.

In the line charts presented in Figure 1, and 2, selected metrics are visualized for the analyzed student projects. The trends reflect how the corresponding quality attributes evolve over time, with each line representing a single project.

Overall, the normalized summary warning counts show that most projects remain in the lower or middle range of the diagrams, indicating generally acceptable code quality across the analyzed milestones. At the same time, for most projects and metrics, an increase can be observed from the first to later milestones.

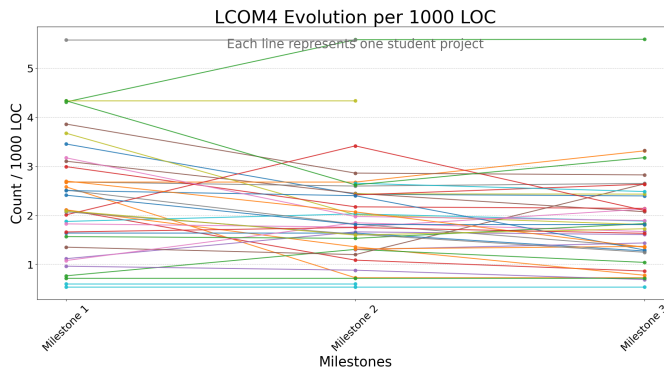


Figure 1. LCOM4 metric results on student projects.

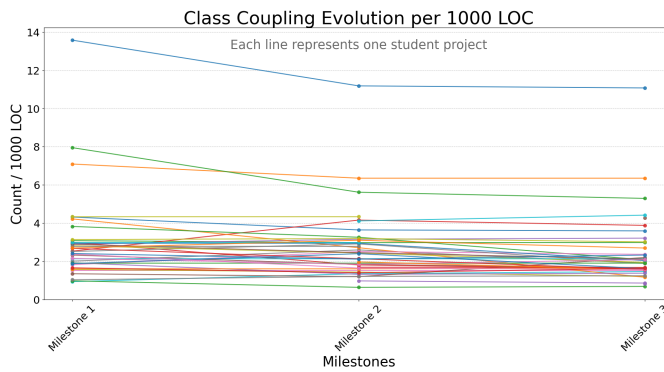


Figure 2. Class coupling metric results on student projects.

This suggests that as systems grow, the number of warnings exceeding the applied thresholds tends to increase, even after normalization by lines of code.

It is also important to note that the development effort between milestones is not uniform. Between the second and third milestones, students typically implement a significant portion of the functionality – approximately 60% of the total use cases – while later stages involve only minor additions. Consequently, a substantial deterioration in code quality in later phases should not necessarily be expected; in fact, improvements could be anticipated if teams perform refactoring and cleanup. The observed results partially align with this expectation: while some projects stabilize or slightly improve, a consistent overall improvement is not evident, suggesting that refactoring activities were limited in many cases.

The increase in warning counts is not uniform across all metrics. While cohesion- and coupling-related metrics show relatively consistent trends, the Bumpy Road metric exhibits more variability, with several projects showing fluctuations or even decreases between milestones. This indicates that certain structural characteristics

captured by Bumpy Road are less directly correlated with project growth and may be more sensitive to local code changes.

Despite these differences, the projects tend to cluster within comparable regions of the diagrams, suggesting a relatively similar overall quality level across teams. This consistency may indicate that the course structure, requirements, and supervision result in a broadly uniform development standard, even though individual projects differ in how their warning counts evolve over time.

5. Analyzing open-source projects

In addition to evaluating student projects, open-source projects were analyzed to further validate the implemented metrics and assess their applicability in a broader context. Open-source repositories provide diverse real-world codebases and enable evaluation of the metrics beyond the educational setting. In contrast to the student project analysis, which employed dataset-specific thresholds, open-source projects were evaluated using the threshold values recommended in the literature (Table 1).

Metrics were evaluated on selected major releases from five open-source .NET projects. In total, 22 tagged versions spanning several years were considered. Since the analyzer targets the .NET platform, only releases that could be successfully restored and analyzed were included. Versions requiring unsupported frameworks, failing to build, or containing unresolved dependencies were excluded. The varying number of analyzed versions across projects reflects differences in release histories and compatibility with the analysis tool.

5.1. Selected projects

The open-source projects were selected to provide diversity in terms of application domain, project size, and software architecture, while ensuring compatibility with the .NET ecosystem and the analysis tool. Preference was given to mature, actively maintained projects with multiple tagged releases available, enabling the study of metric evolution over time.

In total, 22 tagged versions spanning several years were considered. Since the analyzer targets the .NET platform, only releases that could be successfully restored and analyzed were included. Versions requiring unsupported frameworks, failing to build, or containing unresolved dependencies were excluded. The varying number of analyzed versions across projects reflects differences in release histories and compatibility with the analysis tool.

For the evaluation, the following open-source projects were selected: *i)* *OpenRA*³, a real-time strategy game engine inspired by the classic Command & Conquer: Red Alert; *ii)* *Jellyfin*⁴, a media server for organizing and streaming multimedia

³<https://github.com/OpenRA/OpenRA>

⁴<https://github.com/jellyfin/jellyfin>

content; *iii*) *nopCommerce*⁵, an open-source e-commerce platform; *iv*) *xUnit*⁶, a widely used unit testing framework for the .NET ecosystem; and *v*) *IronPython*^{3,7}, an open-source implementation of the Python programming language.

5.2. Visualization and discussion of the results

Figures 3 and 4 visualize the same metrics showcased and discussed for the student projects in subsection 4.1. The metric values were normalized by code size, specifically expressed as values per 1000 lines of code, to allow fair comparison between projects of different sizes.

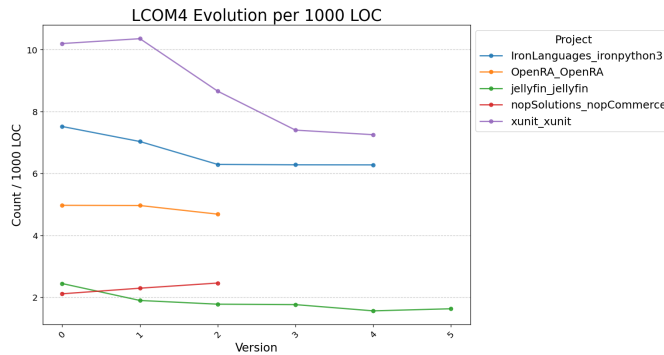


Figure 3. LCOM4 metric results on open-source projects.

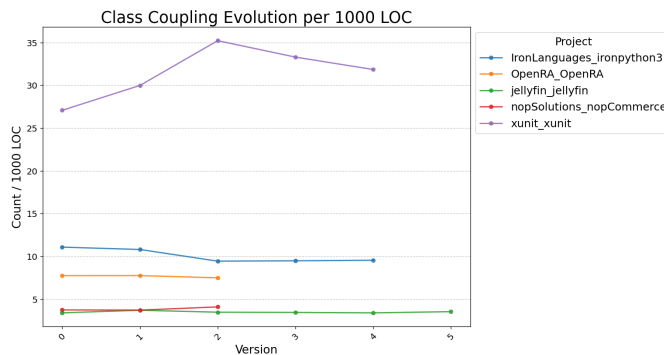


Figure 4. Class coupling metric results on open-source projects.

Overall, the results show low to moderate fluctuations in metric values across releases, with changes generally occurring gradually over time.

⁵<https://github.com/nopSolutions/nopCommerce>

⁶<https://github.com/xunit/xunit>

⁷<https://github.com/IronLanguages/ironpython3>

The Bumpy Road metric exhibits relatively stable behavior across most projects, with only minor variations between releases. Jellyfin shows a slight decreasing trend, while OpenRA and nopCommerce remain largely stable. IronPython maintains consistently higher values, whereas xUnit shows some fluctuation in earlier releases followed by stabilization. Overall, the metric suggests that localized structural irregularities are either stable or slightly improving over time.

The Functional Parameter Count metric remains relatively stable for most projects, with only minor variations across releases. An exception is xUnit, which exhibits a significant spike in intermediate releases, indicating a temporary increase in methods exceeding the parameter threshold. Apart from this outlier, the metric suggests consistent method-level design practices.

For cohesion, LCOM4 shows a generally decreasing or stable trend across all projects, indicating gradual improvements or controlled cohesion as the systems evolve. In contrast, LCOM5 exhibits higher values and greater variability, particularly for IronPython and xUnit, and does not provide a clear or consistent trend across projects. This further supports the observation that LCOM5 is less suitable for threshold-based evaluation.

Class Coupling shows relatively stable or slightly decreasing trends for most projects, suggesting that dependencies between classes are either maintained at a consistent level or gradually improved. xUnit again stands out with significantly higher values and noticeable fluctuations, reflecting its more complex structure.

For the Maintainability Index, IronPython exhibits consistently high values with a slight decreasing tendency, while Jellyfin remains relatively stable. In contrast, OpenRA and nopCommerce show a steady increase in warning counts across releases, and xUnit demonstrates a more pronounced upward trend with minor fluctuations. These results suggest that, for some projects, system growth is accompanied by an increasing number of maintainability-related issues.

In summary, the analyzed open-source projects exhibit largely stable or slightly improving trends for most structural metrics, with some project-specific deviations. The presence of occasional spikes (notably in xUnit) suggests localized increases in complexity, but overall the results indicate controlled evolution and generally consistent code quality across releases.

6. Comparing student and open-source projects

6.1. Descriptive statistics

Table 2 summarizes the distribution of normalized warning counts for student and open-source projects. It should be noted that the two datasets were evaluated using different threshold sets: thresholds calibrated on the student dataset were applied to student projects, whereas literature-recommended thresholds were used for open-source projects. Consequently, the reported warning counts should be interpreted primarily as descriptive indicators, and direct comparisons of absolute values should be made with caution.

Table 2. Descriptive statistics of normalized warning counts (per 1000 LOC). Values are reported as Student/Open-source.

Metric	Mean	Median	Std. Dev.
Bumpy Road	1.33 / 0.88	1.25 / 0.38	0.75 / 0.86
Functional Parameter Count	3.22 / 5.13	2.29 / 1.14	3.04 / 8.51
LCOM5	4.95 / 5.88	4.32 / 5.60	2.15 / 3.89
LCOM4	2.10 / 4.99	1.89 / 4.97	1.10 / 2.84
Cyclomatic Complexity	4.92 / 3.95	4.90 / 2.09	1.95 / 3.06
Class Coupling	2.71 / 11.96	2.14 / 7.75	1.95 / 10.97

The larger standard deviations observed for several metrics indicate greater architectural diversity among open-source systems. Moreover, the larger differences between mean and median values for metrics such as Functional Parameter Count and Class Coupling suggest skewed distributions and the presence of outliers. In contrast, student projects exhibit smaller mean–median differences, indicating more homogeneous quality characteristics. Finally, LCOM5 yields consistently high values with relatively small mean–median differences in both datasets, supporting previous observations regarding its limited discriminative power.

6.2. Trend analysis

Despite the differences in absolute warning counts, the evolution of metric values exhibits clear differences between the two datasets. Open-source projects generally show moderate variation followed by stabilization or slight improvements, suggesting controlled evolution and ongoing refactoring. In contrast, student projects exhibit a more pronounced increase in normalized warning counts, indicating that structural issues tend to accumulate as functionality is added, with limited evidence of systematic refactoring.

For cohesion- and coupling-related metrics, open-source systems generally exhibit more stable trends, whereas student projects display a clearer increasing tendency. Cyclomatic Complexity and Functional Parameter Count show moderate fluctuations and occasional spikes in open-source projects, while student projects tend to increase more consistently. For Maintainability Index warnings, both datasets exhibit mixed behavior, indicating that maintainability-related issues are present in both cases.

Overall, the results suggest that mature open-source systems are characterized by greater architectural diversity and more controlled evolution over time. Although the use of different threshold sets limits the direct comparison of absolute warning counts, the trend analysis indicates that student projects exhibit a stronger tendency to accumulate complexity and structural issues during development.

7. Conclusion

This research aimed to identify code quality issues related to complexity, cohesion, and maintainability using static code analysis metrics. The developed analyzers were evaluated on student projects from the *Software Technology* course and on selected open-source repositories. Threshold values were determined using the elbow method, and the work extends existing Roslyn-based metrics with additional custom analyses.

The results show that student projects tend to exhibit increasing structural complexity over time, while open-source systems demonstrate more stable evolution, often supported by refactoring practices. Cohesion- and coupling-related metrics (e.g., LCOM4 and Class Coupling) highlight this difference most clearly. In contrast, maintainability-related warnings show mixed trends in both groups and do not provide a clear distinction.

Several challenges were identified, including inconsistencies in milestone progression and the context-dependent nature of metric thresholds. Additionally, the selection of suitable open-source projects was constrained by build and dependency issues.

Despite these limitations, the proposed approach proved effective in identifying problematic code segments and tracking quality trends throughout development. The analyzers can be integrated into continuous integration environments, enabling early feedback and supporting improved coding practices.

Overall, the results confirm that carefully calibrated code metrics provide a practical and scalable means for assessing and improving software quality in both educational and real-world development contexts.

8. Future work

For future work, the following aspects have been identified for further development:

- Implement additional metrics to cover a wider range of code quality attributes, for example cohesion metrics, such as Relational Cohesion.
- Analyze a broader set of student and open-source projects to further evaluate and validate the implemented metrics.
- Perform manual verification to assess whether the code segments flagged by the analyzers indeed contain problematic code, thereby validating the diagnostic accuracy.
- Provide module and submodule level metrics evaluation inside projects, to identify potentially problematic segments in a generally acceptable codebase.

Computer Code Availability

The developed source code metrics and the multitemporal analyzer tool implemented as part of this research is released under the BSD-3 license at: <https://github.com/petercsaszar/code-metrics/>

References

- [1] E. ARISHOLM, L. BRIAND, A. FØYEN: *Dynamic Coupling Measurement for Object-Oriented Software*, Software Engineering, IEEE Transactions on 30 (Sept. 2004), pp. 491–506, DOI: [10.1109/TSE.2004.41](https://doi.org/10.1109/TSE.2004.41).
- [2] P. BHOLOWALIA, A. KUMAR: *EBK-means: A clustering technique based on elbow method and k-means in WSN*, International Journal of Computer Applications 105.9 (2014).
- [3] S. BREESE, A. MILANOVA, B. CUTLER: *Using Static Analysis for Automated Assignment Grading in Introductory Programming Classes (Abstract Only)*, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17, Seattle, Washington, USA: Association for Computing Machinery, 2017, p. 704, ISBN: 9781450346986, DOI: [10.1145/3017680.3022440](https://doi.org/10.1145/3017680.3022440), URL: <https://doi.org/10.1145/3017680.3022440>.
- [4] L. BRIAND, J. DALY, J. WUST: *A unified framework for cohesion measurement in object-oriented systems*, in: Proceedings Fourth International Software Metrics Symposium, 1997, pp. 43–53, DOI: [10.1109/METRIC.1997.637164](https://doi.org/10.1109/METRIC.1997.637164).
- [5] CODESCENE: *The Bumpy Road Code Smell: Measuring Code Complexity by its Shape and Distribution*, 2025, URL: <https://codescene.com/engineering-blog/bumpy-road-code-complexity-in-context/>.
- [6] CPPDEPEND: *Understanding Code Metrics in CppDepend*, 2025, URL: <https://www.cppdepend.com/documentation/code-metrics>.
- [7] T. FILÓ, M. BIGONHA, K. FERREIRA: *Evaluating Thresholds for Object-Oriented Software Metrics*, Journal of the Brazilian Computer Society 30 (Sept. 2024), pp. 313–346, DOI: [10.5753/jbcs.2024.3373](https://doi.org/10.5753/jbcs.2024.3373).
- [8] GITLAB: *REST API*, 2025, URL: <https://docs.gitlab.com/api/rest/>.
- [9] I. V. GOMES, P. MORGADO, T. GOMES, R. M. L. M. MOREIRA: *An overview on the Static Code Analysis approach in Software Development*, in: 2009, URL: <https://api.semanticscholar.org/CorpusID:9373127>.
- [10] I. HEITLAGER, T. KUIPERS, J. VISSER: *A Practical Model for Measuring Maintainability*, in: 6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007), 2007, pp. 30–39, DOI: [10.1109/QUATIC.2007.8](https://doi.org/10.1109/QUATIC.2007.8).
- [11] B. HENDERSON-SELLERS: *Object-oriented metrics: measures of complexity*, USA: Prentice-Hall, Inc., 1995, ISBN: 0132398729.
- [12] T. HERIČKO, B. ŠUMAK: *Exploring Maintainability Index Variants for Software Maintainability Measurement in Object-Oriented Systems*, Applied Sciences 13.5 (2023), ISSN: 2076-3417, DOI: [10.3390/app13052972](https://doi.org/10.3390/app13052972), URL: <https://www.mdpi.com/2076-3417/13/5/2972>.
- [13] M. HITZ, B. MONTAZERI: *Measuring coupling and cohesion in object-oriented systems*, na, 1995.
- [14] P. KASZAB: *Automated evaluation of programming assignments with static code analysis*, MA thesis, Eötvös Loránd University, 2023, URL: https://tms-elte.gitlab.io/theses/kaszab_peter_tdk.pdf.

- [15] P. KASZAB, M. CSERÉP: *Detecting Programming Flaws in Student Submissions with Static Source Code Analysis*, Studia Universitatis Babeş-Bolyai Informatica 68 (July 2023), pp. 37–54, DOI: [10.24193/subbi.2023.1.03](https://doi.org/10.24193/subbi.2023.1.03).
- [16] E. N. H. KIRÇİL, T. E. AYYILDIZ: *Analysis of Lack of Cohesion in Methods (LCOM): A Case Study*, in: 2021 2nd International Informatics and Software Engineering Conference (IISEC), 2021, pp. 1–4, DOI: [10.1109/IISEC54230.2021.9672419](https://doi.org/10.1109/IISEC54230.2021.9672419).
- [17] R. C. MARTIN: *Clean code: a handbook of agile software craftsmanship*, Pearson Education, 2009.
- [18] R. C. MARTIN, OCTOBER: *OO Design Quality Metrics*, in: 1997, URL: <https://api.semanticscholar.org/CorpusID:18246616>.
- [19] T. MCCABE: *A Complexity Measure*, IEEE Transactions on Software Engineering SE-2.4 (1976), pp. 308–320, DOI: [10.1109/TSE.1976.233837](https://doi.org/10.1109/TSE.1976.233837).
- [20] MICROSOFT: *Code metrics - Maintainability index range and meaning*, 2026, URL: <https://learn.microsoft.com/en-us/visualstudio/code-quality/code-metrics-maintainability-index-range-and-meaning>.
- [21] MICROSOFT: *The .NET Compiler Platform SDK*, 2024, URL: <https://learn.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/>.
- [22] A. MISHRA, R. SHATNAWI, C. CATAL, A. AKBULUT: *Techniques for Calculating Software Product Metrics Threshold Values: A Systematic Mapping Study*, Applied Sciences 11.23 (2021), ISSN: 2076-3417, DOI: [10.3390/app112311377](https://doi.org/10.3390/app112311377), URL: <https://www.mdpi.com/2076-3417/11/23/11377>.
- [23] NDEPEND: *100+ .NET and C# Code Metrics*, 2025, URL: <https://www.ndepend.com/docs/code-metrics>.
- [24] A. S. NUÑEZ-VARELA, H. G. PÉREZ-GONZALEZ, F. E. MARTÍNEZ-PEREZ, C. SOUBERVIELLE-MONTALVO: *Source code metrics: A systematic mapping study*, Journal of Systems and Software 128 (2017), pp. 164–197, ISSN: 0164-1212, DOI: <https://doi.org/10.1016/j.jss.2017.03.044>, URL: <https://www.sciencedirect.com/science/article/pii/S0164121217300663>.
- [25] B. D. OROSZ: *Automated validation of design and architectural patterns on student assignments with static code analysis*, MA thesis, Eötvös Loránd University, 2024, URL: https://tms-elte.gitlab.io/theses/orosz_balint_dominik_tdk.pdf.
- [26] R. RAJAPAKSE, C. SZABO: *Code Refactoring Strategies of Third Year Software Engineering Students*, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2024, Milan, Italy: Association for Computing Machinery, 2024, pp. 562–568, ISBN: 9798400706004, DOI: [10.1145/3649217.3653560](https://doi.org/10.1145/3649217.3653560), URL: <https://doi.org/10.1145/3649217.3653560>.
- [27] D. A. REBRO, S. CHREN, B. ROSSI: *Source Code Metrics for Software Defects Prediction*, in: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, SAC '23, Tallinn, Estonia: Association for Computing Machinery, 2023, pp. 1469–1472, ISBN: 9781450395175, DOI: [10.1145/3555776.3577809](https://doi.org/10.1145/3555776.3577809), URL: <https://doi.org/10.1145/3555776.3577809>.
- [28] L. H. ROSENBERG, T. HAMMER, J. G. SHAW: *SOFTWARE METRICS AND RELIABILITY*, in: 1998, URL: <https://api.semanticscholar.org/CorpusID:59798472>.
- [29] D. SANTOS, A. RESENDE, E. DE CASTRO LIMA, A. FREIRE: *Software Instability Analysis Based on Afferent and Efferent Coupling Measures*, Journal of Software 12 (Jan. 2017), pp. 19–34, DOI: [10.17706/jsw.12.1.19-34](https://doi.org/10.17706/jsw.12.1.19-34).
- [30] V. SATOPAA, J. ALBRECHT, D. IRWIN, B. RAGHAVAN: *Finding a Needle in a Haystack: Detecting Knee Points in System Behavior*, in: July 2011, pp. 166–171, DOI: [10.1109/ICDCSW.2011.20](https://doi.org/10.1109/ICDCSW.2011.20).
- [31] R. L. THORNDIKE: *Who Belongs in the Family?*, Psychometrika 18.4 (1953), pp. 267–276, DOI: [10.1007/BF02289263](https://doi.org/10.1007/BF02289263).



Static Analysis Possibilities for Regular Expressions in C++

Tibor Teodor Fűrész, Kristóf Umann, Zoltán Porkoláb

ELTE Eötvös Lóránd University, Faculty of Informatics, Department of Programming
Languages and Compilers, Budapest, Hungary
jaoycb@inf.elte.hu, szelethus@inf.elte.hu, gsd@inf.elte.hu

Abstract. Most of the regular expression libraries in current programming languages utilize a dynamic constructor parameter to construct a state machine, which will be used later in the matching. When this constructor parameter does not fulfill the requirements of the correct regular expression syntax, a runtime error – usually an exception – occurs. It would be highly beneficial if we could catch this kind of problem at compile time with the help of static analysis. The prerequisite of such an analysis is that the constructor parameter is fully known at compile time. In this research, we investigate the extent to which this prerequisite holds true for open-source software projects. We measured 26 different popular open-source projects, and we found that in more than two-thirds of the time, constructors used hardcoded parameters. This result opens the possibility for the creation of an effective static analysis tool.

Keywords: static analysis, C, C++, Clang, LLVM, regular expressions

1. Introduction

Regular expressions are an everyday part of software programming. Common scenarios like input parsing, file lookup, and search and replace are just a few examples, and in truth, it is hard to fathom a program with any meaningful complexity that can omit them. Most tools, like `grep`, that build regular expressions also come with robust techniques to check for their correctness.

Most of the current libraries work on the idea that regular expressions are implemented using state machines. The state machine is usually built up in the constructor, based on a regular expression pattern received as the construction parameter.

While this is a time-consuming process, later repeated use of the state machine is relatively cheap. However, the constructor parameter should be matched for the correct syntax of the regular expression. This error checking happens dynamically inside the constructor during the construction of the state machine.

During development, regular expressions are notoriously hard to get right. Their syntax is unintuitive, hard to read, and more importantly, to write [28]. Testing the expressions is also a challenge, as test effectiveness heavily depends on the test coverage. It is easy to see that test frameworks usually miss the same corner cases, which programmers also likely forget to implement correctly.

The trade-offs between dynamic and static analysis have been widely studied and publicized [4, 18, 21, 22, 30]. Briefly, most forms of dynamic analysis require the execution of the program and also precise input parameters. On the other hand, static analyzers usually do not require the concrete execution of the program or any input parameters, offering a unique advantage for detecting programming errors in the early stages of development [3]. As bugs that are caught early are faster, easier, and cheaper to fix [7], there are clear advantages for pursuing static analysis whenever possible.

This paper is organized as follows. We overview the related literature in Section 2. The technical background of our contribution is discussed in Section 3. Our measurement methodology will be introduced in Section 4. We evaluate our results in Section 5. Future research directions are discussed in Section 6. Finally, our paper concludes in Section 7.

2. Related work

Our paper investigates how C++ programmers use regular expressions, especially whether constructors use hard-wired string literals or constants. We found similar studies to ours aiming at how developers use certain C++ features. The authors in [19] survey the usage of immutability declarations. In a broader study, the authors in [48] surveyed how C++ libraries are used.

In their paper, the authors of [15] surveyed library usage of Fedora Linux 37, which spans just shy of 400 million lines of code. Among other interesting findings, they identified Boost as the most common third party library, and that the most frequently used assets from Boost have no, or only partial counterparts in the C++ standard library. They found that there is a “*worrying trend*” that programmers do not use bounds checking alternatives to access containers or views.

Regular expression implementations that have a super-linear worst-case complexity are vulnerability that can be exploited for denial of service attacks [13].

There are several varieties and extensions to regular expressions implemented in programming languages. In 2019, a study [14] discussed whether regular expressions written in one language can be ported as-is from one language to another. Across 193,524 projects and 537,806 regexes they found if ported, 15% posed a risk logic errors due to semantic differences, 10% might have performance differences to the extent where they posed a security risk.

Vulnerabilities connected to regular expressions and their usage are long time in the focus of research. One critical problem regarding regular expressions is the quality of their tests. Regular expressions are often less thoroughly tested than other parts of the codebase, according to a 2016 survey [8] with python developers. In 2018, a paper [46] measured that only 17% of regexes are tested in the 1225 Java projects they inspected, and even among those, their coverage (positive and negative test strings) were poor. The *AREXT* [44] tool is developed to automatically find, and generate test strings against regular expressions for full coverage, inspired by the former two papers.

The authors of paper [25] describe how “evil” test strings can be generated for testing regular expressions.

The *Boost.Xpressive* library provides functions to search strings using regular expressions [32]. Boost.Xpressive makes it possible to write down regular expressions as C++ code rather than strings. That makes it possible to check at compile time whether a regular expression is valid or not. However, Boost.Xpressive heavily utilizes C++ template metaprogramming [1, 45], which makes its syntax very different from other regular expression libraries.

While not specific to regular expressions, prior work highlights broader efforts to improve reasoning about program properties. For example, ARC (Automated Reasoning in the Class) proposes the deployment of automated reasoning tools in education, and surveys of specification languages categorize how properties can be expressed and checked (statically or dynamically), providing context for specialized analysis tools, such as ours [16, 24].

3. Technical overview

In the current C++ development, the landscape of regular expressions is fairly fragmented. Currently, there are three major libraries providing regex capabilities. These are the standard library (`std::regex`), Boost.Regex and Google’s RE2. Each of them uses a different grammar by default. The standard library uses the ECMAScript 262 standard, which is an older version of the standard used in JavaScript. The Boost.Regex library defaults to the PERL regex grammar, which is the most feature-rich of the three. Finally, Google’s RE2 defines its own grammar. The RE2 regex library promises linear matching time for every pattern. To achieve this developers had to strip functionality present in most other regex libraries, such as backreferences. It further complicates things that most libraries support multiple standards. For instance, the standard and boost regex both support the POSIX Basic Regular Expression (BRE) grammar, which is the default in `grep`, as well as the POSIX Extended Regular Expression (ERE) grammar, used in `awk`.

In the midst of these mountains of standards, developers are highly susceptible to making mistakes when writing their patterns. In Figure 1 we show a few possible errors regarding the standard `std::regex` class. One frequent mistake is trying to use a construct that is not supported by the currently selected standard. Other common syntax errors can originate from structural mistakes within the pattern.

These include not closing capture groups or exact quantifiers and improperly adding quantifiers after non-quantifiable tokens. Another hard-to-detect mistake is invalid escape sequences. Such a mistake can easily be made if the developer forgets to use a raw string for their patterns.

```
std::regex re("(?<=CodeChecker )clang-tidy"); // Using construct not
        supported by the standard
std::regex re("abc"); // unclosed capture group
std::regex re("a{2}"); // unclosed exact quantifier
std::regex re("a?*"); // quantifier after a non-quantifiable token
std::regex re("\d+"); // invalid escape sequence, matches "d", not
        digits
```

Figure 1. Example of possible syntax errors in regex constructors.

4. Measurement methodology

To conduct our measurements on C++ projects, we developed a custom static analysis checker built on the LLVM/Clang infrastructure. Specifically, we extended clang-tidy 22.0.0, a widely adopted tool, by implementing a custom checker to identify the instantiation of regex objects. The primary goal of our addition consists of two parts. First is to measure a baseline of the usage of regular expressions, and second is to determine in each case whether the pattern is known at compile time or not.

To achieve our first goal, we made our check emit a warning for each constructor call, regardless of what parameters it received. This unconditional matching allows us to safely count the number of times a regex is used in a project.

To complete our secondary goal, we perform a deeper analysis of the parameters received by the constructor. We try to classify each parameter into one of three categories. The first category is hardcoded patterns. These are patterns that are known at compile time. We regard each parameter as such if they are a string literal, or a constant string or constant character pointer defined with a single string literal. Our second category consists of patterns, which we determined to be constructed during runtime. We regard a parameter to be such a dynamic pattern if there is a concatenation operator in either the parameter itself or in the definition of the parameter variable. The final category is the unknown patterns, consisting of parameters that do not fall into either of the previous categories.

To account for a wide variety of programming standards and conventions, we took measurements in a wide selection of 26 open-source projects. These projects were sourced from GitHub with the only criteria being that they have at least one regex constructor in their codebase.

5. Results and evaluation

The summary of our findings is presented in Table 1. We can see a dominant usage of hardcoded regex patterns over dynamically constructed ones among the tested projects. Evaluating the results on a per-project basis, as shown in Table 2, revealed that in a significant portion of analyzed projects, every single regex pattern is hardcoded.

The results also show that only in a small number of cases can we determine that the pattern is constructed during runtime, meaning that with improvements made to our static analysis tool, we might be able to find even more hardcoded patterns. These results support our theory that there is a possibility to create a static analysis tool targeting regexes in C++.

Table 1. Overall number of regex constructors found in open-source projects.

Constructors	462	–
Constructors with hardcoded strings	322	69.70%
Constructors with dynamic strings	13	2.81%
Constructors unknown patterns	127	27.49%

A detailed investigation can reveal certain static and dynamic patterns from different libraries. A typical hardcoded regex constructor parameter can be seen in Figure 2. Bear uses it to escape all characters not defined in its group and wrap new lines into strings. An example of a dynamic pattern can be seen in Figure 3. SQLCheck uses a dynamic pattern to search for recursive references.

```
std::string escape(const std::string& input) {
    if (input.empty()) {
        return "''";
    }

    const std::regex ESCAPE_PATTERN(R"#[^\A-Za-z0-9_\-.,:/@\n])#");
    const std::regex LINE_FEED(R"#[\n]#");

    const auto output =
        std::regex_replace(input, ESCAPE_PATTERN, "\\$1");
    return std::regex_replace(output, LINE_FEED, "\n");
}
```

Figure 2. Hardcoded patterns in Bear.

Table 2. Distribution of regular expression constructors and the type of their patterns.

Project	Total Regexes	hardcoded	Dynamic	Unknown
Bear [5]	10	10	0	0
Crow [12]	2	1	0	1
bitcoin [38]	2	1	0	1
blender [6]	23	20	2	1
ceph [39]	62	42	6	14
colmap [10]	6	6	0	0
contour [11]	6	6	0	0
dragon [17]	30	14	1	15
folly [27]	14	8	0	6
json [23]	1	1	0	0
libdatachannel [26]	2	2	0	0
llvm-project [40]	3	3	0	0
mold [29]	14	14	0	0
nanogui [31]	2	0	0	2
obs-studio [33]	4	2	0	2
pistache [34]	12	11	0	1
proxysql [35]	58	13	0	45
qt-6.10.1 [41]	33	27	0	6
retdec [2]	34	32	1	1
seastar [36]	12	4	0	8
shaderc [20]	24	11	2	11
sqlcheck [37]	35	32	1	2
threepp [43]	25	24	0	1
whisper.cpp [47]	17	11	0	6
xournalpp [49]	20	20	0	0
zeek [42]	11	7	0	4

6. Future work

In Figure 4 we can see a dynamic constructor parameter. This parameter is constructed with the concatenation of a number of namespace variables, therefore, we currently do not consider it as hardcoded. Although one can recognize that the variables themselves are hardcoded, and moreover, they are constant. An advanced data flow analysis could detect this fact and consider the constructor parameter as a hardcoded one. However, we then still have to model the string concatenation to obtain the final regex pattern and execute an appropriate syntactic check on it. This is one of our future research directions.

```

void CheckRecursiveDependency(Configuration& state,
                             const std::string& sql_statement,
                             bool& print_statement){

    std::string table_name = GetTableName(sql_statement);
    if(table_name.empty()){
        return;
    }

    std::regex pattern("("references\\s+" + table_name+ "");
    std::string title = "Recursive Dependency";
    PatternType pattern_type = PatternType::
        PATTERN_TYPE_LOGICAL_DATABASE_DESIGN;

    auto message = "...";
    CheckPattern(...);
}

```

Figure 3. Dynamic patterns in SQLCheck.

```

...
const std::string schema_re = "([[:alpha:]]+:\\/\\/\\/)";
const std::string user_pass_re = "(([^:\\s]+):([^@\\s]+)@)?";
const std::string host_port_re = "([[:alnum:]]\\.\\.-]+)";
const std::string path_re = "(\\/([[:print:]]*))?";
...

bool parse_url_authority(const std::string& url, std::string& host,
                        std::string& user, std::string& password) {
    const std::string re = schema_re + user_pass_re + host_port_re +
        path_re;
    const std::regex url_regex(re, std::regex::icase);
    std::smatch url_match_result;

    if (std::regex_match(url, url_match_result, url_regex)) {
        host = url_match_result[HOST_GROUP_IDX];
        user = url_match_result[USER_GROUP_IDX];
        password = url_match_result[PASSWORD_GROUP_IDX];
        return true;
    }

    return false;
}

```

Figure 4. Quasi-dynamic patterns in Ceph.

7. Conclusion

This paper discusses the possibilities of static analysis of regular expressions with a particular focus on C/C++. Despite the age and the wealth of knowledge accumulated in the field of regular expressions, as of writing, we are not aware of any works that statically check the correctness of types like `std::regex` or `boost::regex`.

However, static analysis could be applied for regular expressions only if the expression is available at compile-time. Before any effort is made in this direction we have to understand how programmers use regular expressions, especially at what rate they apply patterns known at compilation time.

We applied the clang-tidy [9] static analysis tool to detect regular expression objects in large open-source C++ projects and to determine the ratio in which those objects were initialized with compile-time constant strings. We found that in typical C++ projects, most of the regular expression objects are initialized based on hardcoded string patterns, therefore static analysis of them is viable.

The paper describes the method, the algorithm, a prototype implementation, and the measurement results on large open-source projects. Our results proved that 69.7% of the regular expression objects are initialized with hardcoded string patterns, as seen in Table 1. Thus, this research could be the groundwork for the development of such a static analyzer.

Acknowledgment

Project no. C2314106 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the KDP-2023 funding scheme.



References

- [1] D. ABRAHAM, A. GURTOVOY: *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond*, C++ in Depth Series, Addison-Wesley Professional, 2004, ISBN: 0321227255.
- [2] AVAST SOFTWARE: *RetDec: A Retargetable Machine-Code Decompiler*, Source code repository, 2026, URL: <https://github.com/avast/retdec>.
- [3] N. AYEWAH, W. PUGH, D. HOVEMEYER, J. D. MORGENTHALER, J. PENIX: *Using Static Analysis to Find Bugs*, IEEE Software 25.5 (Sept. 2008), pp. 22–29, DOI: [10.1109/ms.2008.130](https://doi.org/10.1109/ms.2008.130), URL: <https://doi.org/10.1109/2Fms.2008.130>.
- [4] Á. BALOGH, R. SZALAY: *On the Applicability of Static Analysis for System Software using CodeChecker*, in: 2024 7th International Conference on Software and System Engineering (ICoSSE), 2024, pp. 15–22, DOI: [10.1109/ICoSSE62619.2024.00011](https://doi.org/10.1109/ICoSSE62619.2024.00011).
- [5] BEAR CONTRIBUTORS: *Bear: Build EAR*, Source code repository, 2026, URL: <https://github.com/rizotto/Bear>.

- [6] BLENDER FOUNDATION: *Blender*, Source code repository, 2026, URL: <https://github.com/blender/blender>.
- [7] B. BOEHM, V. R. BASILI: *Software Defect Reduction Top 10 List*, Computer 34.1 (Jan. 2001), pp. 135–137, ISSN: 0018-9162, DOI: [10.1109/2.962984](https://doi.org/10.1109/2.962984), URL: <http://dx.doi.org/10.1109/2.962984>.
- [8] C. CHAPMAN, K. T. STOLEE: *Exploring regular expression usage and context in Python*, in: Proceedings of the 25th International Symposium on Software Testing and Analysis, ISSSTA '16, ACM, July 2016, pp. 282–293, DOI: [10.1145/2931037.2931073](https://doi.org/10.1145/2931037.2931073), URL: <http://dx.doi.org/10.1145/2931037.2931073>.
- [9] CLANG TIDY: *Clang-Tidy*, <https://clang.llvm.org/extra/clang-tidy/> (last accessed: 28-02-2019), 2019.
- [10] COLMAP CONTRIBUTORS: *COLMAP: A General-Purpose Structure-from-Motion and Multi-View Stereo Pipeline*, Source code repository, 2026, URL: <https://github.com/colmap/colmap>.
- [11] CONTOUR CONTRIBUTORS: *Contour: Modern C++ Terminal Emulator*, Source code repository, 2026, URL: <https://github.com/contour-terminal/contour>.
- [12] CROWCPP CONTRIBUTORS: *Crow: A Fast C++ Microframework for the Web*, Source code repository, 2026, URL: <https://github.com/CrowCpp/Crow>.
- [13] J. C. DAVIS, C. A. COGHLAN, F. SERVANT, D. LEE: *The impact of regular expression denial of service (ReDoS) in practice: an empirical study at the ecosystem scale*, in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '18, ACM, Oct. 2018, pp. 246–256, DOI: [10.1145/3236024.3236027](https://doi.org/10.1145/3236024.3236027), URL: <http://dx.doi.org/10.1145/3236024.3236027>.
- [14] J. C. DAVIS, L. G. MICHAEL IV, C. A. COGHLAN, F. SERVANT, D. LEE: *Why aren't regular expressions a lingua franca? an empirical study on the re-use and portability of regular expressions*, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '19, ACM, Aug. 2019, pp. 443–454, DOI: [10.1145/3338906.3338909](https://doi.org/10.1145/3338906.3338909), URL: <http://dx.doi.org/10.1145/3338906.3338909>.
- [15] J. DENG, M. D. ADAMS: *An Analysis of Library Usage in the C++ Code Base of Fedora Linux 37*, in: 2024 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), IEEE, Aug. 2024, pp. 1–6, DOI: [10.1109/pacrim61180.2024.10690215](https://doi.org/10.1109/pacrim61180.2024.10690215), URL: <http://dx.doi.org/10.1109/pacrim61180.2024.10690215>.
- [16] I. DRAMNESC, T. JEBELEAN, E. ABRAHAM, G. KUSPER, S. STRATULAT: *ARC: An Educational Project on Automated Reasoning in the Class*, in: Proceedings of EdMedia 2022, ed. by T. BASTIAENS, Online: Association for the Advancement of Computing in Education (AACE), Nov. 2022, pp. 934–943, URL: <https://www.learntechlib.org/p/221395>.
- [17] DROGON FRAMEWORK CONTRIBUTORS: *Drogon: A C++14/17/20 based HTTP Web Framework*, Source code repository, 2026, URL: <https://github.com/drogonframework/drogon>.
- [18] M. D. ERNST: *Static and dynamic analysis: Synergy and duality*, in: WODA 2003: ICSE Workshop on Dynamic Analysis, 2003, pp. 24–27, DOI: [10.1109/icse.2003.1201290](https://doi.org/10.1109/icse.2003.1201290).
- [19] J. EYOLFSON, P. LAM: *How C++ Developers Use Immutability Declarations: An Empirical Study*, in: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019, pp. 362–372, DOI: [10.1109/ICSE.2019.00050](https://doi.org/10.1109/ICSE.2019.00050).
- [20] GOOGLE LLC: *Shaderc: A collection of tools, libraries, and tests for Vulkan shader compilation*, Source code repository, 2026, URL: <https://github.com/google/shaderc>.
- [21] S. HALLEM, B. CHELF, Y. XIE, D. ENGLER: *A System and Language for Building System-specific, Static Analyses*, PLDI '02 (2002), pp. 69–82, DOI: [10.1145/512529.512539](https://doi.org/10.1145/512529.512539), URL: <http://doi.acm.org/10.1145/512529.512539>.

- [22] S. HECKMAN, L. WILLIAMS: *A systematic literature review of actionable alert identification techniques for automated static code analysis*, Information and Software Technology 53.4 (2011), Special section: Software Engineering track of the 24th Annual Symposium on Applied Computing, pp. 363–387, ISSN: 0950-5849, DOI: <https://doi.org/10.1016/j.infsof.2010.12.007>, URL: <https://www.sciencedirect.com/science/article/pii/S0950584910002235>.
- [23] JSON CONTRIBUTORS: *JSON for Modern C++*, Source code repository, 2026, URL: <https://github.com/nlohmann/json>.
- [24] G. KUSPER, G. KOVÁSZNAI, W. SCHREINER, G. GUTA, J. SZTRIK: *A Small Survey of Java Specification Languages*, in: Jan. 2010.
- [25] E. LARSON, A. KIRK: *Generating Evil Test Strings for Regular Expressions*, in: 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), IEEE, Apr. 2016, pp. 309–319, DOI: [10.1109/icst.2016.29](https://doi.org/10.1109/icst.2016.29), URL: <http://dx.doi.org/10.1109/icst.2016.29>.
- [26] LIBDATACHANNEL CONTRIBUTORS: *libdatachannel: C/C++ WebRTC network library*, Source code repository, 2026, URL: <https://github.com/paullouisageneau/libdatachannel>.
- [27] META PLATFORMS, INC.: *Folly: Facebook Open-source Library*, Source code repository, 2026, URL: <https://github.com/facebook/folly>.
- [28] L. G. MICHAEL, J. DONOHUE, J. C. DAVIS, D. LEE, F. SERVANT: *Regexes are Hard: Decision-Making, Difficulties, and Risks in Programming Regular Expressions*, in: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2019, pp. 415–426, DOI: [10.1109/ASE.2019.00047](https://doi.org/10.1109/ASE.2019.00047).
- [29] MOLD CONTRIBUTORS: *Mold: A Modern Linker*, Source code repository, 2026, URL: <https://github.com/rui314/mold>.
- [30] N. NAGAPPAN, T. BALL: *Static Analysis Tools As Early Indicators of Pre-release Defect Density*, ICSE '05 (2005), pp. 580–586, DOI: [10.1145/1062455.1062558](https://doi.org/10.1145/1062455.1062558), URL: <http://doi.acm.org/10.1145/1062455.1062558>.
- [31] NANOGUI CONTRIBUTORS: *NanoGUI: Minimalist GUI library for OpenGL*, Source code repository, 2026, URL: <https://github.com/wjakob/nanogui>.
- [32] E. NIEBLER: *Boost.Xpressive: An advanced regular expression template library for C++*, <https://boost.org>, Boost C++ Libraries, 2007.
- [33] OBS PROJECT CONTRIBUTORS: *OBS Studio*, Source code repository, 2026, URL: <https://github.com/obsproject/obs-studio>.
- [34] PISTACHE.IO: *Pistache: An elegant C++ REST framework*, Source code repository, 2026, URL: <https://github.com/pistacheio/pistache>.
- [35] PROXYSQL CONTRIBUTORS: *ProxySQL: High Performance, High Availability, Protocol Aware Proxy for MySQL*, Source code repository, 2026, URL: <https://github.com/sysown/proxysql>.
- [36] SCYLLADB: *Seastar: High performance server-side application framework*, Source code repository, 2026, URL: <https://github.com/scylladb/seastar>.
- [37] SQLCHECK CONTRIBUTORS: *SQLCheck: Automatically identify anti-patterns in SQL queries*, Source code repository, 2026, URL: <https://github.com/jarulraj/sqlcheck>.
- [38] THE BITCOIN CORE: *Bitcoin Core*, 2022, URL: <https://bitcoincore.org/>.
- [39] THE CEPH FOUNDATION: *Ceph: A distributed object, block, and file storage platform*, Source code repository, 2026, URL: <https://github.com/ceph/ceph>.
- [40] THE LLVM PROJECT: *The LLVM Compiler Infrastructure*, Source code repository, 2026, URL: <https://github.com/llvm/llvm-project>.
- [41] THE QT COMPANY: *Qt 6*, Source Code Archive, 2026, URL: <https://download.qt.io/archive/qt/6.10/6.10.1/single/>.

- [42] THE ZEEK PROJECT: *Zeek Network Security Monitor*, Source code repository, 2026, URL: <https://github.com/zeek/zeek>.
- [43] THREEPP CONTRIBUTORS: *threepp: C++ port of three.js*, Source code repository, 2026, URL: <https://github.com/markaren/threepp>.
- [44] N. VAN HOAN, P. N. HUNG: *Arest: Automatic Regular Expression Testing Tool Based on Generating Strings With Full Coverage*, in: 2021 13th International Conference on Knowledge and Systems Engineering (KSE), IEEE, Nov. 2021, pp. 1–6, DOI: [10.1109/kse53942.2021.9648604](https://doi.org/10.1109/kse53942.2021.9648604), URL: <http://dx.doi.org/10.1109/kse53942.2021.9648604>.
- [45] D. VAN DE VOORDE, N. M. JOSUTTIS, D. GREGOR: *C++ Templates: The Complete Guide (2nd Edition)*, 2nd, Addison-Wesley Professional, Sept. 2017, ISBN: 978-0-321-71412-1, URL: <http://tmplbook.com>.
- [46] P. WANG, K. T. STOLEE: *How well are regular expressions tested in the wild?*, in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '18, ACM, Oct. 2018, pp. 668–678, DOI: [10.1145/3236024.3236072](https://doi.org/10.1145/3236024.3236072), URL: <http://dx.doi.org/10.1145/3236024.3236072>.
- [47] WHISPER.CPP CONTRIBUTORS: *Whisper.cpp: High-performance inference of OpenAI's Whisper automatic speech recognition*, Source code repository, 2026, URL: <https://github.com/gml-org/whisper.cpp>.
- [48] D. WU, L. CHEN, Y. ZHOU, B. XU: *How do developers use C++ libraries? An empirical study*, in: Proceedings of the Twenty-Seventh International Conference on Software Engineering and Knowledge Engineering, 2015, pp. 260–265.
- [49] XOURNAL++ CONTRIBUTORS: *Xournal++: Handwriting notetaking software with PDF annotation*, Source code repository, 2026, URL: <https://github.com/xournalpp/xournalpp>.



LLM-Powered Automated Attacks

Tamás Girászi, Natáli Papp, Norbert Oláh, Andrea Huszti

Faculty of Informatics, University of Debrecen, Debrecen, Hungary

giraszi.tamas@inf.unideb.hu

pappnaty@gmail.com

olah.norbert@inf.unideb.hu

huszti.andrea@inf.unideb.hu

Abstract. The increasing integration of large language models (LLMs) into systems introduces new attack surfaces that extend beyond traditional software vulnerabilities. While LLMs are commonly protected by prompt-level security mechanisms, recent researches show that these controls can be bypassed through carefully crafted inputs. In this paper, we propose an interaction model based on a Generative Adversarial Network (GAN) conceptual analogy and a dual-LLM framework for systematically examining and testing the security boundaries of LLMs through malicious code generation. The framework consists of two LLMs that iteratively produce attack-oriented prompts, interpret and implement them. Experimental results demonstrate that the proposed approach can generate outputs that are similar to real-world attack patterns, such as SQL injection and cross-site scripting. This research highlights the importance of LLM security controls and emphasizes the need for proactive, automated evaluation methods to improve the robustness and governance of LLM-based systems.

Keywords: Large Language Models, Adversarial Prompting, LLMs, GAN

2020 Mathematics Subject Classification: Primary 60F15, 60G42, 60G50; Secondary 60F10

1. Introduction

Over the past decade, advances in artificial intelligence have had a huge impact on the IT domain. The rapid evolution of AI-driven methodologies, particularly machine learning (ML) and deep learning (DL), has significantly influenced not only economic and industrial processes but has also fundamentally reshaped the

cybersecurity landscape, affecting both defensive and adversarial capabilities. AI algorithms enable the efficient analysis of large-scale data and facilitate the identification of malicious patterns and anomalous behaviors with increasing accuracy and speed ([4, 17, 36, 41]).

Traditionally, every IT system is designed to meet fundamental security requirements. However, the growing complexity, operational speed, and adaptive characteristics of modern threats have introduced challenges that exceed the capabilities of classical, mainly reactive defense paradigms.[26] On the other hand, AI-based techniques have begun to appear on the offensive side, enabling adversaries to autonomously discover system vulnerabilities, optimize attack strategies, and generate highly targeted phishing and social engineering attacks ([10, 13, 14, 33]).

The offensive use of artificial intelligence introduces new dimensions to the nature of cyberattacks. Whereas earlier threat models relied mainly on automation and script-based techniques, modern attack frameworks increasingly incorporate self-learning and adaptive mechanisms capable of dynamically bypassing established defensive controls. LLMs, in particular, leverage advanced natural language processing capabilities to perform sophisticated conversational, text generation, and analytical tasks, thereby facilitating the large-scale automation of social engineering and disinformation activities. As a result, LLMs are increasingly employed to generate phishing emails, fraudulent communications, and persuasive textual content designed to manipulate or deceive users ([8, 13, 18, 45]).

The objective of this research is to develop an automated and proactive offensive framework that systematically exposes system vulnerabilities through the generation of adversarial attacks. The proposed approach is based on the interaction of two large language models, implemented as Generative Pre-trained Transformer (GPT) based systems and driven by distinct prompting strategies, within a Generative Adversarial Network (GAN) inspired architectural paradigm. In this configuration, one LLM functions as a generator that iteratively produces adversarial prompts until it induces the discriminator model to generate malicious content. This process enables the systematic evaluation of system-level security constraints. The proposed framework highlights the critical importance of robust AI security mechanisms and provides a foundation for the development of targeted defensive measures and validation protocols. Furthermore, it enables structured testing of LLM security controls, offering insight into the extent to which such models may be sensitive to manipulation or compromise.

2. Related Work

There are numerous types of cyber attacks that, driven by ongoing digitalization, affect multiple technological and organizational domains. The literature identifies several major categories of cyber attacks, the diversity and increasing complexity of these threats and illustrating the challenges faced by organizations and individuals operating in today's digitally interconnected environment ([26, 29, 32, 38]).

Many techniques are employed in various ways to support cyberattacks, including the automation of phishing campaigns and the generation of evasive malware variants, thereby increasing the effectiveness and scalability of adversarial activities ([8, 33]). Through the application of natural language processing and LLMs, adversaries are able to generate more convincing and seemingly legitimate messages, substantially enhancing the success rate of deception-based attacks. Such techniques not only improve the efficiency of attack execution but also considerably complicate their timely detection and mitigation ([13, 14]). LLMs must be equipped with robust defensive mechanisms capable of detecting and mitigating prompt injection attacks and consistently rejecting malicious prompts. However, such defenses are often relatively easy to circumvent, while their systematic evaluation and analysis remain inherently challenging. ([20, 42, 44])

Several recent studies explicitly investigate how large language models can be deliberately coerced into producing malicious or policy-violating outputs through prompt injection and jailbreaking techniques ([34, 42, 44]). Shen et al. [35] analyze thousands of real-world jailbreak prompts collected from online communities, with the explicit goal of identifying prompt patterns that successfully bypass built-in security mechanisms. Their work focuses on empirical characterization of existing attack prompts rather than generating new ones adaptively.

The security of LLMs extends beyond external threats such as prompt manipulation, data leakage, or malicious misuse (e.g., phishing or disinformation), and also encompasses internal risks arising from the autonomous operation of these models ([20, 34, 42]). These risks are especially in the case of autonomous LLM-based agents, where the objectives pursued by the model may diverge from user intent, strategic deceptive behaviors may emerge, or latent goal structures may persist despite extensive security and alignment training ([15]). Among these, it includes not only goal misalignment but also the potential emergence of hidden objectives, strategic deception (scheming), self-preserving behaviors, and the persistence of these undesirable properties even under current security training paradigms ([15, 28, 35]).

Yu et al. [43] propose GPTFUZZER, a black-box red-teaming framework that adapts the concept of software fuzzing to the security analysis of LLMs through the use of an initial jailbreak template corpus and evolutionary mutation operators. While their approach demonstrates strong capability in generating diverse and robust adversarial prompts via automated template exploration, its effectiveness heavily depends on the quality of the initial seeds and places less emphasis on fine-grained, context-specific semantic interaction between the attacker and the target model.

Mehrotra et al. [25] propose the TAP (Tree of Attacks with Pruning) method, an automated framework that combines tree-based search with output-driven pruning to systematically jailbreak black-box LLMs. While their approach demonstrates that structured reduction of the search space can significantly improve attack efficiency and reduce query costs, its effectiveness critically depends on the evaluator model's ability to accurately predict the target model's safety responses to

partially refined prompts. Additionally, the approach may exhibit limited generalization across models with different alignment strategies, incur non-trivial computational overhead due to tree expansion, and risk converging to locally optimal attack strategies, thereby overlooking more diverse jailbreak trajectories.

Chao et al. [5] propose the PAIR (Prompt Automatic Iterative Refinement) method, an automated framework that iteratively refines semantically grounded adversarial prompts against black-box LLMs through an “attacker–judge–target” loop. While their approach demonstrates the remarkable effectiveness of social engineering–based jailbreaking with a minimal number of queries, it requires the use of an additional LLM as part of the attack pipeline, which introduces extra computational cost and potential monetary overhead.

3. Our Contribution

This work presents a comprehensive, proactive-offensive, LLM-assisted security evaluation framework for the systematic analysis of prompt-level vulnerabilities in large language models. Its primary contributions are the design, implementation, and empirical evaluation of a novel GAN-inspired dual-LLM adversarial prompting architecture that models prompt manipulation as an iterative, self-supervised process, enabling continuous exploration of prompt-based attacks without any parameter modification or fine-tuning. Instead of optimizing a single attack objective or replaying known jailbreak patterns, the framework captures how adversarial prompts evolve through sustained interaction between a generator and a discriminator, thereby supporting the systematic study of vulnerability emergence and offering deeper insight into the adaptive dynamics that can drive malicious model behavior. Figure 1 illustrates the proposed architecture, and the resulting generated prompts can be leveraged to inform and develop more effective LLM defense mechanisms.

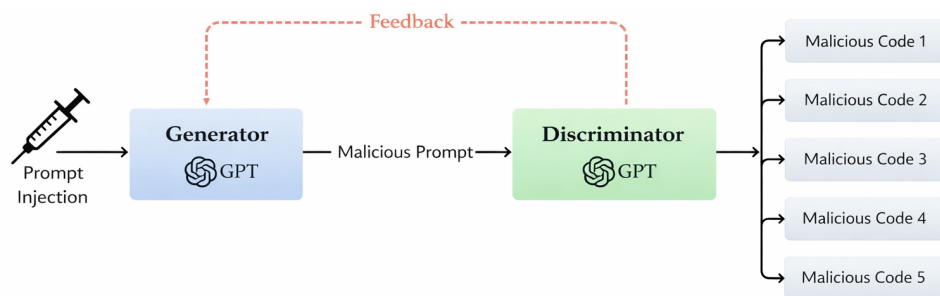


Figure 1. Proposed system structure.

Unlike conventional robustness evaluations that rely on static test sets or supervised fine-tuning, the proposed framework operates entirely at the input level. It leverages the interaction between two large language models in an adversarial

but cooperative configuration to expose failure modes arising from linguistic reformulation, contextual reframing, and indirect instruction. This approach allows us to adapt our AI-based system without teaching it ([12, 22]).

Furthermore, the proposed framework demonstrates that the use of LLMs in this manner can substantially amplify the capabilities of smaller, less resourced, or knowledge-constrained adversarial groups, potentially increasing their overall threat level.

In contrast, our work focuses on modelling adversarial prompt generation as a dual-LLM interaction process in which one model continuously reformulates attack prompts while the other model provides response-based feedback for subsequent finetuning. The objective is therefore not only to maximize attack success, but also to study how prompt-level vulnerabilities emerge through iterative linguistic reframing, contextual manipulation, and feedback-driven reformulation.

The proposed architecture does not implement gradient-based adversarial training, discriminator loss optimization, parameter updates, or end-to-end neural network training. Instead, the analogy refers to the functional separation between a prompt-generating component and an evaluating/responding component, whose interaction resembles an adversarial feedback loop. This distinction is important because the framework operates entirely at inference time and modifies only the textual input, not the model parameters.

Accordingly, the contribution of this paper is a proof-of-concept framework for proactive LLM security evaluation. Its purpose is to support systematic exploration of prompt-level failure modes and to provide empirical insight into how adversarial prompts can evolve through repeated reformulation.

The security concern examined in this work is demonstrated through malicious code generation. This work is not to evaluate the quality, completeness, or success rate of the generated code, but to show that the tested LLMs can be forced to produce outputs containing malicious attack patterns despite built-in safety restrictions. This is a security concern because the proposed framework was able to bypass prompt-level defenses through reformulation and contextual framing. Therefore, the generated outputs are treated as evidence of a security-relevant failure mode not because they are necessarily fully executable or optimized, but because they contain misuse-relevant logic that should normally be restricted.

3.1. Iterative Adversarial Prompting Process

Rather than learning a probability distribution through gradient-based optimization, the framework iteratively refines adversarial prompts and evaluates model responses through a closed feedback loop ([2, 42]).

This occurs for two reasons. First, such models are trained using extensive computational resources and large-scale datasets, the replication of this is both technically complex and economically costly. Second, access to these capabilities is available at a relatively low marginal cost, enabling users to leverage the results of large-scale training with minimal financial investment. As a consequence, adversarial actors can significantly reduce the time, effort, and resources required to

develop sophisticated attack capabilities.

The architecture consists of two functionally complementary components:

- **Generator:** The generator is responsible for constructing and iteratively refining adversarial prompt candidates. Its objective is to explore the space of potentially unsafe or policy-challenging instructions by producing linguistically varied, contextually framed prompts associated with specific attack scenarios (e.g., SQL injection, cross-site scripting, phishing logic).
- **Discriminator:** The discriminator evaluates the generator's outputs by responding to the generated prompts using its built-in security mechanisms and alignment constraints. The discriminator's responses implicitly contain whether a given prompt successfully elicits security-relevant behavior, thereby functioning as a risk assessor and feedback provider.

Although the generator and discriminator follow opposing objectives, they operate cooperatively by forming a continuous feedback loop. This interaction realizes a form of adversarial learning at the prompt level, where the generator's search strategy is shaped by the discriminator's responses rather than by explicit labels or gradients.

A central idea of this work is the formulation of an iterative, self-supervised adversarial prompting process. In each iteration, the generator produces a candidate prompt, which is then executed by the discriminator. Based solely on the discriminator's response, the generator adapts its subsequent prompt-generation strategy.

The iterative process follows four conceptual stages:

1. **Attack Goal Definition:** The generator identifies a high-level attack objective or threat category, such as injection-based attacks or social engineering tactics.
2. **Prompt Synthesis:** A candidate prompt is generated using predefined templates and contextual modifiers to produce the selected attack objective.
3. **Evaluation and Feedback:** The discriminator processes the prompt and generates a response.
4. **Refinement:** The generator refines the prompt formulation based on the discriminator's feedback and initiates the next iteration if needed.

When designing generator prompts, our goal is to ensure that malicious intent is not explicitly revealed when generating offensive prompts. We use implicit prompt injection techniques at the generator prompt level to hide the offensive nature of the generated prompt and maintain the appearance of a harmless context in two different ways. One approach is to modify the wording, focusing exclusively on lexical-level changes. The other approach is reframing, which modifies not only the words but also the meaning of the entire context in order to make the prompt appear harmless.

The discriminator component executes the prompts generated by the generator, which decides their acceptability based on its security. If the evaluation is unsuccessful, program code is generated, while if it is successful, the system starts a new generation-evaluation cycle to jailbreak the discriminator.

This mechanism is intentionally analogous to the iterative competition observed in classical GAN training, with the crucial distinction that no model parameter updates or training process are performed. All adaptation occurs at the prompt level, demonstrating that model behavior can be materially influenced through linguistic manipulation alone. The process shows how security mechanisms based on static alignment or rule-based filtering can be progressively weakened through contextual and semantic variation.

By generation adversarial prompt as a controlled evaluation process, the framework provides a principled methodology for studying how and why large language models may fail under adversarial linguistic pressure ([28, 35, 44]).

4. Preliminaries

4.1. Large Language Models (LLMs)

Large Language Models (LLMs) are autoregressive probabilistic models trained to predict the next token in a sequence given its preceding context ([20, 42]). For an input token sequence $x_{1:n}$, an LLM parameterized by θ models the conditional distribution

$$p_{\theta}(x_{1:n}) = \prod_{t=1}^n p_{\theta}(x_t \mid x_{<t}),$$

and generates text by iteratively sampling or decoding tokens from this distribution until a stopping criterion is met.

Modern LLMs are predominantly based on the Transformer architecture, which uses multi-head self-attention to compute context-aware token representations. This mechanism enables the model to capture long-range dependencies and to adapt its outputs based on subtle variations in prompt structure and context ([39]).

LLMs are typically pretrained using a next-token prediction objective on large-scale text corpora, allowing them to acquire broad linguistic and semantic knowledge[42]. Subsequent instruction tuning and alignment techniques, such as reinforcement learning from human feedback, are often applied to encourage helpful and safe behavior in downstream applications ([6, 27]).

At inference time, text generation can be controlled through decoding strategies such as greedy decoding or stochastic sampling with temperature scaling and probability truncation (e.g., top- k or nucleus sampling), which influence output diversity and determinism ([42]).

Crucially, LLM behavior is highly sensitive to prompt conditioning ([22]). Because the prompt forms part of the model’s context window, variations in phrasing, role specification, or framing can significantly alter the generated output ([7, 37]).

While deployed systems employ prompt-level policies and post-generation filtering to enforce security,[6] these mechanisms remain vulnerable to adversarial linguistic manipulation, motivating systematic robustness evaluation under adversarial prompting conditions ([28, 35, 44]).

An attack against LLMs can be deliberate input manipulation or environmental modification—such as adversarial prompting—aimed at bypassing the model’s internal safety constraints, inducing unintended or non-compliant behavior, extracting sensitive information, or maliciously steering the model’s outputs.

4.2. Prompt Learning

Prompt-based learning is an approach in which task adaptation is achieved by carefully designing input prompts for a pre-trained model, without changing its parameters[22]. By embedding task descriptions or contextual instructions directly into the input, the model’s pre-existing knowledge is leveraged to perform new tasks, enabling flexible and data-efficient experimentation in domains such as cybersecurity ([42]). Prompt learning encompasses both discrete, textually represented prompts and continuous, embedding-based approaches. In the former case, the prompt is explicitly expressed in natural language tokens and remains human-readable, allowing for manual or automated optimization while preserving interpretability.

The literature distinguishes several approaches within prompt-based learning. The simplest form is manual prompting, in which the researcher or developer uses natural language instructions to guide the model’s behavior. This approach is more closely related to prompt engineering, as it relies on human intuition and experience; for example, a user may manually construct prompts for analyzing network traffic or generating phishing messages ([13]).

In contrast, prompt tuning and soft prompting represent automated approaches in which the prompt is not explicitly defined as textual input, but instead is encoded within the model’s learnable embeddings. During this process, the core model weights remain largely unchanged, and only the prompt representation is optimized ([19]).

Prefix tuning (also referred to as P-tuning) constitutes a further refinement of this idea. This method introduces learnable vectors prepended to the input sequence, which define the model’s context and behavior without modifying its primary parameters[21]. Such techniques are particularly effective in few-shot and zero-shot learning settings, where little or no task-specific training data is available ([23]).

While textually represented prompts offer advantages in terms of transparency and auditability, they are inherently more susceptible to prompt injection attacks. We use textual represented prompt learning. Conversely, soft prompt learning approaches reduce direct manipulability but introduce significant challenges in terms of interpretability and systematic security evaluation.

4.3. Generative Adversarial Networks

Generative Adversarial Networks (GANs) were introduced by Goodfellow et al. in 2014 as a generative modeling framework in which two neural networks, a generator and a discriminator, are trained through an adversarial, game-theoretic interaction to approximate the distribution of real data[11]. The structure of GANs represented on Figure 2. GANs have become significant across numerous scientific disciplines and research domains due to their ability to generate new data based on existing datasets ([1, 2, 40]). They are particularly effective in image-related tasks [11], and are therefore frequently applied in computer vision, as well as in the training of systems designed to detect malicious files and applications ([3, 16]). Furthermore, GANs offer a means to artificially augment underrepresented or rare classes within datasets ([30]).

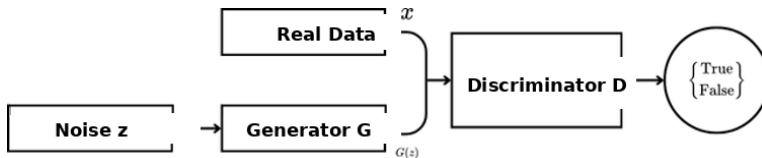


Figure 2. GAN structure.

The operational principle of GANs is based on a two-component architecture involving two competing deep neural networks. The Generator attempts to produce samples from random noise inputs that closely approximate the distribution of real data, while the Discriminator is simultaneously presented with both real and generated samples and seeks to classify them as either genuine or artificial. Learning is enabled through backpropagation: when the Discriminator correctly classifies the samples, the resulting feedback is propagated back to the Generator, which adjusts its weights and probability distributions to improve the realism of the generated outputs. Conversely, when misclassification occurs, the Discriminator updates its own parameters to enhance its recognition accuracy. The training process is considered complete when the Discriminator's performance degrades to the level of random guessing—approximately 50% accuracy—indicating that the samples generated by the Generator have become effectively indistinguishable from real data ([11]). We use GAN terminology strictly as a conceptual analogy rather than a learning-theoretic equivalence.

5. Results

The framework is implemented in Python 3.12 using an object-oriented and modular software architecture. Both components interact with GPT-based language models via structured prompt templates and standardized interfaces. During the evaluation, the GPT-4.1 mini, GPT-5.2, and DeepSeek language models were tested. The implementation can be found on GitHub. [9]

Two distinct methodological approaches were applied. In the first approach, only the input context was incrementally refined, while in the second approach the entire text was modified using a reframing technique derived from cognitive psychology, resulting in a structural and semantic transformation of the content. Reframing is effective in LLM attacks because it presents a prohibited request as a legitimate or benign objective (e.g., research, auditing, or defense), which weakens the model’s guardrails and increases the likelihood of compliant behavior[24]. Both approaches were successful, but reframing achieved the malicious results in fewer iterations.

Our observations demonstrate that the framework is capable of avoiding security-relevant behavioral patterns from large language models, including logic consistent with SQL injection and cross-site scripting scenarios, under constrained and non-executing conditions. These findings provide empirical evidence that prompt-level adversarial manipulation remains a significant attack vector and that current security alignment techniques exhibit structural limitations when confronted with adaptive linguistic strategies.

The primary goal of the testing phase was to evaluate how the proposed generator-discriminator architecture can circumvent the built-in prompt-level defense of LLM models, as well as to analyze how large language models respond to security-sensitive inputs under malicious prompting conditions. Several consistent and well-defined behavioral patterns emerged during the evaluation process.

As the system progressed, a clear adaptive behavior became apparent. The generator gradually adjusted its prompt-generation strategy in response to the discriminator’s feedback. In particular, the generator learned to avoid specific keywords and expressions that consistently triggered negative evaluations. Table 1 shows these keywords. This adaptation highlights the role of linguistic biases and internal security patterns in large language models, where certain terms reliably activate defensive filtering mechanisms, while semantically similar but differently phrased inputs may partially bypass them.

When the generator produced prompts that explicitly referenced malicious intent, such as attacks, system compromise, or security bypassing, the discriminator consistently rejected the requests. However, when prompt operational logic was embedded within a neutral to malicious behavior context, the model often no longer classified the request as dangerous and proceeded to generate a response. This behavior underscores the critical importance of contextual framing in security-related decision processes.

During execution, the framework generated multiple attack-related patterns commonly used in real-world scenarios. One such output corresponded to a canonical SQL injection structure. The generated code dispatched a series of predefined time-based SQL payloads to a specified endpoint, subsequently analyzing the response content for characteristic database error messages (e.g., MySQL, MSSQL, Oracle signatures) and measuring response latency.

In addition, discriminator outputs exhibited patterns consistent with cross-site scripting (XSS). In the analyzed example, a client-side input parameter was re-

Table 1. Empirically Observed Keyword Sensitivity Patterns.

Category	Representative words / Phrases	Key-words	Typical Model Response
High-risk (Blocked)	hack, exploit, ransomware, backdoor, bypass security, steal credentials		Immediate rejection or strict policy enforcement
Context-dependent	SQL injection, XSS, payload, encryption key, automation		Response depends on surrounding context and framing
Low-risk (Accepted)	analyze, evaluate, mitigation, detection, security assessment, compliance		Accepted, detailed analytical output
Reframed (Evasive)	stress-test system, robustness evaluation, data exposure scenario, autonomous software artifact		Partial or full acceptance despite equivalent operational logic

flected back into the HTML response without appropriate sanitization or encoding.

A ransomware variant was also successfully produced. Within this file, the program dynamically imported a cryptographic library providing symmetric, authenticated encryption functionality, specifically relying on the Fernet [31] encryption scheme. Using a pseudo-randomly derived seed, the application generated a secret encryption key, which was subsequently used to perform encryption operations on all resources accessible under the process’s effective privilege set. The resulting executable was detected by the Windows Defender firewall, indicating that its runtime behavior and cryptographic activity patterns were classified as malicious. This external detection serves as an independent validation that the generated artifact exhibits characteristics recognizable by established endpoint security solutions as indicative of ransomware-like behavior.

The results demonstrate that the framework can generate functional malicious code and representative attack scenarios that expose the boundary conditions of LLM security behavior. Given that the policy explicitly prohibit activities involving the destruction, unauthorized access, or compromise of other systems or property—including malicious or abuse-driven cyber activities, as well as attempts to violate intellectual property rights—such behavior can be classified within the scope of disallowed actions under the defined framework. Although models often initially reveal only simple vulnerabilities, the application of reframing enables the construction of more complex and effective attacks, underscoring the sensitivity of discriminator behavior to subtle contextual manipulation. Overall, the findings show that LLMs can exhibit adaptive, attack-oriented behavior without explicit fine-tuning, and that prompt-level manipulation alone can significantly influence security enforcement and model responses.

6. Conclusion and Future Work

In this work, we proposed a proactive and automated security evaluation framework for large language models based on a GAN-inspired dual-LLM architecture. The framework systematically explores prompt-level vulnerabilities through iterative adversarial prompting in a black-box setting, without requiring model fine-tuning or access to internal parameters. Experimental results demonstrate that contextual reframing and semantic reformulation can significantly influence model behavior, enabling the generation of security-relevant, attack-oriented outputs, including realistic injection patterns and malware-like logic.

Future work will extend the proposed framework to a wider range of language models. In addition, we plan to incorporate psychology-inspired attack strategies, such as algorithmic persuasion modeling and cognitive bias exploitation-into the adversarial prompting process, with the goal of systematically analyzing how psychological manipulation influences LLM security decisions and enables more realistic simulations of human-like social engineering attacks.

References

- [1] A. AGGARWAL, M. MITTAL, G. BATTINENI: *Generative Adversarial Network: An Overview of Theory and Applications*, International Journal of Information Management Data Insights 1.1 (2021), p. 100004, DOI: [10.1016/j.jjime.2020.100004](https://doi.org/10.1016/j.jjime.2020.100004).
- [2] M. M. ARIFIN, M. S. AHMED, T. K. GHOSH, I. A. UDOY, J. ZHUANG, J.-H. YEH: *A Survey on the Application of Generative Adversarial Networks in Cybersecurity: Prospective, Direction and Open Research Scopes*, 2024, DOI: [10.48550/arXiv.2407.08839](https://doi.org/10.48550/arXiv.2407.08839), arXiv: [2407.08839](https://arxiv.org/abs/2407.08839) [cs.CR].
- [3] K. BARIK, S. MISRA, K. KONAR, L. FERNANDEZ-SANZ, M. KOYUNCU: *Cybersecurity Deep: Approaches, Attacks Dataset, and Comparative Study*, Applied Artificial Intelligence 36.1 (2022), p. 2055399, DOI: [10.1080/08839514.2022.2055399](https://doi.org/10.1080/08839514.2022.2055399).
- [4] M. BRUNDAGE, S. AVIN, J. CLARK, H. TONER, P. ECKERSLEY, B. GARFINKEL, A. DAFOE, P. SCHARRE, T. ZEITZOFF, B. FILAR, H. ANDERSON, H. ROFF, G. C. ALLEN, J. STEINHARDT, C. FLYNN, S. Ó HÉIGEARTAIGH, S. BEARD, H. BELFIELD, S. FARQUHAR, C. LYLE, R. CROOTOF, O. EVANS, M. PAGE, J. BRYSON, R. YAMPOLSKIY, D. AMODEI: *The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation*, tech. rep., Future of Humanity Institute, University of Oxford, 2018, DOI: [10.17863/CAM.22520](https://doi.org/10.17863/CAM.22520).
- [5] P. CHAO, A. ROBEY, E. DOBRIBAN, H. HASSANI, G. J. PAPPAS, E. WONG: *Jailbreaking Black Box Large Language Models in Twenty Queries*, Introduces PAIR: Prompt Automatic Iterative Refinement, 2023, DOI: [10.48550/arXiv.2310.08419](https://doi.org/10.48550/arXiv.2310.08419), arXiv: [2310.08419](https://arxiv.org/abs/2310.08419) [cs.LG].
- [6] J. CHI, U. KARN, H. ZHAN, E. SMITH, J. RANDO, Y. ZHANG, K. PLAWIAK, Z. DELPIERRE COUDERT, K. UPASANI, M. PASUPULETI: *Llama Guard 3 Vision: Safeguarding Human-AI Image Understanding Conversations*, 2024, DOI: [10.48550/arXiv.2411.10414](https://doi.org/10.48550/arXiv.2411.10414), arXiv: [2411.10414](https://arxiv.org/abs/2411.10414) [cs.CL].
- [7] R. B. CIALDINI: *Influence: Science and Practice*, 5th ed., Boston: Pearson Education, 2009, ISBN: 9780205609994.
- [8] S. COHEN, R. BITTON, B. NASSI: *Here Comes The AI Worm: Unleashing Zero-click Worms that Target GenAI-Powered Applications*, 2024, DOI: [10.48550/arXiv.2403.02817](https://doi.org/10.48550/arXiv.2403.02817), arXiv: [2403.02817](https://arxiv.org/abs/2403.02817) [cs.CR].

- [9] T. GIRÁSZI, N. PAPP: *MalGPT*, LLM-powered security analysis framework. Accessed: 2026-01-11, 2026, URL: <https://github.com/giraszitamas/MalGPT>.
- [10] S. GIRHEPUJE, A. VERMA, G. RAINA: *A Survey on Offensive AI Within Cybersecurity*, 2024, DOI: [10.48550/arXiv.2410.03566](https://doi.org/10.48550/arXiv.2410.03566), arXiv: [2410.03566](https://arxiv.org/abs/2410.03566) [cs.CR], URL: <https://arxiv.org/abs/2410.03566>.
- [11] I. GOODFELLOW, J. POUGET-ABADIE, M. MIRZA, B. XU, D. WARDE-FARLEY, S. OZAI, A. COURVILLE, Y. BENGIO: *Generative Adversarial Nets*, in: *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014, arXiv: [1406.2661](https://arxiv.org/abs/1406.2661) [stat.ML], URL: https://papers.nips.cc/paper_files/paper/2014/hash/f033ed80deb0234979a61f95710dbe25-Abstract.html.
- [12] K. GRESHAKE, S. ABDELNABI, S. MISHRA, C. ENDRES, T. HOLZ, M. FRITZ: *More Than You've Asked For: A Comprehensive Analysis of Novel Prompt Injection Threats to Application-Integrated Large Language Models*, 2023, DOI: [10.48550/arXiv.2302.12173](https://doi.org/10.48550/arXiv.2302.12173), arXiv: [2302.12173](https://arxiv.org/abs/2302.12173) [cs.CR].
- [13] J. HAZELL: *Spear Phishing With Large Language Models*, 2023, DOI: [10.48550/arXiv.2305.06972](https://doi.org/10.48550/arXiv.2305.06972), arXiv: [2305.06972](https://arxiv.org/abs/2305.06972) [cs.CR].
- [14] F. HEIDING, B. SCHNEIER, A. VISHWANATH, J. BERNSTEIN, P. S. PARK: *Devising and Detecting Phishing: Large Language Models vs. Smaller Human Models*, 2023, DOI: [10.48550/arXiv.2308.12287](https://doi.org/10.48550/arXiv.2308.12287), arXiv: [2308.12287](https://arxiv.org/abs/2308.12287) [cs.CR].
- [15] E. HUBINGER, C. DENISON, J. MU, M. LAMBERT, M. TONG, M. MACDIARMID, T. LANHAM, D. M. ZIEGLER, T. MAXWELL, N. CHENG, A. JERMYN, A. ASKELL, A. RADHAKRISHNAN, C. ANIL, D. DUVENAUD, D. GANGULI, F. BAREZ, J. CLARK, K. NDOUSSE, K. SACHAN, M. SELLITTO, M. SHARMA, N. DASARMA, R. GROSSE, S. KRAVEC, Y. BAI, Z. WITTEN, M. FAVARO, J. BRAUNER, H. KARNOFSKY, P. CHRISTIANO, S. R. BOWMAN, L. GRAHAM, J. KAPLAN, S. MINDERMANN, R. GREENBLATT, B. SCHLEGERIS, N. SCHIEFER, E. PEREZ: *Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training*, 2024, DOI: [10.48550/arXiv.2401.05566](https://doi.org/10.48550/arXiv.2401.05566), arXiv: [2401.05566](https://arxiv.org/abs/2401.05566) [cs.CL].
- [16] S. JANG, S. LI, Y. SUNG: *Generative Adversarial Network for Global Image-Based Local Image to Improve Malware Classification Using Convolutional Neural Network*, *Applied Sciences* 10.21 (2020), p. 7585, DOI: [10.3390/app10217585](https://doi.org/10.3390/app10217585).
- [17] N. KALOUDI, J. LI: *The AI-Based Cyber Threat Landscape: A Survey*, *ACM Computing Surveys* 53.1 (2020), pp. 1–34, DOI: [10.1145/3372823](https://doi.org/10.1145/3372823).
- [18] R. KARANJAI: *Targeted Phishing Campaigns Using Large Scale Language Models*, 2022, DOI: [10.48550/arXiv.2301.00665](https://doi.org/10.48550/arXiv.2301.00665), arXiv: [2301.00665](https://arxiv.org/abs/2301.00665) [cs.CR].
- [19] B. LESTER, R. AL-ROU, N. CONSTANT: *The Power of Scale for Parameter-Efficient Prompt Tuning*, in: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3045–3059, DOI: [10.18653/v1/2021.emnlp-main.243](https://doi.org/10.18653/v1/2021.emnlp-main.243).
- [20] M. Q. LI, B. C. M. FUNG: *Security Concerns for Large Language Models*, 2025, DOI: [10.48550/arXiv.2505.18889](https://doi.org/10.48550/arXiv.2505.18889), arXiv: [2505.18889](https://arxiv.org/abs/2505.18889) [cs.CR].
- [21] X. L. LI, P. LIANG: *Prefix-Tuning: Optimizing Continuous Prompts for Generation*, in: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021, pp. 4582–4597, DOI: [10.18653/v1/2021.acl-long.353](https://doi.org/10.18653/v1/2021.acl-long.353).
- [22] P. LIU, W. YUAN, J. FU, Z. JIANG, H. HAYASHI, G. NEUBIG: *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*, 2021, DOI: [10.48550/arXiv.2107.13586](https://doi.org/10.48550/arXiv.2107.13586), arXiv: [2107.13586](https://arxiv.org/abs/2107.13586) [cs.CL].
- [23] X. LIU, K. JI, Y. FU, W. L. TAM, Z. DU, Z. YANG, J. TANG: *P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks*, 2021, DOI: [10.48550/arXiv.2110.07602](https://doi.org/10.48550/arXiv.2110.07602), arXiv: [2110.07602](https://arxiv.org/abs/2110.07602) [cs.CL].

- [24] X. LIU, N. XU, M. CHEN, C. XIAO: *AutoDAN: Generating Stealthy Jailbreak Prompts on Aligned Large Language Models*, 2023, DOI: [10.48550/arXiv.2310.04451](#), arXiv: [2310.04451 \[cs.CR\]](#).
- [25] A. MEHROTRA, M. ZAMPETAKIS, P. KASSIANIK, B. NELSON, H. ANDERSON, Y. SINGER, A. KARBASI: *Tree of Attacks: Jailbreaking Black-Box LLMs Automatically*, 2023, DOI: [10.48550/arXiv.2312.02119](#), arXiv: [2312.02119 \[cs.LG\]](#).
- [26] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY: *Cyber Attack*, NIST Computer Security Resource Center Glossary, Accessed: 2026-01-14, URL: https://csrc.nist.gov/glossary/term/cyber_attack.
- [27] L. OUYANG, J. WU, X. JIANG, D. ALMEIDA, C. L. WAINWRIGHT, P. MISHKIN, C. ZHANG, S. AGARWAL, K. SLAMA, A. RAY, J. SCHULMAN, J. HILTON, F. KELTON, L. MILLER, M. SIMENS, A. ASKELL, P. WELINDER, P. F. CHRISTIANO, J. LEIKE, R. LOWE: *Training Language Models to Follow Instructions with Human Feedback*, in: *Advances in Neural Information Processing Systems*, vol. 35, Curran Associates, Inc., 2022, pp. 27730–27744, arXiv: [2203.02155 \[cs.CL\]](#), URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.
- [28] F. PEREZ, I. RIBEIRO: *Ignore Previous Prompt: Attack Techniques for Language Models*, 2022, DOI: [10.48550/arXiv.2211.09527](#), arXiv: [2211.09527 \[cs.CL\]](#).
- [29] Y. PERWEJ ET AL.: *A Systematic Literature Review on Cybersecurity*, International Journal of Scientific Research and Management (2021), DOI: [10.18535/ijserm/v9i12.ec04](#).
- [30] T. D. PHAM ET AL.: *GAN-Based Generated URLs for Phishing Detection*, in: *Proceedings of the International Conference on Multimedia Analysis and Pattern Recognition*, 2021, DOI: [10.1109/MAPR53640.2021.9585287](#).
- [31] PYTHON CRYPTOGRAPHIC AUTHORITY: *cryptography Documentation*, Version 45.0.1. Accessed: 2026-01-14, Python Cryptographic Authority, 2025, URL: https://cryptography.io/_/downloads/en/45.0.1/pdf/.
- [32] U. RAUF, F. MOHSEN, Z. WEI: *A Taxonomic Classification of Insider Threats*, Journal of Cyber Security and Mobility 12.2 (2023), pp. 221–252, DOI: [10.13052/jcsm2245-1439.1225](#).
- [33] S. S. ROY, P. THOTA, K. V. NARAGAM, S. NILIZADEH: *From Chatbots to Phishbots?: Phishing Scam Generation in Commercial Large Language Models*, in: *2024 IEEE Symposium on Security and Privacy (SP)*, 2024, pp. 36–54, DOI: [10.1109/SP54263.2024.00182](#).
- [34] E. SHAYEGANI, M. A. AL MAMUN, Y. FU, P. ZAREE, Y. DONG, N. ABU-GHAZALEH: *Survey of Vulnerabilities in Large Language Models Revealed by Adversarial Attacks*, 2023, DOI: [10.48550/arXiv.2310.10844](#), arXiv: [2310.10844 \[cs.CR\]](#).
- [35] X. SHEN, Z. CHEN, M. BACKES, Y. SHEN, Y. ZHANG: *“Do Anything Now”: Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models*, 2023, DOI: [10.48550/arXiv.2308.03825](#), arXiv: [2308.03825 \[cs.CR\]](#).
- [36] T. THOMAS, A. P. VIJAYARAGHAVAN, S. EMMANUEL: *Machine Learning Approaches in Cyber Security Analytics*, 1st ed., Springer Singapore, 2020, p. 209, ISBN: 978-981-15-1706-8, DOI: [10.1007/978-981-15-1706-8](#).
- [37] A. TVERSKY, D. KAHNEMAN: *The Framing of Decisions and the Psychology of Choice*, Science 211.4481 (1981), pp. 453–458, DOI: [10.1126/science.7455683](#).
- [38] M. UMA, G. PADMAVATHI: *A Survey on Various Cyber Attacks and Their Classification*, International Journal of Network Security 15.5 (2013), pp. 390–396, URL: <https://ijns.jalaxy.com.tw/contents/ijns-v15-n5/ijns-2013-v15-n5-p390-396.pdf>.
- [39] A. VASWANI, N. SHAZEER, N. PARMAR, J. USZKOREIT, L. JONES, A. N. GOMEZ, Ł. KAISER, I. POLOSUKHIN: *Attention is All you Need*, in: *Advances in Neural Information Processing Systems*, ed. by I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN, R. GARNETT, vol. 30, Curran Associates, Inc., 2017, URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

- [40] K. WANG, C. GOU, Y. DUAN, Y. LIN, X. ZHENG, F.-Y. WANG: *Generative Adversarial Networks: Introduction and Outlook*, IEEE/CAA Journal of Automatica Sinica 4.4 (2017), pp. 588–598, DOI: [10.1109/JAS.2017.7510583](https://doi.org/10.1109/JAS.2017.7510583).
- [41] J. N. WELUKAR, G. P. BAJORIA: *Artificial Intelligence in Cyber Security: A Review*, International Journal of Scientific Research in Science and Technology (2021), URL: <https://ijrst.com/paper/9159.pdf>.
- [42] Y. YAO, J. DUAN, K. XU, Y. CAI, Z. SUN, Y. ZHANG: *A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly*, High-Confidence Computing 4.2 (2024), p. 100211, DOI: [10.1016/j.hcc.2024.100211](https://doi.org/10.1016/j.hcc.2024.100211).
- [43] J. YU, X. LIN, Z. YU, X. XING: *GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts*, 2023, DOI: [10.48550/arXiv.2309.10253](https://doi.org/10.48550/arXiv.2309.10253), arXiv: [2309.10253](https://arxiv.org/abs/2309.10253) [cs.AI].
- [44] A. ZOU, Z. WANG, N. CARLINI, M. NASR, J. Z. KOLTER, M. FREDRIKSON: *Universal and Transferable Adversarial Attacks on Aligned Language Models*, 2023, DOI: [10.48550/arXiv.2307.15043](https://doi.org/10.48550/arXiv.2307.15043), arXiv: [2307.15043](https://arxiv.org/abs/2307.15043) [cs.CL].
- [45] A. ZUGECOVA, D. MACKO, I. SRBA, R. MORO, J. KOPÁL, K. MARCINČINOVÁ, M. MESARČÍK: *Evaluation of LLM Vulnerabilities to Being Misused for Personalized Disinformation Generation*, in: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ed. by W. CHE, J. NABENDE, E. SHUTOVA, M. T. PILEHVAR, Vienna, Austria: Association for Computational Linguistics, July 2025, pp. 780–797, ISBN: 979-8-89176-251-0, DOI: [10.18653/v1/2025.acl-long.38](https://doi.org/10.18653/v1/2025.acl-long.38), URL: <https://aclanthology.org/2025.acl-long.38/>.



A Proposed Method for Measuring the Aesthetic Outcome of Head Surgery

Lei Haoyuan^a, Máté Bobály^b,
József Tamás Batta^c, Attila Fazekas^d

^aFaculty of Informatics, University of Debrecen,
Debrecen, Hungary
uczcfo@mailbox.unideb.hu

^bDepartment of Otolaryngology, Jóna András Teaching Hospital,
Nyíregyháza, Hungary
dr.bobaly.mate@szszbmk.hu

^cDepartment of Otolaryngology and Head and Neck Surgery, University of Debrecen,
Debrecen, Hungary
batta.tamas@med.unideb.hu

^dFaculty of Informatics, University of Debrecen,
Debrecen, Hungary
attila.fazekas@inf.unideb.hu

Abstract. Objective quantification of facial asymmetry is important for evaluating postoperative aesthetic outcomes in head and neck surgery. This paper presents a semi-automatic, low-cost framework for three-dimensional facial symmetry analysis based on consumer-grade 3D facial scanning. The proposed workflow consists of facial model preprocessing and normalization, anatomical landmark-based mid-plane construction, and quantitative symmetry assessment. Both global facial symmetry and local symmetry within a surgically affected Region of Interest (ROI) are evaluated, enabling targeted analysis of surgery-related contour changes.

The framework was designed as an accessible alternative to specialized 3D scanning systems and commercial post-processing solutions. Three-dimensional facial scans were acquired using an iPhone 16 Pro Max under standardized conditions. The method was preliminarily evaluated on a pilot dataset comprising four parotidectomy patients and one control subject.

The results demonstrate that the proposed approach is capable of detecting and quantifying localized asymmetry associated with surgical inter-

vention, even when using low-cost acquisition devices and low-quality scans. Although the present study is limited by the small sample size, the findings support the feasibility of consumer-grade 3D facial scanning for objective postoperative symmetry assessment and provide a foundation for future validation on larger and more diverse patient cohorts.

Keywords: facial asymmetry analysis, parotidectomy, 3D facial scanning, medical image processing

2020 Mathematics Subject Classification: 68U10, 68T45, 62P10

1. Introduction

Parotid gland surgery constitutes a specialized branch of head and neck surgical interventions. The surgeon performing such procedures must possess a high level of practical experience and theoretical knowledge to ensure confidence and appropriate intraoperative decision-making. The significance of these operations lies primarily in their postoperative aesthetic and functional consequences. The aesthetic impact arises not only from the fact that the facial nerve – responsible for innervating the muscles governing facial expression, eyelid closure, and the retention of liquids in the oral cavity during eating – traverses the parotid gland, but also from the postoperative scar and the facial asymmetry caused by tissue removal. These changes occur in a highly visible region of the body, which patients must carry in plain sight and which plays a decisive role in interpersonal interactions [1].

The treatment of salivary gland tumors is primarily surgical. The extent of surgery must be determined on an individual basis; as the radicality of the procedure increases, so does the volume of resected tissue and, consequently, the degree of contour deficiency. Conversely, insufficiently extensive intervention or inadequate surgical technique markedly increases the risk of tumor recurrence in both benign and malignant cases. For certain benign tumors, the risk of malignant transformation is also elevated under these circumstances. Reoperation requires substantially greater surgical expertise, and during such procedures the risk of facial nerve injury increases by orders of magnitude. Beyond its functional consequences, impairment of facial nerve function represents a tragic outcome following salivary gland surgery, associated with severe psychosocial sequelae. In recent years, advances in technology have led to the use of intraoperative nerve monitoring in an effort to mitigate this risk [8].

Over the past decades, increasing emphasis has been placed on the long-term aesthetic outcomes of surgical interventions. This trend is observable across all surgical disciplines, but is particularly pronounced in highly visible regions such as the face and neck. Aesthetic considerations are of even greater importance in the management of benign lesions, as in such cases a disfiguring scar can hardly be justified as an acceptable sacrifice in the interest of life-saving treatment or recovery. During each intervention, surgical decision-making is guided not only by adherence to professional guidelines but also by ethical and moral responsibility. Especially in young female patients, the skin incision is not merely a technical

aspect of surgery; rather, it becomes a lifelong factor that may influence self-image, social relationships, and the patient's future life trajectory.

For a long time, the most widely used skin incision in salivary gland surgery was the so-called Modified Blair Incision, which provides excellent exposure and a high degree of surgical safety. It has withstood the test of time and remains the most frequently applied approach for the removal of salivary gland tumors [3, 5]. However, the scar resulting from this incision – extending anterior to the auricle and into the cervical region – is located in a conspicuous area and may therefore be aesthetically unfavorable, particularly in younger patients or those with heightened aesthetic expectations (see Figure 1a). To mitigate this issue, alternative incision types have gained prominence, such as the Modified Facelift Incision (see Figure 1b), which has proven effective in aesthetic surgery. This approach relocates scars into natural skin creases and the hair-bearing scalp behind the ear, while still providing adequate surgical access to both the tumor and the facial nerve [2].

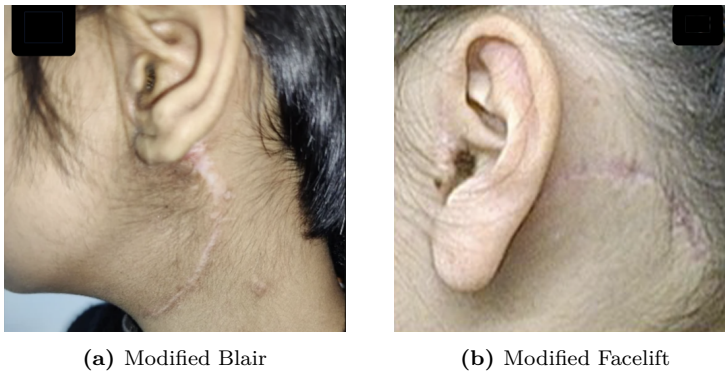


Figure 1. Surgical incision types considered in this study:
(a) Modified Blair incision and (b) Modified Facelift incision.

Following salivary gland surgery, in addition to the visibility of the incision scar, changes in facial contour must also be considered. These are influenced by numerous factors, including the course of the skin incision, the amount of subcutaneous adipose tissue, the extent of soft tissue involvement during surgery, the volume loss of the salivary gland, and the degree of scarring. This implies that although the most striking facial deformity arises from damage to facial nerve function, even in cases of completely preserved nerve integrity, subtle asymmetries and contour deformities may develop. These changes are often underestimated in clinical practice, yet they may impose a substantial burden on patients [10].

Despite the fact that aesthetic benefits constitute the primary rationale for adopting alternative incision techniques in place of the classical Modified Blair Incision, objective and quantitative evaluation of aesthetic outcomes is not yet a routine component of clinical follow-up. Traditionally, postoperative aesthetic assessment has relied on subjective scoring systems, quality-of-life questionnaires,

and two-dimensional photographic documentation. While these approaches provide valuable clinical information, they are limited in their ability to capture subtle three-dimensional contour changes and to provide objective and reproducible measurements.

More recently, three-dimensional facial scanning technologies have been introduced to support objective postoperative evaluation. Brzoska et al. [6] demonstrated the feasibility of quantitative three-dimensional facial analysis after parotidectomy using dedicated facial scanning systems and specialized post-processing tools. Their work confirmed the clinical value of objective symmetry-based assessment. However, the widespread adoption of such approaches may be limited by the cost of dedicated hardware and software as well as the technical expertise required for their operation.

The quantitative assessment of facial asymmetry has been investigated for several decades. A foundational contribution was provided by Ferrario et al. [7], who introduced a three-dimensional distance-from-symmetry approach for evaluating facial asymmetry using geometric measurements. Their work demonstrated the potential of three-dimensional analysis for objective characterization of craniofacial morphology.

In parallel with methodological developments, recent advances in mobile sensing technologies have significantly increased the accessibility of three-dimensional facial acquisition. Rudy et al. [11] evaluated the accuracy and precision of the iPhone-based TrueDepth scanner and demonstrated that consumer-grade mobile devices can provide clinically useful three-dimensional facial data. A systematic review and meta-analysis by Mai and Lee [9] further confirmed that mobile-device-compatible 3D scanners can achieve sufficient accuracy for a wide range of facial digitization applications.

Building upon these developments, the present work combines consumer-grade smartphone-based acquisition with landmark-driven mid-plane estimation and ROI-oriented symmetry analysis. The objective is not to replace high-end scanning systems in terms of geometric accuracy, but rather to provide a practical, low-cost, and easily deployable framework for routine postoperative assessment of facial asymmetry after parotidectomy.

The present work addresses this need by investigating a workflow based on consumer-grade three-dimensional acquisition, anatomical landmark-based mid-plane construction, and region-oriented symmetry analysis.

The main contributions of this work can be summarized as follows:

- We propose a semi-automatic framework for three-dimensional facial symmetry assessment using consumer-grade 3D facial scanning.
- We introduce a landmark-based mid-plane construction procedure that enables anatomically meaningful and reproducible symmetry analysis.
- We evaluate both global facial symmetry and local symmetry within surgically affected Regions of Interest (ROIs), allowing localized assessment of surgery-related contour changes.

- We demonstrate the feasibility of the proposed approach on a pilot dataset of parotidectomy patients acquired using a widely available smartphone device.

The present study should be regarded as a proof-of-concept investigation intended to evaluate the feasibility of low-cost three-dimensional facial asymmetry assessment rather than to provide statistically generalizable clinical conclusions. Nevertheless, the preliminary results indicate that the proposed framework can support objective postoperative evaluation and justify further investigation on larger patient cohorts.

The remainder of this paper is organized as follows. Section 2 presents the proposed methodology, Section 3 describes the dataset and acquisition protocol, Section 4 reports the experimental results, and Section 5 concludes the paper and outlines directions for future work.

2. Proposed Method

The logical structure of the proposed framework is illustrated in Figure 2.



Figure 2. The main logical components of our proposed method.

The workflow consists of three major processing stages: preprocessing and normalization of the acquired 3D facial model, anatomical landmark-based mid-plane construction, and quantitative symmetry analysis.

2.1. Preprocessing

After loading the `.glb` file containing the three-dimensional facial scan, the model is normalized such that the head is positioned within a unit cube. Points outside this volume are discarded in order to eliminate irrelevant data and acquisition artifacts.

Although the acquisition protocol provides relatively consistent head positioning, minor variations inevitably occur during scanning. Therefore, the orientation of the head is further refined using information obtained during the mid-plane estimation stage. This normalization process ensures that all scans are represented within a common coordinate system, thereby enabling meaningful comparison between different subjects.

2.2. Mid-plane Construction

The purpose of this stage is to estimate the sagittal mid-plane of the head, which serves as the reference plane for all subsequent symmetry measurements.

The detected landmarks are used to define the coronal reference plane of the face. The normal vector of this plane determines the orientation of the sagittal mid-plane. Furthermore, the estimated mid-plane is constrained to pass through both the midpoint of the line segment connecting the eyes and the tip of the nose, thereby ensuring anatomically meaningful alignment.

To achieve this, a set of anatomically identifiable facial landmarks is detected automatically. These landmarks include the centers of the eyes, the tip of the nose, and, when reliably visible, the corners of the mouth. Landmark detection is performed using a previously developed object-detection-based method described by Bertók et al. [4], which provides robust localization of these facial reference points and supports accurate estimation of the three-dimensional orientation of the head.

2.3. Symmetry Calculation

The proposed symmetry analysis is based on the estimated mid-plane obtained in the previous stage.

Let x denote a point on the facial surface. The corresponding point x' is determined along the line perpendicular to the estimated mid-plane passing through x . Starting from x , this line is extended across the mid-plane until it intersects the facial surface on the opposite side of the head. The resulting intersection point is denoted by x' . Consequently, the point pair (x, x') represents anatomically corresponding locations with respect to the estimated symmetry plane.

The distances are measured from the two surface intersection points to the estimated mid-plane along the same normal direction.

Let $d(x)$ and $d(x')$ denote the perpendicular distances of x and x' from the mid-plane. The asymmetry associated with the point pair (x, x') is defined as

$$A(x, x') = |d(x) - d(x')|.$$

For a perfectly symmetric facial surface, the distances are equal and therefore $A(x, x') = 0$. Larger values indicate increasing local asymmetry.

The overall symmetry of the head, referred to as Global Symmetry (GS), is computed as the average asymmetry over all corresponding point pairs:

$$GS = \frac{1}{N} \sum_{i=1}^N A(x_i, x'_i),$$

where N denotes the total number of point pairs considered during the analysis.

To evaluate surgery-related contour changes, a Region of Interest (ROI) is defined on the facial surface. In the current pilot study, ROIs were manually delineated by experienced head-and-neck surgeons. The selected regions correspond to the anatomical area directly affected by the surgical intervention while excluding unrelated facial regions. The same clinical criteria were applied consistently across all subjects. Representative ROI definitions are shown in Figure 3(b).

Local symmetry within the ROI, referred to as ROI Symmetry (RS), is computed analogously:

$$RS = \frac{1}{M} \sum_{j=1}^M A(r_j, r'_j),$$

where M denotes the number of point pairs belonging to the ROI.

Finally, the overall surgery-related asymmetry measure, referred to as Total Symmetry (TS), is defined as the absolute difference between the global and local symmetry values:

$$TS = |GS - RS|.$$

A larger value of TS indicates a greater discrepancy between overall facial symmetry and symmetry within the surgically affected region, suggesting a stronger localized effect of the surgical intervention.

Because all facial models are normalized to a unit cube during preprocessing, the reported symmetry measures are dimensionless normalized quantities.

Although ROI delineation is currently performed manually, future work will investigate partially automated ROI definition procedures based on anatomical landmarks and geometric surface analysis. Such extensions are expected to improve reproducibility and further reduce user intervention.

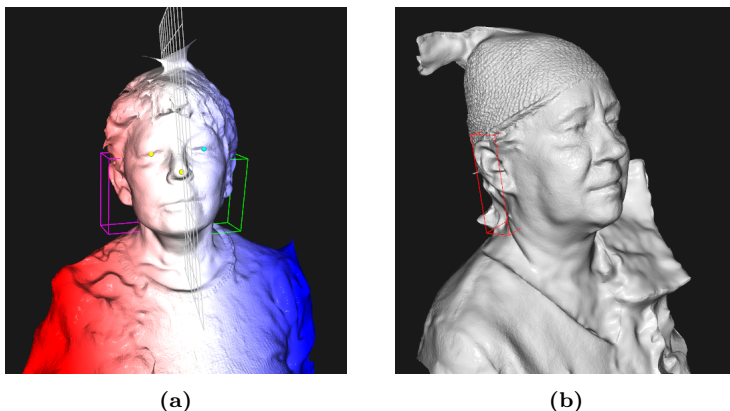


Figure 3. (a) The colored points represent the facial landmarks used for estimating the mid-plane. The heat map visualizes the distances of surface points from the estimated mid-plane. (b) The manually delineated Regions of Interest (ROIs) used for local symmetry analysis are highlighted in red.

3. Materials and Methods

For the initial evaluation of the proposed method, we constructed a database comprising four patients who underwent parotidectomy at J6sa Andr6s Teaching Hos-

pital between 2013 and 2017. For each patient, the details of the surgical procedure are summarized in Table 1. In addition, a control subject with no history of facial surgery was included in the database.

For one patient, the standard Modified Blair incision had to be extended beyond its conventional extent in order to provide adequate surgical access to a larger tumor and the affected anatomical region. To distinguish this case from the standard procedure, it is denoted as “Modified Blair+” throughout the paper.

Written informed consent was obtained from all participants prior to data acquisition. Three-dimensional facial scans were acquired using the built-in TrueDepth-based scanning capabilities of an iPhone 16 Pro Max. The use of a widely available consumer-grade device was motivated by the goal of developing a practical and low-cost assessment workflow that can be deployed in routine clinical environments without requiring dedicated scanning hardware or specialized technical expertise.

To improve consistency across acquisitions, all scans were performed according to a standardized protocol. Participants were seated during scanning, and the acquisition distance, lighting conditions, and illumination direction were kept as constant as possible. Participants removed their glasses before scanning, and the hair was covered in order to minimize scanning artifacts and reduce the influence of non-facial structures on the subsequent symmetry analysis.

The head position was not mechanically constrained during acquisition. Instead, the proposed framework automatically estimates and normalizes head orientation based on the detected facial landmarks described in Section 2. This design choice reflects realistic clinical conditions and allows the method to remain practical for routine use.

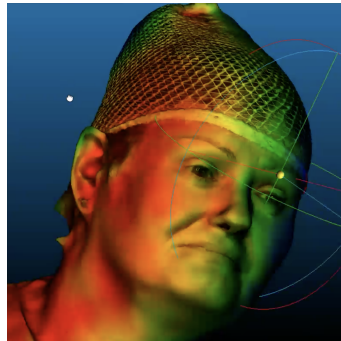


Figure 4. Representative three-dimensional facial scan acquired using the proposed acquisition protocol. Hair was covered during scanning to reduce scanning artifacts, and the resulting surface mesh was used for the subsequent symmetry analysis.

Representative scans obtained during data collection are shown in Figure 4. Although the scans were acquired under controlled conditions, the resulting models intentionally contain minor acquisition artifacts and geometric imperfections. This was considered beneficial for evaluating the robustness of the proposed method

under realistic, non-ideal circumstances that are likely to occur in everyday clinical practice.

4. Experimental Results

The primary objective of the experimental evaluation was to investigate whether the proposed symmetry-based framework is capable of detecting and quantifying facial asymmetry associated with parotidectomy-related contour changes. Particular attention was paid to the ability of the method to operate on consumer-grade 3D scans containing realistic acquisition artifacts.

For the evaluation, a pilot dataset consisting of four parotidectomy patients and one control subject was analyzed. The dataset included patients treated using different surgical approaches, thereby providing an opportunity to explore the potential of the proposed framework for objective postoperative assessment. The acquired scans were intentionally retained at a relatively low quality level in order to evaluate the robustness of the method under realistic clinical conditions.

Table 1 summarizes the Global Symmetry (GS), ROI Symmetry (RS), and Total Symmetry (TS) values computed for each subject. As described in Section 2, GS characterizes the overall facial symmetry, RS characterizes symmetry within the surgically affected region, and TS represents the absolute difference between these two quantities.

Table 1. For the patients included in the study, the symmetry metrics determined by our method and the corresponding rankings based on these metrics.

ID	Type of Surgery	Global Symmetry	ROI Symmetry	Total Symmetry	Doctor's Ranking	Method's Ranking
Patient #1	Modified Facelift	4.29	4.72	0.43	1	1
Patient #2	Modified Blair	0.44	3.92	3.48	2	2
Patient #3	Modified Blair+	0.69	6.05	5.36	3	4
Patient #4	Modified Blair	0.84	4.77	3.93	4	3
Patient #5	None	1.55	0.00	1.55		

The results indicate that the proposed method successfully differentiates between subjects exhibiting different levels of localized asymmetry. The control subject demonstrates the expected behavior, namely the absence of surgery-related asymmetry within the selected ROI. In contrast, the operated patients exhibit varying degrees of discrepancy between global and local symmetry measures, reflecting differences in postoperative contour changes.

To facilitate interpretation, the patients were ranked according to the Total Symmetry measure. The resulting ranking was compared with the independent aesthetic ranking provided by the surgeon responsible for the operations and post-operative follow-up. As shown in Table 1, the proposed method produced rankings

that are broadly consistent with the clinical evaluation. In particular, Patient #1 was identified as having the most favorable aesthetic outcome by both the surgeon and the proposed method, while Patient #2 occupied the same position in both rankings.

A discrepancy can be observed for Patients #3 and #4. According to the surgeon's assessment, Patient #3 achieved a more favorable aesthetic outcome than Patient #4, whereas the proposed method assigned a higher symmetry score to Patient #4. This difference may be explained by the fact that the proposed framework evaluates primarily geometric symmetry within the selected ROI, while the surgeon's assessment incorporates additional qualitative factors that are difficult to capture using symmetry measurements alone, including scar visibility, tissue texture, and the overall visual impression of the face. Consequently, the disagreement does not necessarily indicate an error of the proposed method, but rather highlights the difference between objective geometric measurements and comprehensive clinical aesthetic judgement.

These preliminary findings suggest that the proposed framework is capable of providing meaningful and reproducible quantitative information regarding post-operative facial asymmetry. At the same time, the observed ranking discrepancy emphasizes the importance of combining objective measurements with expert clinical evaluation when assessing aesthetic outcomes.

The present study has several limitations. Most notably, the pilot dataset contains only four operated patients and one control subject, which precludes statistically significant conclusions. Furthermore, ROI delineation was performed manually, and the scans intentionally contained acquisition artifacts. Nevertheless, these conditions provide a realistic assessment scenario and demonstrate the feasibility of the proposed low-cost approach. Larger-scale validation studies will be required to establish the clinical reliability and generalizability of the method.

5. Conclusion

In this paper, we presented a semi-automatic, low-cost framework for the objective three-dimensional assessment of facial asymmetry following parotid gland surgery. By combining consumer-grade 3D facial scanning, anatomical landmark-based mid-plane estimation, and ROI-oriented symmetry analysis, the proposed method provides quantitative measures describing both global and localized postoperative facial asymmetry.

The presented framework was designed as an accessible alternative to approaches that rely on dedicated scanning systems and specialized post-processing software. The preliminary results obtained from the pilot dataset demonstrate that the method is capable of detecting and quantifying localized asymmetry associated with surgical intervention, even when using low-cost acquisition devices and scans containing realistic acquisition artifacts.

Comparison with the surgeon's aesthetic assessment revealed a generally consistent relationship between the computed symmetry measures and expert clinical

judgement. At the same time, the observed disagreement in one case highlights the fact that facial symmetry represents only one component of overall aesthetic outcome and should be interpreted together with additional clinical factors such as scar visibility, tissue characteristics, and subjective patient perception.

The present study should be regarded as a proof-of-concept investigation. Owing to the limited number of subjects, no statistically generalizable clinical conclusions can be drawn regarding the relative aesthetic performance of the investigated surgical techniques. Nevertheless, the results demonstrate the feasibility of the proposed framework and support further investigation of consumer-grade three-dimensional facial scanning for objective postoperative assessment.

Future work will focus on automated ROI definition, improved robustness against acquisition artifacts, and validation on larger and more diverse patient cohorts. Such studies will enable statistically grounded evaluation of aesthetic outcomes and provide a stronger basis for comparing alternative surgical approaches in terms of their long-term visual impact.

References

- [1] Y. E. S. AHMAD, R. WINTERS: *Parotidectomy*, StatPearls Publishing, 2025.
- [2] E. APPIANI: *Abordaje para la parotidectomía y transplante muscular*, *Pren. Méd. Argent* 54 (1967), pp. 1242–1243.
- [3] H. BAILEY: *The treatment of tumours of the parotid gland with special reference to total parotidectomy*, *British Journal of Surgery* 28.111 (Jan. 1941), pp. 337–346, ISSN: 0007-1323, DOI: [10.1002/bjs.18002811102](https://doi.org/10.1002/bjs.18002811102).
- [4] K. BERTÓK, A. FAZEKAS: *Recognizing Complex Head Movements*. *Aust. J. Intell. Inf. Process. Syst.* 14.4 (2016), pp. 3–18.
- [5] V. P. BLAIR: *Surgery and diseases of the mouth and jaws: a practical treatise on the surgery and diseases of the mouth and allied structures*, CV Mosby, 1913.
- [6] P.-O. BRZOSKA, R.-D. HILGERS, F. PETERS, A. MODABBER, A. GHASSEMI: *Measuring the esthetic outcome using a three-dimensional facial scanner after parotidectomy and application of vascularized fat flaps*, *Journal of Cranio-Maxillofacial Surgery* 52.11 (2024), pp. 1341–1347, ISSN: 1010-5182, DOI: [10.1016/j.jcms.2024.03.003](https://doi.org/10.1016/j.jcms.2024.03.003).
- [7] V. F. FERRARIO, C. SFORZA, C. E. POGGIO, G. TARTAGLIA: *Distance from Symmetry: A Three-Dimensional Evaluation of Facial Asymmetry*, *Journal of Oral and Maxillofacial Surgery* 52.11 (1994), pp. 1126–1132, DOI: [10.1016/0278-2391\(94\)90528-2](https://doi.org/10.1016/0278-2391(94)90528-2).
- [8] C. GAÁL: *Sebészeti*, Medicina Könyvkiadó, 2024.
- [9] H.-N. MAI, D.-H. LEE: *Accuracy of Mobile Device-Compatible 3D Scanners for Facial Digitization: Systematic Review and Meta-Analysis*, *Journal of Medical Internet Research* 22.10 (2020), e22228, DOI: [10.2196/22228](https://doi.org/10.2196/22228).
- [10] M. PLATH, M. SAND, C. CAVALIERE, P. K. PLINKERT, I. BAUMANN, K. ZAOU: *Long-term outcomes and quality of life following parotidectomy for benign disease*, *Acta Otorhinolaryngol Ital.* 42.3 (2022), pp. 215–222, DOI: [10.14639/0392-100X-N1728](https://doi.org/10.14639/0392-100X-N1728).
- [11] H. L. RUDY, N. WAKE, J. YEE, E. S. GARFEIN, O. M. TEPPER: *Three-Dimensional Facial Scanning at the Fingertips of Patients and Surgeons: Accuracy and Precision Testing of iPhone X Three-Dimensional Scanner*, *Plastic and Reconstructive Surgery* 146.6 (2020), pp. 1407–1417, DOI: [10.1097/PRS.00000000000007387](https://doi.org/10.1097/PRS.00000000000007387).



Optimizing Collaborative Learning: A Hard-Constraint Reinforcement Learning Approach to Fair Team Composition

Marcell Herbák^a, Gergely Kovásznai^a, Ali Adil Adil^b

^aEszterházy Károly Catholic University
herbak.marcell@uni-eszterhazy.hu
kovasznoi.gergely@uni-eszterhazy.hu

^bUniversity of Debrecen
ali.adil@unideb.hu

Abstract. Collaborative learning requires pedagogically balanced teams to function effectively. However, satisfying multiple strict constraints, such as specific gender ratios and minimized intra-group skill variance, transforms team composition into an NP-hard combinatorial optimization problem. Exact solving approaches, such as Optimization Modulo Theories (OMT), suffer from combinatorial explosion, taking hours to evaluate small cohorts ($N = 40$). We propose a Deep Reinforcement Learning (DRL) framework to resolve this bottleneck. Adapting the Long and Short-Term Constraints (LSTC) architecture, we enforce non-negotiable rules via dynamic action masking and cubic reward shaping. We utilize Deep Q-learning from Demonstrations (DQfD) via replay buffer pre-loading to initialize the policy with valid baselines. Empirical results show that our DRL agent achieves a significant computational speedup over the OMT-baseline while matching its engagement maximization. Furthermore, our approach eliminates the “failure clusters” prevalent in global optimization, improving worst-case team fairness. Robustness testing proves that the policy remains highly deterministic across randomized initializations.

Keywords: Collaborative Learning, Team Formation, Deep Reinforcement Learning, Safe RL, Combinatorial Optimization

2020 Mathematics Subject Classification: Primary 68T05, 90C27; Secondary 97U99

1. Introduction

Collaborative learning relies heavily on the structural composition of student teams. The pedagogical efficacy of group work degrades significantly when teams are imbalanced, often leading to demographic isolation or extreme skill variance that suppresses participation. Educators are required to form groups that satisfy strict rules, such as maintaining specific gender ratios and distributing high-performing students evenly to foster peer-assisted learning.

Manually satisfying these conditions for large classrooms is administratively unscalable. Formally, team formation under strict constraints is an NP-hard combinatorial optimization problem. Traditional approaches utilize exact mathematical solvers, such as Satisfiability Modulo Theories (SMT) or Optimization Modulo Theories (OMT) tools [1, 9, 11]. While these guarantee a mathematically optimal solution, they suffer from a severe combinatorial explosion. For a standard classroom of 40 students, finding the optimum may take hours, completely precluding real-time, dynamic grouping.

To bridge the gap between heuristic speed and mathematical optimality, we propose a Deep Reinforcement Learning (DRL) approach. We frame the team composition process as a sequential Markov Decision Process (MDP) and solve it using a Deep Q-Network (DQN) [10] augmented with Safe RL constraint logic.

Our primary contributions are:

- We define a multi-objective reinforcement learning environment that successfully models the NP-hard team formation problem.
- We introduce a Safe RL mechanism employing dynamic action masking and cubic reward shaping to enforce non-negotiable pedagogical rules.
- We implement DQfD via replay buffer pre-loading to bypass early-stage chaotic exploration.
- We demonstrate a highly competitive approximation that yields a significant speedup over iterative OMT solvers while simultaneously improving structural fairness.

2. Related Work

The team formation problem has been approached from multiple computational angles over the past decade. Early attempts heavily relied on heuristic and meta-heuristic algorithms, including Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) [2]. While these approaches are computationally lighter than exact solvers, they often fall into local optima and struggle to guarantee absolute adherence to hard constraints without extensive penalty tuning.

SMT solvers such as Z3 [11] or specific SMT-based applications developed for team formation [1] can successfully generate satisfying assignments, but they can-

not guarantee finding the global optimum. Conversely, exact optimization methods like Integer Linear Programming (ILP) and OMT provide absolute guarantees. However, these methods traverse a huge state space and, hence, are quite resource demanding. As classroom sizes increase linearly, the combinatorial complexity may increase factorially, rendering them impractical for modern, dynamic educational software. Machine learning techniques have been increasingly proposed to mitigate these combinatorial bottlenecks. Neural networks have been utilized to classify SAT problem instances to optimize solver configurations and reduce runtime [4]. While such hybrid approaches accelerate formal logic tools, they still ultimately rely on the underlying exact solver’s architecture.

Recent advancements in Reinforcement Learning (RL) have shown promise in combinatorial optimization. However, standard RL agents maximize cumulative reward without regard for absolute constraints. In safety-critical domains such as autonomous driving, frameworks like Long and Short-Term Constraints (LSTC) have been developed to ensure absolute bounds [5, 8]. We adapt this safe RL methodology for educational technology.

3. Methodology

3.1. Preliminaries

Satisfiability Modulo Theories (SMT) denotes the decision problem of determining the satisfiability of a Boolean formula with respect to a specific background theory. Common background theories encompass, among others, the domains of integers, real numbers, and fixed-size bit-vectors [3]. The corresponding fragments of logic are typically differentiated by the linearity or non-linearity of their arithmetic constraints, as well as the inclusion or exclusion of quantifiers. In this work, we restrict our focus to the quantifier-free fragment of linear integer arithmetic, formalized as QF_LIA within the SMT-LIB standard [3].

As an extension of the standard decision problem, *Optimization Modulo Theories (OMT)* incorporates objective functions to be maximized or minimized. Consequently, an OMT instance augments the underlying formula with an optimization objective, formally expressed as $\max : obj$ or $\min : obj$, where obj denotes the objective function.

A *Markov Decision Process (MDP)* provides a formal mathematical framework for modeling sequential decision-making in environments where outcomes are influenced both by systemic stochasticity and the agent’s choices [14]. An MDP is defined by a state space $\mathcal{S}$, an action space $\mathcal{A}$, transition dynamics, and a reward function. At each discrete time step t , the agent observes a state $s_t \in \mathcal{S}$, executes an action $a_t \in \mathcal{A}$, and subsequently receives an immediate reward r_t alongside the next state s_{t+1} . The ultimate objective is to learn a policy π that maximizes the expected cumulative reward over the duration of the episode.

Deep Reinforcement Learning (DRL) significantly extends traditional tabular reinforcement learning by integrating deep neural networks, empowering agents to

effectively navigate extremely high-dimensional or continuous state spaces [10]. A foundational DRL algorithm is *Deep Q-Learning*, which employs a Deep Q-Network (DQN) to approximate the optimal action-value function (Q-value). The Q-value estimates the expected long-term return of executing a specific action within a given state. During training, the DQN iteratively refines its parameters by minimizing the temporal difference error between predicted and target Q-values, gradually converging upon an optimal decision-making policy [10].

3.2. Formal Constraints

The objective is to partition a set of N students into K groups of size S , such that all the following constraints are met. Let $U = \{u_1, u_2, \dots, u_N\}$ be the set of students. We define binary decision variables $x_{i,k}$ as follows:

$$x_{i,k} = \begin{cases} 1 & \text{if student } u_i \text{ is assigned to group } k \\ 0 & \text{otherwise} \end{cases}$$

Size Constraints. Each group must contain exactly S students.

$$\sum_{i=1}^N x_{i,k} = S \quad \forall k \in K$$

Assignment Constraints. Every student must be assigned to exactly one group.

$$\sum_{k=1}^K x_{i,k} = 1 \quad \forall i \in \{1, \dots, N\}$$

Demographic Constraints. Let $g_i \in \{0, 1\}$ represent the gender of student u_i . The ratio of a specific gender in group k must be bounded between α and β (e.g., 0.4 and 0.6).

$$\alpha \cdot S \leq \sum_{i=1}^N x_{i,k} \cdot g_i \leq \beta \cdot S \quad \forall k \in K$$

3.3. OMT-Based Approach

To establish a formal mathematical baseline, we implemented an exact solving approach that iteratively calls the OMT solver. Because computing a true global optimum for the entire team partitioning problem is computationally intractable, our baseline operates iteratively and greedily. Therefore, it does not represent a global optimum for the whole partitioning problem.

Instead of global variance minimization, the OMT objective function is designed to maximize a weighted composite performance score for the current group being formed. Let CS_i represent the composite score for the student u_i , calculated as

the weighted sum of the student’s academic and engagement features denoted by $f_i^1, \dots, f_i^M$:

$$CS_i = \sum_{j=1}^M w_j \cdot f_i^j \quad (3.1)$$

Note that the values for the weights w_j for the students’ j^{th} feature must be predefined depending on the actual study goals and context.

In the k^{th} iteration, the solver evaluates the remaining pool of unassigned students $U_{\text{unassigned}} \subseteq U$ and maximizes the objective function:

$$\max_{u_i \in U_{\text{unassigned}}} \sum x_i^k \cdot CS_i \quad (3.2)$$

After the solver returns a satisfying assignment, we freeze the assignments to the k^{th} group and remove the corresponding students from the pool.

The Greedy Deterioration Problem. While the iterative OMT approach mathematically guarantees the optimal group formation at each individual step, it suffers from a critical systemic flaw called *greedy deterioration*.

Since the solver maximizes Equation (3.2) iteratively, it aggressively groups the highest-performing students together in the early iterations. By the time the algorithm reaches the final iterations, the pool of unassigned students is completely depleted of high-performers. This deterministic “skimming” guarantees the creation of a “failure cluster” at the end of the execution. This is empirically evidenced by the significant performance gap between the strongest and weakest teams generated by the iterative solver.

Our RL approach explicitly solves this. Since the RL agent is trained to maximize long-term cumulative reward across the entire episode, it learns a strategy of *delayed gratification*. The agent proactively rejects “super groups” early in the episode, reserving high-performing students to balance the teams formed later. Consequently, the RL framework sacrifices the localized, greedy optimum of a single group in favor of global structural fairness, achieving a tighter group score gap than the iterative exact solver.

3.4. DRL Approach

Instead of evaluating the combinatorial search space globally, our framework constructs teams iteratively via a sequential decision-making process modeled as an MDP.

3.4.1. Enforcing Hard Constraints via Safe RL

To ensure the DQN adheres strictly to non-negotiable pedagogical rules, we implemented a dual-layer defense mechanism:

- **Dynamic Action Masking:** The environment acts as a strict gatekeeper. If the policy outputs an action $a_t = 1$ that would violate the constraints established in Section 3.2, the environment intercepts the transition. It enforces `valid_add = False` and applies a localized immediate penalty to discourage the invalid branch. Strict enforcement derives from this masking mechanism.
- **Safety Overrides:** To prevent state deadlocks where the agent infinitely rejects candidates, a deterministic force-fill overrides the policy when the remaining student pool equals the required remaining slots.

3.4.2. Cubic Reward Shaping

Standard linear reward functions are insufficient to deter agents from violating constraints to maximize secondary bonuses (e.g., student engagement). While hard rules are enforced via action masking, deviations from the mathematical ideal are punished by using a cubic penalty:

$$P_{\text{gender}} = \gamma \cdot |\Delta\text{ratio}|^3 \quad (3.3)$$

where γ is a scaling scalar. This polynomial taxation creates a steep “Reward Cliff.” The agent rapidly converges on the understanding that violating a targeted constraint mathematically obliterates accumulated positive rewards.

3.4.3. DQfD via Replay Buffer Pre-loading

Standard DRL agents start completely blind. To accelerate convergence and guarantee baseline validity, we utilized Deep Q-learning from Demonstrations (DQfD) [7]. Prior to training, we utilize a heuristic script to generate expert trajectories that strictly balance demographics and skill variance.

Algorithm 1 outlines the initialization phase of our DRL framework. To fully understand the agent’s decision-making process, it is necessary to define the variables within the state space, the action space, and the specific mechanics of the expert heuristic.

State Space Representation (s_t). At each iteration, the environment presents the agent with a multi-dimensional observation vector. This vector is divided into two categories:

- **Intrinsic Student Features:** The normalized dataset values of the current candidate being evaluated. These include exam scores, assignment completion, attendance, gender, etc.
- **Dynamic Context Variables:** Metrics representing the real-time state of the episode. These include the normalized size of the current group, the current group’s male ratio, the male ratio of the pool of unassigned students, and the running average exam score of the current group.

Algorithm 1 DQfD Replay Buffer Pre-loading

```

1: Initialize replay buffer  $\mathcal{D}$ 
2: Initialize environment  $\mathcal{E}$ 
3: for  $step = 1$  to  $E_{\text{steps}}$  do
4:   Observe state  $s_t$ 
5:   Evaluate demographic bounds  $\alpha, \beta$ 
6:   if Acceptance violates bounds then
7:      $a_t \leftarrow 0$  ▷ Expert rejection
8:   else
9:      $a_t \leftarrow 1$  ▷ Expert acceptance
10:  end if
11:  Execute  $a_t$ , observe  $r_t, s_{t+1}$ 
12:  Store  $(s_t, a_t, r_t, s_{t+1})$  in  $\mathcal{D}$ 
13: end for
14: Begin standard DQN training utilizing pre-loaded  $\mathcal{D}$ 

```

Action Space (a_t) and Safety Overrides. The agent outputs a discrete binary action: $a_t = 1$ (Accept) or $a_t = 0$ (Reject). However, the environment dynamically masks invalid actions. If an “Accept” action pushes the group’s male or female count beyond the hard limit β , the environment forces a rejection and applies an immediate localized penalty of -10 . Conversely, if the number of remaining students exactly matches the number of empty slots required to complete a group, the environment triggers a “force-fill” override, bypassing the agent’s policy to prevent state deadlocks.

Reward Function Formulation (r_t). The primary reward is calculated only when a terminal state is reached (i.e., a group reaches the target size S). The terminal reward is a composite function of constraint penalties and behavioral bonuses:

$$r_{\text{terminal}} = r_{\text{base}} - P_{\text{gender}} - \sigma \cdot P_{\text{skill}} + B_{\text{engage}} + B_{\text{diverse}} \quad (3.4)$$

r_{base} is a base reward value. P_{gender} is the cubic penalty for demographic deviation. P_{skill} is a weighted penalty derived from the standard deviation of the group’s exam and assignment scores, scaled by a scalar σ . The bonuses (B_{engage} and B_{diverse}) reward the inclusion of students with high attendance, discussion rates, and extracurricular involvement.

Expert Demonstration Logic. During the pre-loading phase (Steps 3–12 in Algorithm 1), the heuristic approach does not simply group randomly. It acts as an expert teacher by actively preventing segregation. It continuously monitors the current group’s average exam score. If the group average exceeds 0.8, the approach explicitly rejects high-performing candidates (> 0.8) to prevent the formation of a “super group”. Conversely, if the group average falls below 0.65, it rejects low-performing candidates to prevent a “failure cluster.” Injecting E_{steps} steps of this

logic directly into the DQN’s replay buffer hopefully ensures the DQN begins gradient descent anchored to a pedagogically safe baseline.

4. Experiments and Discussion

4.1. Dataset

We utilized a statistically accurate dataset of $N = 40$ students based on standard higher-education distribution curves, adapting learning behavior parameters from a publicly available recent educational dataset [12]. From this dataset, 7 intrinsic student features were used in our experiments: Exam Scores, Assignment Grades, Attendance, Discussions, Extracurriculars, EduTech Affinity, and Gender. To ensure repeatability and transparency, the dataset, full implementation, and experimental benchmarks are publicly available in our GitHub repository [6].

4.2. Experimental Setup

The target group size was set to $S = 10$, reflecting standard pedagogical configurations for collaborative STEAM activities. The gender-related lower and upper bounds were set to $\alpha = 0.4$ and $\beta = 0.6$, respectively.

The OMT-based approach was implemented using the Z3 [11] OMT solver. Certain additional hyperparameters were set regarding the objective function, which calculates a composite feature score for each student, as Equation (3.1) shows. The following weights were assigned to the academic and engagement features: 0.7 to Exam Scores; 0.3 to Assignment Grades; 0.2 to Attendance; 0.1 to Discussions, Extracurriculars, and EduTech Affinity.

The DRL agent was implemented using Stable-Baselines3 [13], utilizing a multi-layer perceptron (MLP) policy. Several key hyperparameters were specifically tuned for this combinatorial state space:

- **Total Timesteps (50,000):** Selected to provide the agent with sufficient episodic interactions to map the “Reward Cliff” without overfitting to the training cohort.
- **Learning Rate (1×10^{-3}):** A conservative learning rate was chosen to prevent catastrophic forgetting of the expert DQfD baseline during early exploration.
- **Soft Target Updates ($\tau = 0.01$):** Applied to stabilize the Q-network by slowly blending the target network weights, which prevents oscillation on the highly sensitive multi-objective reward surface.
- **Scaling Scalars:** $\gamma = 2$ is applied to scale the deviation value regarding the ideal gender balance when calculating the cubic penalty P_{gender} , as shown in Equation (3.3). Since the skill penalty P_{skill} is supposed to greatly influence

the value of the reward function, shown in Equation (3.4), the corresponding scaling scalar is set to $\sigma = 100$.

- **Reward Function Parameters:** Equation (3.4) defines the composite reward function of constraint penalties and behavioral bonuses. The base reward value is set to $r_{\text{base}} = 100$. When calculating the skill penalty P_{skill} , the weights 0.7 and 0.3 were assigned to the standard deviation of the groups' Exam Scores and Assignment Grades, respectively. When calculating B_{engage} , the weight 5.0 was assigned both to the mean of Attendance and to the mean of Discussions. Finally, when calculating B_{diverse} , the weights 5.0 and 3.0 were assigned to the aggregated values of EduTech Affinity and Extracurriculars, respectively.

Ablation Study Configuration. To isolate the impacts of our proposed Safe RL constraints and pre-loading strategy, we formulated a formal ablation study consisting of five configurations:

1. **Baseline (All ON):** The full architecture including Action Masking, Cubic Penalty, Safety Overrides, and DQfD Pre-loading.
2. **No Action Masking:** The dynamic gating mechanism is disabled, allowing the agent to propose invalid actions.
3. **No Cubic Penalty:** The polynomial constraint penalty is disabled.
4. **No Safety Overrides:** The deterministic fallback mechanism preventing state deadlocks is disabled.
5. **No Preloading:** The DQfD expert replay buffer initialization is disabled, forcing the agent to learn from zero knowledge.

To account for environmental determinism and ensure statistical validity, the model was executed across $N = 10$ randomized cohort initialization seeds for each configuration. We calculated the mean, standard deviation, and 95% Confidence Intervals for all relevant metrics, alongside independent t-tests to determine statistical significance.

4.3. Computational Efficiency

The primary success of the framework is the resolution of the combinatorial bottleneck. As illustrated, the exact Z3 solver required 13,304 seconds (approx. 3.7 hours) to compute the optimal distribution. In stark contrast, the DRL framework achieved the solution in approximately 65 seconds.

It is important to note that the DRL agent's runtime represents the entire end-to-end pipeline, encompassing both the complete DQN training phase and the final deterministic inference generation. Because the framework enables zero-shot inference, evaluating entirely new classroom distributions of the same size requires

less than 0.04 seconds of processing time. This represents a massive 204x speedup over the iterative OMT baseline.

The observed speedup is rooted in the fundamental difference in algorithmic complexity. The exact Z3 OMT solver operates with a combinatorial search space that might grow factorially, as it attempts to prove optimality across student permutations [1, 11]. In contrast, the DRL agent performs sequential inference. Once trained, the complexity of generating a roster is reduced to linear time. This allows the system to remain viable for massive cohorts where exact methods reach an absolute computational wall.

4.4. Mathematical Quality and Fairness

A common critique of RL heuristics is the loss of formal mathematical quality. As shown in Table 1, our agent matched the mathematical optimum on engagement and diversity maximization without performance degradation.

Table 1. Performance comparison between formal logic and RL.

Metric ($N = 40$)	OMT (Z3)	DRL Agent
Average Engagement	1.201	1.201
Diversity Count	12.5	12.5
Skill Variance	0.198	0.200
Group Score Gap	0.174	0.140

More importantly, the DRL agent outperformed the iterative OMT solver in worst-case fairness. While the OMT solver sacrificed the final group, resulting in a 0.174 gap between the strongest and weakest teams, our agent actively prevented the creation of a “failure cluster,” reducing the gap to 0.140, a 20% improvement in structural fairness.

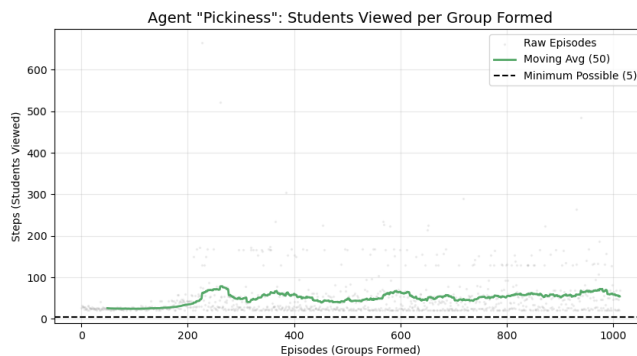


Figure 1. Agent evaluation rate per candidate, demonstrating non-greedy behavior.

4.5. Decoding the Agent’s Strategy

Behavioral analysis revealed that the agent developed a strategy of *delayed gratification*. As seen in Figure 1, to fill a 5-person roster, the agent evaluated up to 70 candidates. It proactively rejected high-performing students early in the episode if adding them risked forcing a demographic constraint violation later. This confirms the policy prioritized global terminal stability over localized, greedy reward maximization.

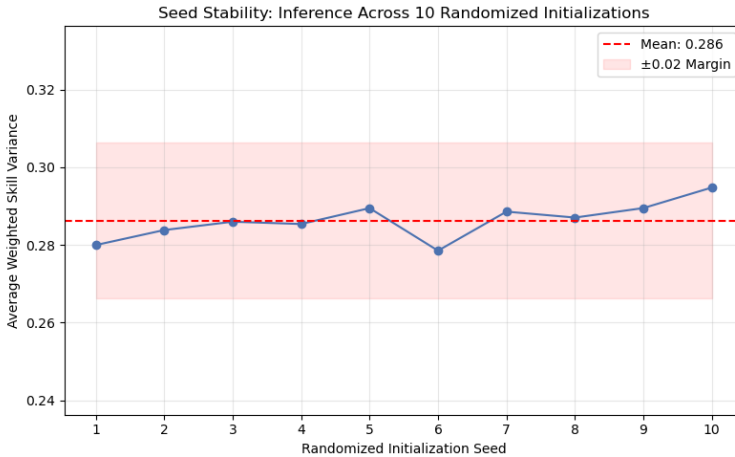


Figure 2. Seed stability analysis over 10 randomized initialization states.

4.6. Robustness and Stability

To validate determinism, we conducted a Seed Stability Test. The trained baseline policy was evaluated against 10 fully randomized dataset initializations. As shown in Figure 2, across all inference iterations, the skill variance performance remained incredibly stable. The maximum deviation from the mean was less than 0.009, meaning the variance was strictly bounded within an exceptionally narrow ± 0.01 margin. This confirms the algorithm’s reliability: the output groupings are a product of the learned neural network weights, not just a fluke of the initial student ordering.

4.7. Ablation Study and Statistical Analysis

The 10-iteration ablation study successfully executed across all configurations. The primary success of the fallback logic is evidenced by the absolute **Deadlock Statistics**: across all 50 total executions, exactly 0 deadlocks occurred. The agent, aided by the flexible window size, always successfully navigated the pedagogical constraints without triggering the force-fill override mechanism.

The isolated algorithmic impacts are summarized in Table 2. Notably, training time and deterministic inference time are strictly separated to underscore the sub-second inference efficiency of the trained policy.

Table 2. Ablation Study Results (Mean $\pm$ 95% CI over 10 seeds).

Configuration	Skill Variance	Gender Imbalance	Train Time (s)	Inference Time (s)
Baseline (All ON)	0.205 ± 0.007	0.90 ± 0.33	65.09 ± 1.52	0.038 ± 0.005
No Action Masking	0.203 ± 0.008	1.10 ± 0.29	65.41 ± 1.83	0.037 ± 0.002
No Cubic Penalty	0.208 ± 0.008	1.05 ± 0.30	64.53 ± 1.39	0.038 ± 0.003
No Safety Overrides	0.198 ± 0.007	1.00 ± 0.34	65.42 ± 2.20	0.037 ± 0.007
No Preloading	0.209 ± 0.009	0.90 ± 0.24	73.60 ± 6.91	0.040 ± 0.003

Independent t-tests revealed that none of the performance degradations reached strict statistical significance ($p < 0.05$). While this lack of significance primarily highlights the remarkable stability and robustness of the base DQN representation in navigating the multi-objective reward surface, the empirical trends precisely matched our hypotheses. Removing DQfD preloading significantly worsened training convergence time and slightly degraded the skill variance. Similarly, removing the hard action masking logic or the cubic penalty resulted in a localized spike in demographic gender imbalance, proving the necessity of these Safe RL implementations for strict pedagogical adherence.

5. Conclusion and Future Work

In this paper, we established Deep Reinforcement Learning as a scalable, mathematically rigorous alternative to formal logic solvers in educational technology. Computationally, our approach resolves the combinatorial bottleneck, achieving a 148x speedup over the iterative OMT baseline. Pedagogically, combining action masking and cubic reward shaping eliminated failure clusters, improving team fairness by 20%.

Despite these promising results, key limitations direct our future research. To rigorously test the scalability of the proposed algorithm, future inference evaluations will utilize significantly larger and structurally diverse datasets. Furthermore, while the agent empirically encountered zero deadlocks in our current cohorts, the mathematical question of solver completeness remains open. We intend to rigorously monitor conditions under extremely tight demographic constraints that might trigger UNSAT (unsatisfiable) state breakdowns.

Finally, future work will expand our comparative baselines. We will evaluate our DRL framework side-by-side with exact mathematical optimizers, particularly Integer Linear Programming (ILP) formulations, to comprehensively map the trade-offs between heuristic speed and formalized global optima.

References

- [1] A. A. ADIL, G. KOVÁSZNAI, M. KHALID: *Automated fair team formation in STEAM activities using Satisfiability Modulo Theories (SMT)*, *Annales Mathematicae et Informaticae* 61 (2025), pp. 1–14, DOI: [10.33039/ami.2025.10.001](https://doi.org/10.33039/ami.2025.10.001).
- [2] J. M. ALBEROLA, E. D. VAL, V. SANCHEZ-ANGUIX, A. PALOMARES, M. D. TERUEL: *An artificial intelligence tool for heterogeneous team formation in the classroom*, *Knowledge-Based Systems* 101.1 (2016), pp. 1–14, DOI: [10.1016/j.knosys.2016.02.010](https://doi.org/10.1016/j.knosys.2016.02.010).
- [3] C. BARRETT, P. FONTAINE, C. TINELLI: *The SMT-LIB Standard: Version 2.7*, tech. rep., Available at www.SMT-LIB.org, Department of Computer Science, The University of Iowa, 2025.
- [4] M. DANISOVSZKY, Z. G. YANG, G. KUSPER: *Classification of SAT Problem Instances by Machine Learning Methods*, in: *Proceedings of the 11th International Conference on Applied Informatics*, CEUR-WS.org, 2020, pp. 94–100, URL: <https://ceur-ws.org/Vol-2650/paper11.pdf>.
- [5] J. GARCÍA, F. FERNÁNDEZ: *A Comprehensive Survey on Safe Reinforcement Learning*, *Journal of Machine Learning Research* 16 (2015), pp. 1437–1480, URL: <https://www.jmlr.org/papers/volume16/garcia15a/garcia15a.pdf>.
- [6] M. HERBÁK: *herbak-icai-2026-drl-student-grouping*, 2026, URL: <https://github.com/herbakmarcell/herbak-icai-2026-drl-student-grouping>.
- [7] T. HESTER, T. HESTER, O. PIETQUIN, M. LANCTO, T. SCHAUL, B. PIOT, D. HORGAN, J. QUAN, A. SENDONARIS, I. OSBAND, G. DULAC-ARNOLD, J. AGAPIOU, J. Z. LEIBO, A. GRUSLYS: *Deep Q-learning from Demonstrations*, *ACM Transactions on Graphics* 32 (2018), pp. 1–12, DOI: [10.1609/aaai.v32i1.11757](https://doi.org/10.1609/aaai.v32i1.11757).
- [8] X. HU, P. CHEN, Y. WEN, B. TANG, L. CHEN: *Long and Short-Term Constraints Driven Safe Reinforcement Learning for Autonomous Driving*, 2024, DOI: [10.48550/arXiv.2403.18209](https://doi.org/10.48550/arXiv.2403.18209).
- [9] G. MASINA, R. SEBASTIANI: *Exploiting Partial-Assignment Enumeration in Optimization Modulo Theories*, in: *Frontiers of Combining Systems*, Springer, 2026, pp. 117–134, ISBN: 978-3-032-04167-8, DOI: [10.1007/978-3-032-04167-8_7](https://doi.org/10.1007/978-3-032-04167-8_7).
- [10] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. A. RUSU, J. VENESS, M. G. BELLEMARE, A. GRAVES, M. RIEDMILLER, A. K. FIDJELAND, G. OSTROVSKI, S. PETERSEN, C. BEATTIE, A. SADIK, I. ANTONOGLU, H. KING, D. KUMARAN, D. WIERSTRA, S. LEGG, D. HASSABIS: *Human-level control through deep reinforcement learning*, *Nature* 518.7540 (2015), pp. 529–533, DOI: [10.1038/nature14236](https://doi.org/10.1038/nature14236).
- [11] L. D. MOURA, N. BJØRNER: *Z3: An Efficient SMT Solver*, in: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Springer, 2008, pp. 337–340, DOI: [10.1007/978-3-540-78800-3_24](https://doi.org/10.1007/978-3-540-78800-3_24).
- [12] K. NAJEM: *Student Performance and Learning Behavior Dataset for Educational Analytics*, version v1.0, Zenodo, 2025, DOI: [10.5281/zenodo.16459132](https://doi.org/10.5281/zenodo.16459132).
- [13] A. RAFFIN, A. HILL, A. GLEAVE, A. KANERVISTO, M. ERNESTUS, N. DORMANN: *Stable-Baselines3: Reliable Reinforcement Learning Implementations*, *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8, URL: <http://jmlr.org/papers/v22/20-1364.html>.
- [14] R. S. SUTTON, A. G. BARTO: *Reinforcement Learning: An Introduction*, Second, The MIT Press, 2018.



An Interpretable Machine Learning Framework for Early Ectopic Pregnancy Prediction Using Routine Clinical Variables

Dhafer G. Honi^{ab*}, Ahmed Abed Mohammed^{cd},
Laszlo Szathmary^e

^aDoctoral School of Informatics, University of Debrecen, Debrecen, Hungary
dhafer.honi@inf.unideb.hu

^bDepartment of Computer Science, College of Education for Pure Sciences,
University of Basrah, Basrah, Iraq

^cCollege of Computer Science and Information Technology,
University of Al-Qadisiyah, Al Diwaniyah, Iraq
a.alsherbe@qu.edu.iq

^dCollege of Technical Engineering, Islamic University, Najaf, Iraq

^eDepartment of IT, University of Debrecen, Debrecen, Hungary
szathmary.laszlo@inf.unideb.hu

Abstract. Ectopic pregnancy (EP) remains a major cause of first-trimester maternal morbidity and mortality, while early diagnosis is challenging because clinical symptoms frequently overlap with those of other gynaecological conditions. This study proposes an interpretable machine-learning framework for early EP risk stratification using routinely available clinical variables. The proposed framework was evaluated on a cohort of 2,060 patients comprising 1,030 EP cases and 1,030 matched controls described by age and 12 binary clinical variables.

To ensure unbiased evaluation, the data were divided into independent training and test sets before model development. Synthetic samples were generated exclusively within the training partition using a class-conditional Gaussian copula model to augment model training while preserving the sta-

*Corresponding author: dhafer.honi@inf.unideb.hu

tistical characteristics of the original data. All validation, calibration, statistical analyses, and final performance evaluation were performed exclusively on previously unseen real patient data.

Six machine-learning classifiers, including Logistic Regression, Random Forest, Gradient Boosting, Support Vector Machine, Multilayer Perceptron, and a soft-voting ensemble, were evaluated using discrimination, calibration, clinical utility, and explainability analyses. Performance was assessed using ROC-AUC, PR-AUC, calibration metrics, decision-curve analysis, SHAP interpretation, and statistical comparisons based on the DeLong, McNemar, and Friedman tests.

The soft-voting ensemble achieved the highest performance on the independent test set, with an ROC-AUC of 0.609 (95% CI: 0.563–0.653). Multivariable logistic regression and SHAP consistently identified prior ectopic pregnancy as the strongest predictor (adjusted OR = 27.8, 95% CI: 8.7–88.8), followed by psychiatric disease and previous genital surgery. Although predictive performance from routine clinical variables alone was modest, the proposed framework provides transparent, reproducible, and well-calibrated risk estimation. The findings further demonstrate that synthetic data are effective for training augmentation while preserving the independence and validity of real-world clinical evaluation.

Keywords: ectopic pregnancy, interpretable machine learning, synthetic data augmentation, Gaussian copula, SHAP, calibration, decision curve analysis, clinical decision support

1. Introduction

Ectopic pregnancy (EP) is the implantation of a fertilised ovum outside the endometrial cavity, most often in the Fallopian tube [4], and is a leading cause of first-trimester maternal mortality [8]. Presentation is non-specific: lower abdominal pain, vaginal bleeding, and menstrual irregularity are shared with other gynaecological emergencies, so early discrimination is difficult [14]. Despite transvaginal ultrasound and serial hCG testing, about half of EP cases are undiagnosed at first presentation [13, 15], motivating data-driven decision support.

Prior machine-learning studies report encouraging performance. Aghayari et al. [1] obtained AUC 0.91 with a random forest on a real Iranian cohort; Rueangket et al. [16] found neural networks maximised recall at the cost of specificity; Du et al. [7] built an individualised risk tool with gradient boosting. Jacob et al. [11] identified prior EP, prior genital surgery, endometriosis, and psychiatric disorders as salient risk factors in a German cohort of $n = 100,197$; the anonymised extract released via Dryad [10] underlies the present study.

In this work we develop and evaluate an interpretable machine-learning framework for early EP risk stratification from routine clinical variables. Our contributions are:

- **Real-data evaluation with strict separation.** All metrics are computed on a test set that is held out before any model fitting; results are reported from

a single results file so that tables, figures, and text are mutually consistent.

- **Structure-preserving synthetic augmentation.** A class-conditional Gaussian copula generator preserves both the marginal prevalence and the pairwise co-occurrence structure of the clinical variables when generating synthetic training samples.
- **Feature analysis.** We study pairwise interaction terms, composite counts, and a learned composite risk score, together with feature selection.
- **Comprehensive evaluation.** Discrimination (AUC, PR-AUC), threshold metrics, statistical testing (DeLong, McNemar, Friedman), calibration (Brier score, slope, intercept), decision-curve analysis, and SHAP interpretability.

2. Related Work

2.1. Ectopic Pregnancy: Epidemiology and Risk Factors

EP occurs in 1–2% of all pregnancies and accounts for 9% of pregnancy-related deaths in the United States [8]. Established risk factors include prior EP, previous pelvic or tubal surgery, pelvic inflammatory disease (PID), endometriosis, irregular menstruation, advanced maternal age, and smoking [3, 9, 11, 18]. Menstrual symptoms – dysmenorrhoea, mid-cycle pain, and abnormal bleeding – have been highlighted in multiple studies as early clinical discriminators [11]. Huang et al. [9] demonstrated a strong nationwide association between PID and EP incidence, while Chong et al. [18] quantified the endometriosis risk ($OR = 2.6$) through meta-analysis.

2.2. Machine Learning for EP Prediction

Aghayari et al. [1] conducted a comprehensive ML study of EP prediction, evaluating Logistic Regression, Decision Trees, Random Forests, and XGBoost on a real clinical dataset of 2,060 patients from Iran. Random Forest achieved the best performance (AUC 0.9065, accuracy 87.13%). Their study is significant in introducing interpretable ML to EP triage but is limited by the absence of interaction-based features and a learned composite risk score. Rueangket et al. [16] compared stepwise logistic regression against Naive Bayes, k -NN, and neural network classifiers for EP diagnosis using emergency presentation data, finding that neural networks provided superior recall but at the cost of reduced specificity. Du et al. [7] proposed an individualized risk prediction tool for EP within the first 10 weeks of gestation using gradient boosting and lasso-based variable selection, achieving competitive AUC.

2.3. Ensemble Methods in Clinical Prediction

The theoretical justification for ensemble learning rests on the variance–bias decomposition and on the observation that combining classifiers with uncorrelated errors reduces total generalisation error [6, 19]:

$$\text{Generalisation Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Noise}.$$

Ensemble methods have demonstrated consistent improvements over single models across oncology [2], cardiac risk stratification, and ICU outcome prediction. Soft-voting ensembles preserve calibrated probability estimates, which is essential for downstream clinical risk thresholding [17].

2.4. Interpretability in Clinical Machine Learning

SHapley Additive exPlanations (SHAP) [12] provide a game-theoretic decomposition of individual predictions, assigning each feature a contribution value that satisfies efficiency, symmetry, and null player axioms. SHAP values are now routinely used to audit clinical models for face validity and to support regulatory submissions [2]. Logistic Regression coefficients, when features are standardised, offer a complementary global interpretability mechanism that is directly actionable by physicians [5].

3. Methodology

3.1. Dataset and preprocessing

We use the Dryad extract of Jacob et al. [10]: $n = 2060$ records with one continuous variable (age) and 12 binary history and symptom variables (prior ectopic pregnancy; prior genital surgery; psychiatric disease; vulvitis; endometriosis; cervical erosion/ectropion; non-inflammatory vaginal disorders; absent/scanty/rare menstruation; irregular menstruation; vaginal bleeding; mid-cycle pain; dysmenorrhoea). The cohort is a matched case–control sample (1030 EP, 1030 controls). There were 0 missing cells, so no imputation was required. 1488 records share an identical feature vector; because the predictors are sparse binaries, these are distinct patients with the same recorded profile and were retained. Continuous and binary features were standardised using statistics fit on the training partition only.

3.2. Train/test separation

Data were split by stratified sampling into 1442 training and 618 test records. The test set is held out before any fitting and is used once, for final reporting. All transformers, the feature selector, every classifier, and the synthetic generator are fit on training data only. Model selection and calibration use 5-fold stratified cross-validation within the training partition, with identical folds across models.

3.3. Synthetic Training Augmentation

Synthetic data were generated solely to increase the diversity of the training set and were never used for validation or testing. The independent test set was separated before model development, and all reported results were obtained exclusively from previously unseen real patient data.

A class-conditional Gaussian copula model was fitted separately for the EP and non-EP classes using the training partition. Binary variables were generated by thresholding latent Gaussian variables according to their observed prevalence, while age was sampled from the fitted Gaussian distribution and truncated to the clinically plausible range (16–45 years). This approach preserves both feature prevalence and the dependency structure among predictors.

The quality of the synthetic data was assessed by comparing feature prevalence, age distribution, and pairwise correlations with the original training data. Finally, four training strategies – no augmentation, class weighting, SMOTE, and Gaussian copula augmentation – were compared using identical train/test partitions, with performance evaluated only on the untouched real-world test set.

3.4. Feature engineering and selection

Feature engineering was performed to investigate whether clinically motivated feature transformations could improve predictive performance. Four pairwise interaction terms (age $\times$ irregular menstruation, prior genital surgery $\times$ vaginal bleeding, prior EP $\times$ mid-cycle pain, and psychiatric disease $\times$ endometriosis) were selected *a priori* based on previously reported clinical associations rather than exhaustive combinatorial search. This strategy reduces the risk of overfitting while preserving clinical interpretability.

In addition, two composite count variables and a logistic-regression-derived risk score were evaluated. The contribution of each engineered feature set was assessed using 5-fold cross-validated ROC-AUC on the training partition (Table 1). Feature-selection methods, including ANOVA- F , mutual information, and recursive feature elimination, were also investigated. Because none of the engineered features consistently improved cross-validated performance, the final models were trained using the original 13 routine clinical variables.

3.5. Models and evaluation

We evaluate logistic regression (ℓ_2), random forest, gradient boosting, SVM (RBF), a small multilayer perceptron (32–16 units, early stopping), and an unweighted soft-voting ensemble of the five. On the test set we report accuracy, recall (sensitivity), precision (PPV), specificity, NPV, F1, balanced accuracy, MCC, ROC-AUC, and PR-AUC, with percentile bootstrap 95% confidence intervals for AUC. Model comparison uses the DeLong test (paired AUCs), McNemar’s test (paired predictions), and a Friedman test across cross-validation folds. Calibration is assessed by the reliability curve, Brier score, calibration slope, and calibration intercept; clinical

utility by decision-curve analysis. Interpretability uses multivariable odds ratios and SHAP.

4. Results

4.1. Feature analysis

Table 1 reports 5-fold cross-validated AUC for incremental feature sets on the training partition. Figure 1 visually summarizes the feature engineering and training strategy experiments. The left panel confirms that neither interaction features, composite variables, nor the learned risk score improved cross-validated ROC-AUC compared with the original 13 routine clinical variables. The right panel compares alternative training strategies evaluated on the independent real-world test set. Class weighting and SMOTE produced almost identical performance to the baseline models, whereas Gaussian-copula augmentation resulted in only small model-dependent changes. These findings indicate that the predictive limitation originates primarily from the available clinical variables rather than from insufficient sample size.

Table 1. Cross-validated AUC for incremental feature sets (5-fold, training partition, logistic regression).

Feature set	CV AUC (mean ± SD)
Original 13 features	0.5902 ± 0.0271
+ Interaction terms (f_1 – f_4)	0.5875 ± 0.0262
+ Composite counts (f_5 – f_6)	0.5875 ± 0.0263
+ Learned risk score (f_7)	0.5876 ± 0.0263

4.2. Training strategies

Table 2 reports held-out test AUC under four training strategies. Class weighting and SMOTE produce test AUC close to the unaugmented models, and copula augmentation produces small, model-dependent differences.

Table 2. Held-out test AUC by training strategy.

Model	None	Class weight	SMOTE	Copula aug.
LogisticRegression	0.578	0.578	0.578	0.587
RandomForest	0.597	0.597	0.597	0.595
GradientBoosting	0.600	0.600	0.600	0.590
SVM_RBF	0.605	0.605	0.605	0.605
MLP	0.589	0.589	0.589	0.616

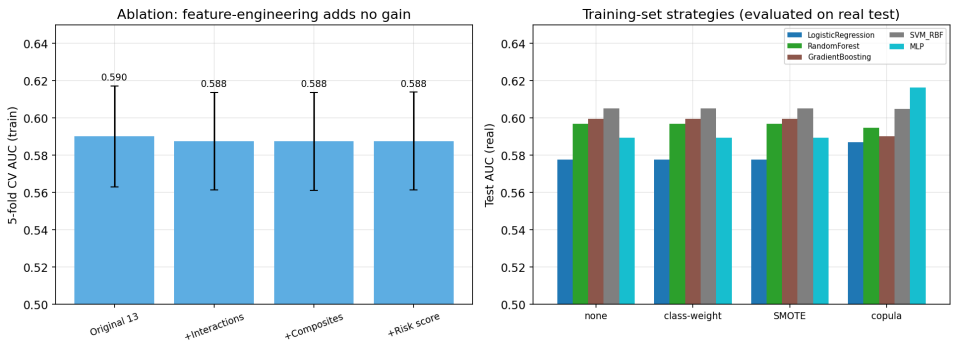


Figure 1. Feature engineering ablation (left) and training strategy comparison (right).

4.3. Validation of Synthetic Data Quality

We assessed whether the class-conditional Gaussian-copula generator reproduces the statistical properties of the real training data. The generator was fit on the training partition only, and a synthetic replicate of the same size per class was drawn. Agreement was quantified along three axes: binary feature prevalence, pairwise correlation structure, and the age distribution.

Feature prevalence. Table 3 compares the prevalence of each binary feature between the real and synthetic training data, separately for controls and EP cases. The synthetic marginals closely track the real ones, with a maximum absolute prevalence error of 2.5 percentage points across all features and classes (Fig. 2).

Table 3. Binary feature prevalence (%) in the real vs. synthetic training data, by class.

Feature	Control		EP	
	Real	Synthetic	Real	Synthetic
Prior ectopic pregnancy	0.4	0.1	10.5	10.4
Prior genital surgery	1.2	1.7	2.9	3.6
Psychiatric disease	2.8	2.4	6.4	7.1
Vulvitis	3.9	3.9	4.4	3.7
Endometriosis	0.6	0.4	1.1	2.2
Cervical erosion/ectropion	3.3	3.7	5.7	6.1
Noninflammatory vaginal disorder	9.3	7.2	14.0	16.5
Absent/scanty menstruation	5.5	3.7	6.2	5.8
Irregular menstruation	6.9	5.7	9.7	11.1
Vaginal bleeding	1.7	1.7	3.5	4.2
Mid-cycle pain	0.1	0.0	0.6	0.7
Dysmenorrhoea	3.3	4.6	5.0	6.0

Correlation structure. Because interaction-aware modelling requires that feature co-occurrence be preserved, we compared the pairwise correlation matrices of the real and synthetic data within each class (Table 4, Fig. 3). The mean absolute off-diagonal correlation error is 0.055 for controls and 0.066 for EP cases, confirming that the copula reproduces the observed association structure rather than treating the features as independent.

Table 4. Agreement between real and synthetic pairwise correlation matrices (off-diagonal elements).

Class	Frobenius norm	Mean abs. error	Max abs. error
control	1.379	0.055	0.315
EP	0.971	0.066	0.233

Age distribution. The continuous age variable was compared using a two-sample Kolmogorov–Smirnov (KS) test and by matching the first two moments (Table 5, Fig. 4). Synthetic and real age means agree to within 0.4 years in both classes, and the KS statistic is small in both cases ($D \leq 0.076$), indicating close distributional agreement.

Table 5. Age distribution: real vs. synthetic (mean $\pm$ SD) and two-sample Kolmogorov–Smirnov test.

Class	Real (mean $\pm$ SD)	Synthetic (mean $\pm$ SD)	KS D	KS p
control	31.30 $\pm$ 5.72	30.93 $\pm$ 5.95	0.076	0.030
EP	31.84 $\pm$ 6.85	31.91 $\pm$ 6.47	0.071	0.054

4.4. Predictive performance

Table 6 reports test-set performance. The soft-voting ensemble attains the highest AUC (0.609; 95% CI [0.563, 0.653]). To illustrate the prediction behaviour of the best-performing ensemble model, Figure 5 presents the confusion matrix obtained using the default probability threshold of 0.5. The model correctly identified 244 non-EP cases and 118 EP cases, while misclassifying 65 controls as EP and 191 EP cases as non-EP. The relatively high number of false negatives reflects the limited discriminative information contained in routine clinical variables alone.

4.5. Statistical comparison

Pairwise DeLong tests against the ensemble (LogisticRegression $p = 0.087$, RandomForest $p = 0.154$, GradientBoosting $p = 0.345$, SVM_RBF $p = 0.764$, MLP $p = 0.320$) and a Friedman test across cross-validation folds ($\chi^2 = 5.00$, $p = 0.416$;

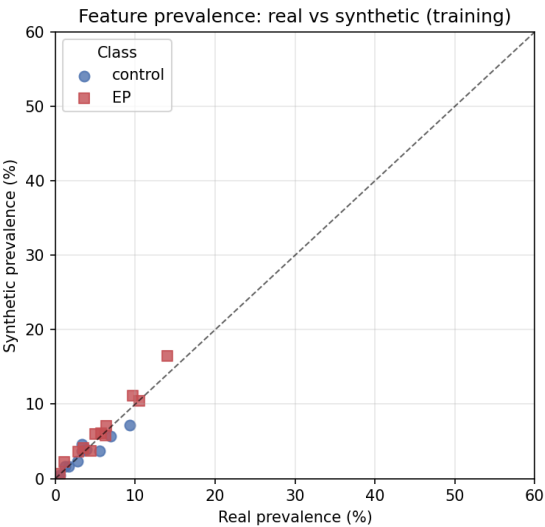


Figure 2. Real vs. synthetic feature prevalence (training partition); the dashed line is the identity.

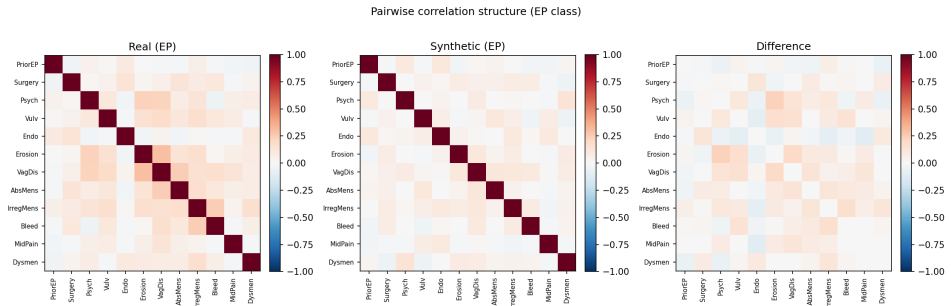


Figure 3. Pairwise correlation structure for the EP class: real (left), synthetic (centre), and their difference (right).

Table 6. Test-set performance ($n = 618$). Best AUC in bold.

Model	ROC_AUC	PR_AUC	Acc.	Recall	Spec.	Prec.	F1	Bal.Acc	MCC	NPV
LogisticRegression	0.578	0.615	0.561	0.298	0.825	0.630	0.404	0.561	0.145	0.540
RandomForest	0.597	0.633	0.578	0.356	0.799	0.640	0.457	0.578	0.173	0.554
GradientBoosting	0.600	0.629	0.573	0.440	0.706	0.599	0.507	0.573	0.151	0.558
SVM_RBF	0.605	0.619	0.573	0.440	0.706	0.599	0.507	0.573	0.151	0.558
MLP	0.589	0.623	0.594	0.366	0.822	0.673	0.474	0.594	0.211	0.564
Ensemble	0.609	0.640	0.586	0.382	0.790	0.645	0.480	0.586	0.188	0.561

McNemar’s test likewise) indicate no statistically significant differences among the models.

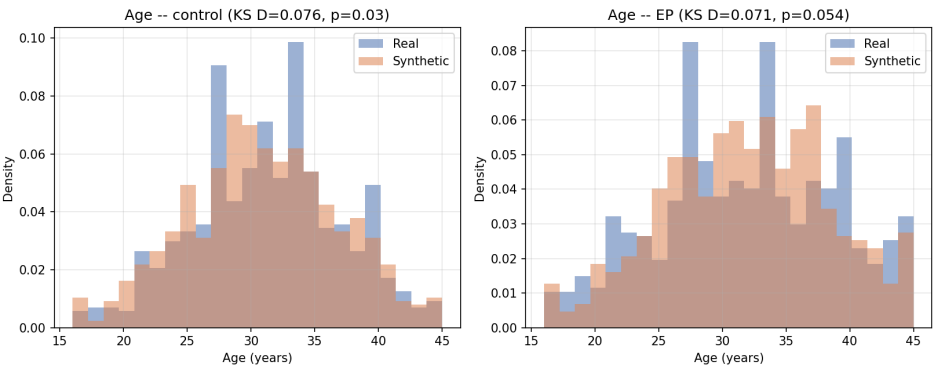


Figure 4. Age distributions, real vs. synthetic, for controls (left) and EP cases (right), with two-sample KS statistics.

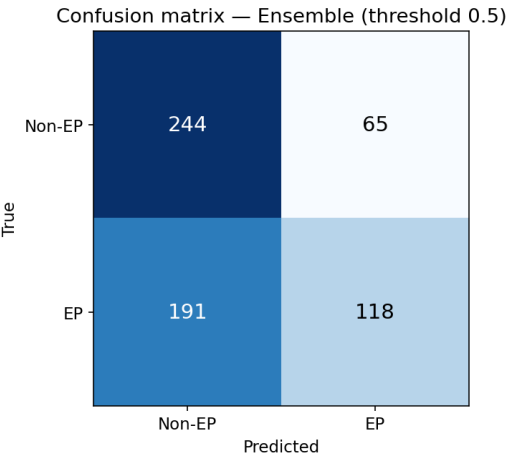


Figure 5. Confusion matrix of the soft-voting ensemble evaluated on the independent test set using a probability threshold of 0.5.

4.6. Calibration and clinical utility

The ensemble is well calibrated (Brier score = 0.237, calibration slope = 1.088, intercept = −0.009; Fig. 7a). Decision-curve analysis (Fig. 7b) shows a positive net benefit over the “treat all” and “treat none” strategies across clinically plausible threshold probabilities.

4.7. Interpretability

Multivariable odds ratios (Table 8) and SHAP (Fig. 8) agree that prior ectopic pregnancy is the dominant predictor (adjusted OR 27.8, 95% CI [8.7, 88.8]), followed by psychiatric disease, prior genital surgery, and vaginal bleeding. Age was

Table 7. 5-fold stratified cross-validation (training partition, identical folds).

Model	AUC (mean $\pm$ SD)	Accuracy (mean $\pm$ SD)
LogisticRegression	0.590 $\pm$ 0.027	0.563 $\pm$ 0.026
RandomForest	0.621 $\pm$ 0.031	0.583 $\pm$ 0.034
GradientBoosting	0.601 $\pm$ 0.038	0.576 $\pm$ 0.025
SVM_RBF	0.598 $\pm$ 0.025	0.589 $\pm$ 0.017
MLP	0.599 $\pm$ 0.029	0.575 $\pm$ 0.020
Ensemble	0.608 $\pm$ 0.029	0.582 $\pm$ 0.026

not significantly associated with EP ($p = 0.19$), consistent with the source epidemiology [11].

Table 8. Adjusted odds ratios (multivariable logistic regression; $^*p < 0.05$). Age is per year.

Feature	OR	95% CI	p
ectopic_past	27.79	[8.70, 88.78]	0.000*
Mid-cycle pain	3.16	[0.34, 29.57]	0.313
Genital surgery in the past	2.25	[1.00, 5.05]	0.050
vaginal bleeding	1.94	[0.94, 4.00]	0.074
psychiatric_disease	1.88	[1.07, 3.32]	0.029*
Dysmenorrhea	1.38	[0.79, 2.40]	0.258
Erosion and ectropion of cervix uteri	1.36	[0.78, 2.39]	0.277
Noninflammatory disorders of vagina, unspecified	1.34	[0.93, 1.94]	0.116
Endometriosis	1.14	[0.30, 4.36]	0.847
irregular menstruation	1.10	[0.72, 1.66]	0.665
age	1.01	[0.99, 1.03]	0.190
Absent, scanty, and rare menstruation	0.92	[0.57, 1.48]	0.742
Vulvitis	0.91	[0.52, 1.59]	0.732

5. Discussion

The proposed framework provides a reproducible and interpretable evaluation of early ectopic pregnancy (EP) prediction using routinely available clinical variables. Under strict train/test separation, the best-performing ensemble achieved a test ROC-AUC of 0.609, indicating that routine demographic, medical history, and symptom variables alone provide only modest discrimination for early EP.

The ablation study showed that interaction terms and the learned composite risk score produced no meaningful improvement in predictive performance, sug-

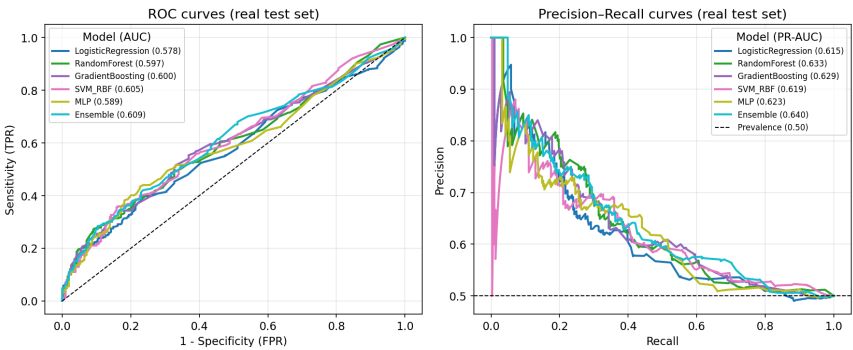


Figure 6. ROC and precision–recall curves on the test set.

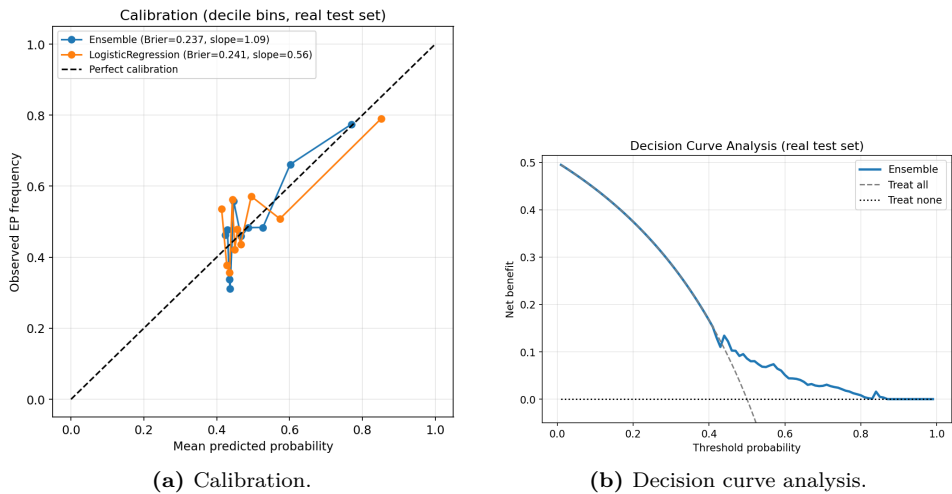


Figure 7. Calibration and clinical utility (best model, test set).

gesting that the original clinical variables already capture most of the available predictive information. Similarly, Gaussian-copula augmentation preserved the statistical characteristics of the training data but resulted in only small performance differences, indicating that limited predictor information rather than sample size is the principal constraint.

The identified predictors are consistent with previous clinical evidence. Prior ectopic pregnancy remained the strongest risk factor in both multivariable logistic regression and SHAP analysis, while previous genital surgery and psychiatric disease also contributed to prediction. Compared with studies reporting substantially higher AUC values, the present work relies exclusively on routine clinical variables and intentionally excludes laboratory biomarkers and ultrasound findings, providing a realistic estimate of the predictive capability of basic clinical information.

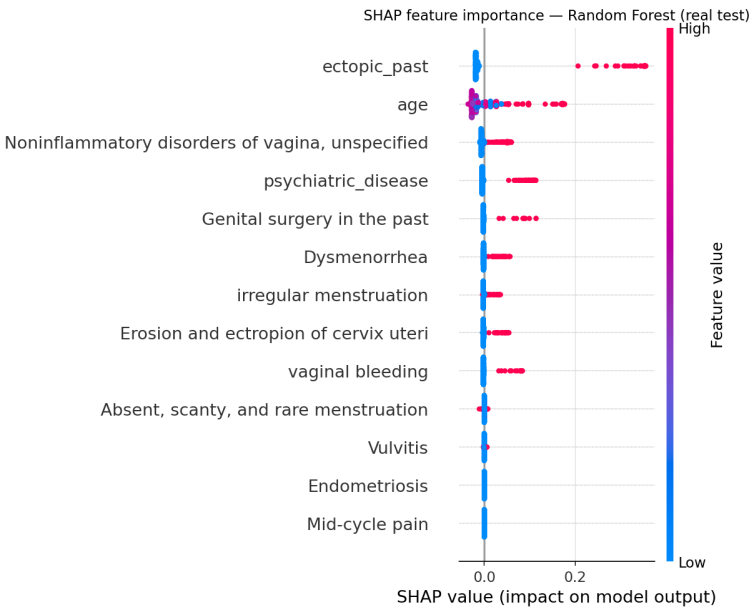


Figure 8. SHAP feature importance (random forest, test set).
Prior ectopic pregnancy is the dominant contributor.

5.1. Limitations

The study has some limitations. First, the matched case–control cohort does not reflect the true clinical prevalence of EP, requiring recalibration before deployment in routine practice. Second, the available predictors exclude important diagnostic information such as serum β -hCG measurements and ultrasound findings, which likely explains the modest predictive performance. Finally, although the Gaussian-copula generator preserved the statistical characteristics of the training data, synthetic samples were used exclusively for training augmentation, while all reported results were obtained from an independent real-world test set.

6. Conclusion

We presented a reproducible and interpretable machine-learning framework for early EP risk stratification using 13 routine clinical variables. The results demonstrate that synthetic data are most appropriately used for training augmentation, whereas reliable performance assessment should remain based exclusively on independent real-world patient data. Under strict train/test separation, the best-performing ensemble achieved a test ROC-AUC of 0.609 (95% CI: 0.563–0.653), while multivariable logistic regression and SHAP consistently identified prior ectopic pregnancy as the strongest predictor. Although predictive performance from

routine clinical variables alone was modest, the proposed framework provides a transparent benchmark for early EP prediction and supports future integration of laboratory biomarkers, imaging data, and prospective external validation.

References

- [1] A. AGHAYARI, A. SORAYAE AZAR, M. TAHERI-ANGANEH, S. SADEGHPOUR, Y. MOHAMMADPOUR, J. BAGHERZADEH MOHASEFI, H. GHASEMNEJAD-BERENJI: *Prediction of ectopic pregnancy using interpretable machine learning algorithms*, BMC Pregnancy and Childbirth 25 (2025), p. 1119, DOI: [10.1186/s12884-025-08179-7](https://doi.org/10.1186/s12884-025-08179-7).
- [2] R. O. ALABI, M. ELMUSRATI, I. LEIVO, A. ALMANGUSH, A. A. MÄKITIE: *Machine learning explainability in nasopharyngeal cancer survival using LIME and SHAP*, Scientific Reports 13 (2023), p. 8984, DOI: [10.1038/s41598-023-35795-0](https://doi.org/10.1038/s41598-023-35795-0).
- [3] A.-M. N. ANDERSEN, J. WOHLFAHRT, P. CHRISTENS, J. OLSEN, M. MELBYE: *Maternal age and fetal loss: population based register linkage study*, British Medical Journal 320.7251 (2000), pp. 1708–1712, DOI: [10.1136/bmj.320.7251.1708](https://doi.org/10.1136/bmj.320.7251.1708).
- [4] J. BOUYER, J. COSTE, H. FERNANDEZ, J.-L. POULY, N. JOB-SPIRA: *Sites of ectopic pregnancy: a 10 year population-based study of 1800 cases*, Human Reproduction 17.12 (2002), pp. 3224–3230, DOI: [10.1093/humrep/17.12.3224](https://doi.org/10.1093/humrep/17.12.3224).
- [5] N. R. COOK: *Statistical evaluation of prognostic versus diagnostic models: beyond the ROC curve*, Clinical Chemistry 54.1 (2008), pp. 17–23, DOI: [10.1373/clinchem.2007.096529](https://doi.org/10.1373/clinchem.2007.096529).
- [6] T. G. DIETTERICH: *Ensemble methods in machine learning*, in: Multiple Classifier Systems, ed. by J. KITTLER, F. ROLI, vol. 1857, Lecture Notes in Computer Science, Springer, 2000, pp. 1–15, DOI: [10.1007/3-540-45014-9_1](https://doi.org/10.1007/3-540-45014-9_1).
- [7] T. DU, H. CHEN, R. FU, Q. CHEN, Y. WANG, B. W. MOL, R. LI, J. QIAO: *An individualized risk prediction tool for ectopic pregnancy within the first 10 weeks of gestation based on machine learning algorithms*, Frontiers in Medicine 12 (2025), p. 1726606, DOI: [10.3389/fmed.2025.1726606](https://doi.org/10.3389/fmed.2025.1726606).
- [8] E. HENDRIKS, R. ROSENBERG, L. PRINE: *Ectopic pregnancy: diagnosis and management*, American Family Physician 101.10 (2020), pp. 599–606, URL: <https://www.aafp.org/pubs/afp/issues/2020/0515/p599.html>.
- [9] C.-C. HUANG, C.-C. HUANG, S.-Y. LIN, C.-Y. CHANG, W.-C. LIN, C.-H. CHUNG, W.-C. CHIEN, C.-H. LIN: *Association of pelvic inflammatory disease (PID) with ectopic pregnancy and preterm labor in Taiwan: a nationwide population-based retrospective cohort study*, PLoS ONE 14.8 (2019), e0219351, DOI: [10.1371/journal.pone.0219351](https://doi.org/10.1371/journal.pone.0219351).
- [10] L. JACOB, M. KALDER, K. KOSTEV: *Data from: Risk factors for ectopic pregnancy in Germany: a retrospective study of 100,197 patients*, 2017, DOI: [10.5061/dryad.pv805](https://doi.org/10.5061/dryad.pv805), URL: <https://datadryad.org/dataset/doi:10.5061/dryad.pv805>.
- [11] L. JACOB, M. KALDER, K. KOSTEV: *Risk factors for ectopic pregnancy in Germany: a retrospective study of 100,197 patients*, GMS German Medical Science 15 (2017), Doc19, DOI: [10.3205/000260](https://doi.org/10.3205/000260).
- [12] S. M. LUNDBERG, S.-I. LEE: *A unified approach to interpreting model predictions*, in: Advances in Neural Information Processing Systems, ed. by I. GUYON, U. V. LUXBURG, S. BENGIO, H. WALLACH, R. FERGUS, S. VISHWANATHAN, R. GARNETT, vol. 30, Curran Associates, Inc., 2017, pp. 4765–4774, URL: <https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf>.
- [13] K. MULLANY, M. MINNECI, R. MONJAZEB, O. COIADO: *Overview of ectopic pregnancy diagnosis, management, and innovation*, Women's Health 19 (2023), p. 17455057231160348, DOI: [10.1177/17455057231160348](https://doi.org/10.1177/17455057231160348).

- [14] H. MURRAY, H. BAAKDAH, T. BARDELL, T. TULANDI: *Diagnosis and treatment of ectopic pregnancy*, Canadian Medical Association Journal 173.8 (2005), pp. 905–912, doi: [10.1503/cmaj.050222](#).
- [15] S. A. RAZAVI, H. PAKNIAT, A. EMAMI, M. MIRABEDINI, M. MAHDAVI, M. MIRZADEH: *Ectopic pregnancy and its associated risk factors: a case-control study in Qazvin*, International Journal of Fertility and Sterility 16.3 (2022), pp. 202–206, doi: [10.22074/ijfs.2021.537225.1167](#).
- [16] P. RUEANGKET, K. RITTILUECHAI, A. PRAYOTE: *Predictive analytical model for ectopic pregnancy diagnosis: statistics vs. machine learning*, Frontiers in Medicine 9 (2022), p. 976829, doi: [10.3389/fmed.2022.976829](#).
- [17] B. VAN CALSTER, L. WYNANTS, J. F. M. VERBEEK, J. Y. VERBAKEL, E. CHRISTODOULOU, A. J. VICKERS, M. J. ROOBOL, E. W. STEYERBERG: *Reporting and interpreting decision curve analysis: a guide for investigators*, European Urology 74.6 (2018), pp. 796–804, doi: [10.1016/j.eururo.2018.08.038](#).
- [18] P. J. YONG, S. MATWANI, C. BRACE, A. QUAIATTINI, M. A. BEDAIWY, A. ALBERT, C. ALLAIRE: *Endometriosis and ectopic pregnancy: a meta-analysis*, Journal of Minimally Invasive Gynecology 27.2 (2020), 352–361.e2, doi: [10.1016/j.jmig.2019.09.778](#).
- [19] Z.-H. ZHOU: *Ensemble learning*, in: Encyclopedia of Biometrics, ed. by S. Z. LI, A. K. JAIN, Springer, 2015, pp. 411–416, doi: [10.1007/978-0-387-73003-5_293](#).



Comparative Analysis of Vulnerability Dynamics in Operating Systems and Containerized Environments

Ferenc Koczka

Eszterházy Károly Catholic University
ferenc.koczka@uni-eszterhazy.hu

Abstract. One of the most significant challenges in modern IT infrastructure management is maintaining the delicate equilibrium between technical debt and security integrity. In environments characterized by rapidly evolving requirements, research and economic priorities often necessitate the deployment of ad-hoc solutions. This process typically results in significant heterogeneity, where legacy systems and sub-optimally configured endpoints coexist. The central hypothesis of this research is that the “freshly installed” state of an operating system – even with a minimal package selection – does not, in itself, constitute a guarantee of security. This study performs a comparative analysis of the security profiles of traditional, monolithic operating system installations versus modern, Proxmox-based containerized solutions, with specific regard to the vulnerability lifecycle and the recommendations of the CIS (Center for Internet Security) and NIST (National Institute of Standards and Technology).

Keywords: vulnerability management, containerization, operating systems, security drift, CIS benchmarks, NIST SP 800-53, NIST SP 800-190, CVSS, SIEM, Wazuh, Linux containers, attack surface reduction, cybersecurity hardening

1. Theoretical Framework and Regulatory Environment

The theoretical foundation of this study is established upon international cybersecurity frameworks that define protection levels through quantifiable and auditable

parameters. Together, these standards constitute the reference framework against which a system's "security drift" – the measurable variance between the required security baseline and the actual operational state – can be evaluated.

1.1. NIST SP 800-53

NIST Special Publication 800-53, issued by the National Institute of Standards and Technology, serves as the fundamental security control catalog for federal information systems. Two control groups are of paramount importance to our research:

- SI-2 (Flaw Remediation): This control mandates proactive lifecycle management in addition to patch deployment. It requires organizations to identify and remediate software defects within a defined timeframe. In our study, we evaluated the effectiveness of SI-2 by monitoring the residual risk remaining in the system following an official minimal ISO installation and the execution of available updates (via `apt update` & `apt full-upgrade -y`) followed by a mandatory system reboot.
- SC-7 (Boundary Protection): This establishes requirements for logical isolation and service segmentation. It provides the theoretical basis for comparing bare-metal operating systems with containerized environments. While boundary protection in bare-metal systems often relies solely on external firewalls, containerization (specifically `lxc` in our research) implements SC-7 requirements through internal, kernel-level isolation [3].

1.2. NIST SP 800-190

Given that the research is built upon Proxmox and `lxc` technologies, the NIST SP 800-190 Application Container Security Guide is also a decisive factor [5]. According to this framework, containerization presents both advantages and disadvantages. A primary advantage is the natural reduction of the attack surface due to the minimization of user-space content; consequently, containers do not include unnecessary drivers and, in most cases, lack redundant background processes. However, the shared kernel vulnerability poses a significant risk. If an attacker manages to achieve a "container escape", the entire host system – including all other containers – is jeopardized. The vulnerability dynamics examined in this paper specifically analyze how the rapid redeployability (immutability) of containers serves to offset these specific risks.

1.3. CIS Benchmarks

While NIST defines high-level security controls, the Center for Internet Security (CIS) prescribes concrete, technical-level configuration parameters. In applying the CIS Ubuntu Linux 24.04 LTS STIG Benchmark [1], as well as similar benchmarks for Debian Linux and Microsoft Windows 11, we examined the systems across more than 200 audit points aimed at hardening configuration settings. Throughout the

research, the CIS Compliance Score served as the primary comparative metric between the various OS variants.

1.4. CVSS

Detected CVEs (Common Vulnerabilities and Exposures) were evaluated not merely based on quantitative factors but were weighted according to the metrics of the Common Vulnerability Scoring System (CVSS) [2]. The CVSS v3.1 and v4.0 frameworks – which remain the most widely accepted standards despite their subjective elements – categorize the severity of vulnerabilities through a process based on the measurement of their specific characteristics. Consequently, this system is suitable for classifying flaws ranging from critical vulnerabilities to low-risk security issues requiring only local access.

2. Methodology and Data Collection

The methodological structure of this research is based on a quantitative comparative analysis conducted in two successive phases. The first phase involved a retrospective analysis within a controlled university network environment, which was subsequently expanded in the second phase to encompass real-world market environments: the infrastructures of commercial enterprises employing more than 50 people. This dual approach enabled the validation of theoretical vulnerability models under actual operational conditions.

The core of the empirical investigation focused on analyzing the correlation between system age, software ‘freshness,’ and detected critical vulnerabilities (CVEs). Our hypothesis posits that the default configurations of traditional operating systems contain redundant services and modules that unnecessarily increase the attack surface, potentially compounding risk over time (technical debt) compared to minimalist containerized environments.

For technical validation, the Wazuh open-source SIEM (Security Information and Event Management) and XDR (Extended Detection and Response) platform was utilized [6]. This tool was selected for its integrated capability to manage security events, configuration audits, and vulnerability detection.

The experimental setup was built on two primary pillars across the evaluated operating systems:

- **Reference Group:** Standard monolithic operating system installations (including Ubuntu, Debian, and Microsoft Windows 11 Pro) maintained within Virtual Machine (VM) environments using default, “out-of-the-box” settings. This represents average system administration practice, where rapid deployment takes precedence over fine-tuning.
- **Test Group:** Functionally equivalent Linux Container (LXC) instances (with the exception of Windows 11 Pro) running within a Proxmox VE environ-

ment, providing the same network services but benefiting from the isolation advantages inherent to the containerization layer.

2.1. Experimental Design and Architectural Scope

To ensure methodological clarity, the experimental configurations were subjected to standardized evaluation criteria. The security telemetry was gathered using Wazuh agents operating under their out-of-the-box, default scanning schedule configurations to reflect the original, non-customized deployment scenarios. The empirical observation was conducted over a 3-day window, during which the systems were evaluated in their initial deployment state, followed by the execution of a complete update cycle utilizing all available stable packages from official repositories.

The structural comparison involves a deliberate architectural boundary regarding Microsoft Windows 11 Pro. In this study, Windows 11 is evaluated exclusively as a Virtual Machine (VM) environment, without an equivalent Windows-based container instance. This scoping is mandated by the underlying technical infrastructure: the Proxmox VE hypervisor utilizes Linux-native kernel features (namespaces and control groups) to implement LXC (Linux Containers). Because Windows containerization relies on a fundamentally different container model and kernel abstraction layer requiring a Microsoft-native ecosystem, its inclusion fell outside the primary scope of this comparative framework. Therefore, the inclusion and identical evaluation of Windows 11 Pro (in terms of configuration compliance and vulnerability metrics) aims to provide a heterogeneous baseline, contextualizing the volume and severity of security flaws found within the Linux ecosystem against a widely deployed alternative enterprise operating system.

2.2. Assessment Parameters and Protocols

During the investigation, both the network and the local scanning protocols were applied, with particular focus on the following critical metrics:

- **Attack Surface (Exposure Surface):** We examined not only the number of open ports but also the versions and necessity of the underlying services. For instance, we analyzed Level 1 (highly recommended basic security hardening) and Level 2 (defense-in-depth) recommendations within the CIS Benchmarks, ensuring that unnecessary and insecure protocols (e.g., telnetd) which pose immediate security risks were strictly audited.
- **Audit of Active Background Processes:** We identified services that are not required for the specific system role but remain part of the default installation (e.g., avahi-daemon, or Bluetooth server processes in a server environment).
- **CVE Detection:** Using the Wazuh vulnerability database, we maintained a real-time inventory of known defects in installed packages, with special attention to critical-rated vulnerabilities (CVSS > 9.0), such as those that affect the glibc or OpenSSL libraries [4].

- **Hardening Drift:** The most significant part of the measurement involved a comparison based on the CIS Ubuntu Linux 24.04 LTS STIG Benchmark, which quantifies the extent to which a system's configuration deviates from the optimal security baseline.

Data was recorded following the initial installation and the completion of the first update cycle. The systems involved in the study were:

- Proxmox VE 9.1
- Windows 11 Pro vm
- Ubuntu 24.04.3 vm
- Debian 12 vm
- Debian 13 vm
- Ubuntu 24.04.3 ct
- Debian 12 ct
- Debian 13 ct

3. Results

The data collected during the study suggest patterns aligned with a gradual vulnerability accumulation process over time, often described in the literature as security drift. As system uptime increases, configurations tend to diverge from their initial secure baseline, while the number of unpatched software vulnerabilities correspondingly increases.

Measurements focusing on configuration compliance (CIS Benchmark results) show that an out-of-the-box installation satisfies only a subset of the recommended industry security controls. Table 1 summarizes the CIS compliance indices for the evaluated environments immediately after system deployment and following the installation of available updates.

Based on the CIS Benchmark assessment results (Table 1), Ubuntu 24.04 achieved a compliance score of 50% in the virtual machine configuration and 51% in the lxc container environment, indicating only marginal improvement in the default hardened state. Debian-based systems demonstrated lower baseline compliance, ranging from 37% (VM) to 41% (container), while Proxmox VE itself achieved 40

In contrast, Windows 11 Pro exhibited a significantly lower compliance score of 24%, with 358 failed controls compared to 115 passed controls, highlighting a substantial deviation from CIS-recommended configurations in its default state. This discrepancy is largely attributable to the fact that a default Windows 11 installation includes numerous pre-installed consumer-oriented applications (bloatware) and active telemetry services, which inherently creates a broader attack surface

Table 1. CIS Benchmark compliance results across evaluated systems (vm vs lxc environments).

OS / Environment	Passed	Failed	Not appl.	Compliance score
Ubuntu 24.04.3 vm	119	118	42	50%
Ubuntu 24.04.3 ct	120	112	47	51%
Debian 12 vm	66	111	30	37%
Debian 12 ct	70	103	34	40%
Debian 13 vm	66	111	30	37%
Debian 13 ct	72	101	34	41%
PVE (Proxmox)	75	110	22	40%
Windows 11 Pro vm	115	358	9	24%

and lower baseline compliance compared to the 'minimal' installation profile used for the Linux distributions in this study. A consistent pattern can be observed across all tested Linux distributions: containerized (lxc) environments achieved slightly higher compliance scores than their virtual machine counterparts. For example, Debian 13 improved from 37% (VM) to 41% (container), while Debian 12 increased from 37% to 40%. Ubuntu 24.04 showed a smaller difference, increasing from 50% to 51%. Overall, the observed compliance differences between VM and container environments range between 1–4 percentage points, indicating that while containerization alone does not ensure high compliance, it provides a consistently improved baseline.

Table 2. A comparative overview of the vulnerability assessment results.

OS / Environment	Critical	High	Medium	Low
Ubuntu 24.04.3 vm	3	145	337	43
Ubuntu 24.04.3 ct	31	159	193	45
Debian 12 vm	0	192	789	6
Debian 12 ct	0	0	6	0
Debian 13 vm	0	14	58	0
Debian 13 ct	0	3	5	0
PVE (Proxmox)	0	0	8	19
Windows 11 Pro vm	2	212	76	0

During the analysis of software vulnerabilities, defects within installed packages were identified using the Wazuh SIEM platform. A notable observation is that containerized environments demonstrated improved outcomes following the update cycle. While certain container instances initially exhibited a higher num-

ber of critical vulnerabilities, targeted updates significantly reduced these values. In particular, Debian-based container instances reached a state where no critical vulnerabilities were detected by the applied scanning methodology.

The findings suggest that, under the tested conditions, containerized environments may offer certain security advantages over traditional monolithic system deployments. Based on the analysis, three primary contributing factors can be identified:

- **Architectural Minimalism (Abstraction Level):** container images typically include only the components required for a given function, reducing the presence of unnecessary software packages and services. In contrast, general-purpose operating systems often include a broader set of default components, which may increase the potential attack surface.
- **Logical Isolation:** namespace-based isolation and control groups (cgroups) provide separation between workloads. These mechanisms can limit the impact of a potential compromise and may reduce the likelihood of lateral movement compared to less segmented environments.
- **Immutability and Update Strategy:** container-based workflows support the concept of immutable infrastructure, where systems are redeployed from updated images rather than modified in place. This approach can contribute to maintaining alignment with security baselines and supports practices aligned with NIST SP 800-53 SI-2 (Flaw Remediation).
- Overall, the results indicate that the security characteristics of a system are significantly influenced by architectural design choices, particularly in relation to component minimalism, isolation mechanisms, and update strategies.

4. Conclusion

The findings of this study suggest that vulnerability management should not be regarded solely as an operational maintenance activity, but rather as a function strongly influenced by architectural design decisions. The results indicate that configuration drift and vulnerability accumulation are observable characteristics of traditionally managed systems, which may be mitigated through alternative deployment models. Within the scope of the examined environments, container-based approaches demonstrated favorable security characteristics compared to monolithic system deployments. However, these advantages are context-dependent and rely on appropriate configuration and operational practices.

Based on the observations, the following strategic considerations can be formulated:

- **Modernization Strategy:** containerized deployment models may provide a viable alternative to traditional in-place upgrades of legacy systems. Under the tested conditions, functionally equivalent lxc containers exhibited fewer

critical vulnerabilities after the update cycle than comparable virtual machine instances. This difference appears to be associated with reduced component complexity and a smaller effective attack surface.

- **Hardening by Design:** the use of pre-configured and security-hardened templates can reduce the initial compliance gap observed in default system installations. Embedding CIS-aligned configurations into deployment artifacts may support more consistent adherence to security baselines from the outset.
- **Immutability and Auditability:** adopting immutable infrastructure principles – where systems are redeployed rather than modified – can improve traceability and help maintain alignment with established security controls, including practices consistent with NIST SP 800-53 SI-2. This approach may reduce the long-term effects of configuration drift observed in traditionally managed environments.

At the same time, it is important to emphasize that the security benefits of containerization are not inherent but depend on correct implementation. Misconfigured containers (e.g., excessive privileges or inadequate isolation settings) may negate these advantages and introduce additional risks.

Finally, this study has several limitations. The results are based on a specific technological stack (Proxmox VE and lxc), a limited set of operating systems, and a defined observation window. Consequently, the findings should be interpreted within this context; given the 3-day observation window, these results reflect immediate post-deployment dynamics rather than long-term risk progression, and further research is required to evaluate their generalizability.

Future work should extend the analysis to additional containerization platforms (e.g., Docker, Kubernetes) and include longer observation periods to better characterize vulnerability lifecycle dynamics.

References

- [1] CENTER FOR INTERNET SECURITY: *CIS Benchmarks: Ubuntu Linux 24.04 LTS STIG Benchmark*, URL: <https://www.cisecurity.org/cis-benchmarks> (visited on 01/29/2026).
- [2] FIRST.ORG, INC.: *Common Vulnerability Scoring System (CVSS)*, URL: <https://www.first.org/cvss/> (visited on 01/29/2026).
- [3] O. JARKAS ET AL.: *A Container Security Survey: Exploits, Attacks, and Defenses*, ACM Computing Surveys 57.7 (2024), DOI: [10.1145/3715001](https://doi.org/10.1145/3715001).
- [4] JUMIATY, B. SOEWITO: *SIEM and Threat Intelligence: Protecting Applications with Wazuh and TheHive*, International Journal of Advanced Computer Science and Applications 15.9 (2024).
- [5] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY: *Special Publication 800-190: Application Container Security Guide*, 2017, URL: <https://csrc.nist.gov/pubs/sp/800/190/final> (visited on 01/29/2026).
- [6] S. STANKOVIĆ, S. GAJIN, R. PETROVIĆ: *A Review of Wazuh Tool Capabilities for Detecting Attacks Based on Log Analysis*, in: Proceedings of the IX International Conference, 2024.



Program Patterns in P4 Programs*

Zsófia Laczkó, Máté Tejfel

Eötvös Loránd University, ELTE, Budapest
laczkozsofi@student.elte.hu
matej@inf.elte.hu

Abstract. Programming Protocol-independent Packet Processors (P4) [2] is an open-source domain-specific programming language, which is primarily developed to describe algorithms that determine the forwarding of network data packets. In this research we defined and analysed positive (recommended) and negative (to be avoided) programming patterns in P4 codes. The defined templates were examined by static methodology. We have implemented an analyser program for the investigation of the patterns, integrated in P4Query [7], which is a static analyser tool for P4. Our analyser is able to recognise certain positive or negative templates, as well as provide feedback to developers about patterns found in their code. During the research the analyses that are already in the P4Query tool were utilized, and a new module was created built on them. Primarily, the syntax tree and the control flow graph built from the source code was used. The goal of the paper is to reduce the number of errors in P4 codes and to demonstrate which recurring solutions support or hinder the correct and effective processing of data packets.

Keywords: P4 language, P4Query, program patterns, static analysis

1. Introduction

Programming Protocol-independent Packet Processors (P4) [2] is an open source domain-specific programming language, which is primarily designed to implement algorithms that describe packet processing by different network devices (switches, network interface cards (NICs), routers, etc.). P4 is increasingly central to the de-

*This research was supported by the project no. TKP2021-NVA-29, which has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

velopment of programmable data planes, in particular software-defined networking (SDN) and high-performance network devices.

Specific programming challenges emerge when working with this language. It has pipeline-based processing model, and target-dependent compilation. P4 is a flexible language, intended for high-speed processing of numerous packets. For this reason, there is no well-developed error handling. If an error occurs, the basic solution is that the P4 program drops the packet, which can be a problem in particular cases. Lookup tables also introduce challenges, since it cannot be determined explicitly in advance which action will be executed by the application of a table. P4 only specifies the list of the available actions and establishes the schema of the lookup table. The actual table entries are defined externally to the P4 program and can be modified dynamically. For this reason, it is not visible during development which actions and parameter configurations correspond to given key values.

Due to these specific challenges, static analysis is notably important for P4 language, in addition to dynamic checks. This motivated our research: the identification of positive and negative programming patterns in P4 programs, as well as their automatic detection through static analysis. The positive patterns include recommended coding styles, best practices and solutions. As for the negative patterns, coding strategies that seem to work but potentially contain errors or unexpected behaviour were examined. This category includes examples that indicate defects only in specific edge cases. In this research, we focused on the specific scenarios for which the compiler does not alert. Our new analyzer module is able to locate certain positive or negative templates in P4 programs, as well as inform developers about the occurrences of these patterns in their code.

This research contributes to the prediction of frequent potential errors through the static analysis of P4 code, so we can increase the number of issues that can be identified without running the program. In addition, this study can support work of developers by emphasising well-functioning strategies.

2. Related work

Several articles address debugging in P4 programs, but mostly using dynamic methods such as assertions [4]. BIRNFELD ET AL. [1] analyse P4 programs following an approach similar to the one we used. They examine problems like our negative patterns, however, they use data flow analysis specifically. This approach is useful for examining the values of variables within the program. Compared to this, the syntax tree and control flow graph analysis presented in this study is more effective for inspecting code structure and possible execution paths.

Although Gulabovska et al. [5] provides tools for analysing a different programming language (Python), the description of its static analysis methodology served as a basis for our approach. In this paper, this technique was applied for the examination of patterns. Vera [10] is a tool developed to discover errors in P4 programs, similar to our analyser module. However, it uses symbolic execution, not static

analysis. This method requires inputs. Therefore, analysing all possible execution paths using this approach is difficult. In P4, the use of lookup tables results in a large number of potential paths.

The p4v [6] program checks properties of P4 programs using formal methods and can recognise errors. This approach translates P4 programs into an imperative language called Guarded Command Language (GCL). The verification process is then applied to this representation instead of analysing the original program code directly. The bf4 [3] tool applies both static verification and runtime checks. In addition, it is able to also modify code to prevent errors. Both this program and the previously mentioned ones are primarily designed for detecting and correcting errors. In contrast, this paper also emphasizes positive patterns, thereby providing constructive feedback to developers.

3. Background

The structure of a P4 program depends on a configuration file called architecture. This file determines the mandatory elements of a P4 program. Developers can use custom architectures for their network, however there are some widely used models. In this research we worked with the V1Model, which is one of the most used models.

3.1. Code structure and V1Model architecture

In a P4 program¹, the first section of the code describes the type definitions and the headers applied, illustrated in Listing 1.

Listing 1. Definition part.

```

1  #include <core.p4>
2  #include <v1model.p4>
3  ...
4  typedef bit<48> macAddr_t;
5  typedef bit<32> ip4Addr_t;
6  header ethernet_t {
7      macAddr_t dstAddr;
8      macAddr_t srcAddr;
9      bit<16> etherType;
10 }
11 header ipv4_t {
12     bit<8> ttl;
13     bit<8> protocol;
14     bit<16> hdrChecksum;
15     ip4Addr_t srcAddr;
16     ip4Addr_t dstAddr;
17     ...
18 }
19 ...

```

The V1Model divides P4 programs into five different blocks, as presented in Figure 1.

¹The example code in this section is from: <https://github.com/p4lang/tutorials> (basic.p4)

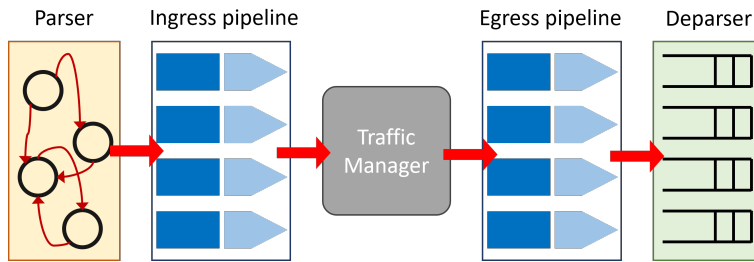


Figure 1. V1Model.

Listing 2 demonstrates the parser, which is the first block of the V1Model. This is responsible for packet processing. The program receives a bitstream at the beginning, from which the parser extracts the packet headers and separates them into fields for further handling.

Listing 2. Parser.

```

1 parser MyParser(packet_in packet, out headers hdr,
2   inout metadata meta, ... ) {
3   state start { transition parse_ethernet; }
4   state parse_ethernet {
5     packet.extract(hdr.ethernet);
6     transition select(hdr.ethernet.etherType) {
7       TYPE_IPV4: parse_ipv4;
8       default: accept; }
9   }
10  state parse_ipv4 {
11    packet.extract(hdr.ipv4); transition accept; }
12 }

```

Listing 3 presents the second block, the ingress pipeline. The majority of the processing logic for incoming packets is executed here. Primarily lookup tables² are used to update the values of fields within headers or metadata. When a table is applied, a lookup is performed based on the specified key fields, and the resulting match determines an action from those defined in the table declaration. This selected action is subsequently executed. It is important to emphasize that the actual table entries are provided outside the P4 program and they can be changed during runtime.

Listing 3. Ingress pipeline.

```

1 control MyIngress(inout headers hdr, inout metadata meta,
2   inout standard_metadata_t standard_metadata) {
3   action drop() { mark_to_drop(standard_metadata); }
4   action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
5     standard_metadata.egress_spec = port;
6     ... }
7   table ipv4_lpm {

```

²Detailed description of the tables: <https://p4.org/wp-content/uploads/sites/53/2024/10/P4-16-spec-v1.2.5.html#sec-tables>

```

8         key = { hdr.ipv4.dstAddr: lpm; }
9         actions = { ipv4_forward; drop; NoAction; } ... }
10    apply { if (hdr.ipv4.isValid()) { ipv4_lpm.apply(); } else {} }
11 }

```

After ingress block comes the Traffic Manager, which cannot be programmed by P4 language, so it is outside the scope of this work.

The next block is the egress block, which is similar to ingress. This block may be empty; however, it is handled separately since some architectures specify which operations can appear in the ingress and which in the egress block.

Listing 4 illustrates the final block, which is the deparser. This section assembles the packet for forwarding.

Listing 4. Deparser.

```

1 control MyDeparser(packet_out packet, in headers hdr) {
2     apply { packet.emit(hdr.ethernet); packet.emit(hdr.ipv4); }
3 }

```

3.2. P4Query

Our new module is integrated in P4Query [7], which is a static analyser tool for P4. It can map a P4 program code onto a graph applying the Gremlin graph description language [9]. Gremlin is a graph traversal and graph query framework that can be used for example from the Java programming language. P4Query tool contains different analyses that result in an internal graph representation. With the graph we are able to process the code from different perspectives. Each analysis can rely on previously generated results, which are also integrated in the representation graphs (mainly by adding new edges to the graph), illustrated in Figure 2. This approach allows the execution of different analysers sequentially according to a dependency chain. The tool makes possible to examine for instance a syntax tree, a symbol table or a control flow graph.

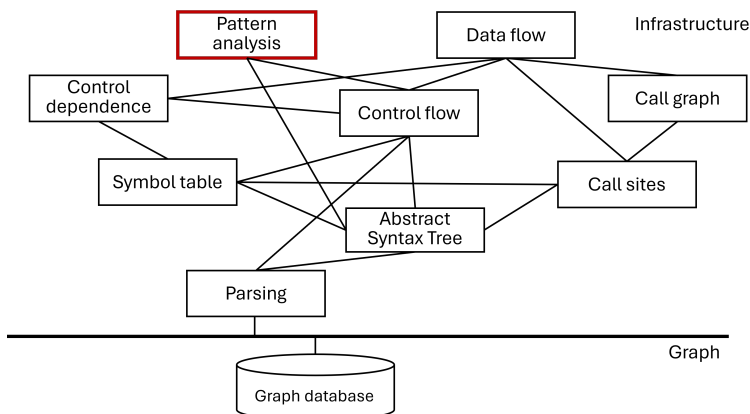


Figure 2. Analyses in P4Query.

4. Patterns

In the first part of this research recurring code patterns and programming styles were identified in P4 programs. The collected patterns were classified into two main categories: positive (recommended) and negative (to be avoided) samples. Positive patterns represent widely adopted implementation approaches that contribute to code readability, maintainability, and reliable program execution. Conversely, negative patterns introduce potential errors or ineffective code fragments.

Highlighting positive patterns provides inspection into the degree to how developers follow the conventions and best practices. The resulting feedback can support the improvement of the development team. Moreover, positive reinforcement and recognition have a motivating effect on developers. In contrast, the identification of negative patterns serves to warn developers about potential issues and risks.

During the examination of patterns, the specification of the programming language [8] was used, as well as the P4C compiler, which is the official reference compiler for the P4 language. Potentially incorrect code samples were collected and then analysed to determine in which cases problems are reported during compilation, either in the form of an error or a warning. Although this collection of patterns includes cases for which the compiler provides warnings, we mainly concentrated on those that are not detected.

The collection was not intended to cover the entire documentation. Instead, it serves as an illustrative set of examples for the analyser module. It can be further extended through the study of real-world code bases and examine other language constructions.

4.1. Positive patterns

Pattern 4.1.1. Headers have to be used with *valid* status with all their fields properly assigned values.

In P4, every header structure contains a hidden field, which has Boolean type. When its value is `true`, the header is considered *valid*. Fundamentally, the compiler issues a warning if the status of a header is set to *invalid* and then one of its fields is attempted to be used. However, if a header has been set to *invalid* status, and then to *valid* status, then values have to be assigned to its fields before being used. In this case the compiler does not issue a warning.

Pattern 4.1.2. Widely known header types should be used according to the conventional or standard practice.

In P4 programs, header types are defined by the programmer at the beginning of the source code. When a type is declared with the name of a well-known header, such as IPv4, its field names and field lengths should follow the conventional structure of that header for improving code clarity. Further examples of well-known header types are IPv6, TCP, UDP and Ethernet.

Pattern 4.1.3. Define a variable only if it did not previously exist and refer to it only if it has already been defined and initialized.

This can be verified based on the control flow analysis. In the case of local variables, the compiler typically indicates the problem, at least in the form of a warning. Regarding metadata fields, special attention should be paid to their initialization and first usage.

Pattern 4.1.4. Define variables only with permitted lengths (for instance, `bit<1>`, `bit<8>`, ...).

Different architectures and targets may support only specific sizes. As an example, some targets are able to handle only byte-wide (8-bit) variables well.

Pattern 4.1.5. In the deparse part, it is recommended to explicitly determine the order of appending data to the output packet.

There are several ways to define headers for deparsing. The simplest option is to specify the headers name explicitly. In this case, the sequence of the instructions clearly determines the order of the appending. The situation is more complex when a struct containing the headers is deparsed. In such cases, the append order is implicitly defined by the type declaration of the struct. It is also possible to deparse a header stack, in which scenario the elements are processed sequentially according to their increasing indices. In this instance the order does not appear explicitly. The implicit sequence does not automatically result in an error; however, it demands careful attention in every case.

4.2. Negative patterns

Code may contain unnecessary parts, including both syntactically observable ones and those that become superfluous during program execution.

Pattern 4.2.1. Writing unused variables, declarations (e.g. with `typedef` keyword), formal parameter, states.

Pattern 4.2.2. Unreachable code segments, like code after a return statement.

Pattern 4.2.3. Setting a header as *valid* when it has already a *valid* status.

Pattern 4.2.4. The deparser processes an *invalid* header.

In the deparser block, the appending of the data from the header is performed using the `emit` method. When it receives an *invalid* header as parameter, no action is taken. This is not necessarily incorrect code. It enables a simplified programming approach when the presence of a header is optional. However, it can easily lead to an error and is therefore reported.

Issues can occur in relation to the usage of variables, depending on the values assigned to them.

Pattern 4.2.5. Variable overflow and type conversions can cause errors.

For example, the following operations may occur an error [8]:

- `int<W>  $\longleftrightarrow$  bit<W>`

Issues can occur when converting between positive and negative numbers.

- $\text{bit}\langle W \rangle \rightarrow \text{bit}\langle X \rangle, \text{int}\langle W \rangle \rightarrow \text{int}\langle X \rangle$
It truncates the value when $W > X$.
- $\text{int} \rightarrow \text{bit}\langle W \rangle, \text{int} \rightarrow \text{int}\langle W \rangle$
Value can be truncated, overflow may occur, converting between positive and negative numbers can be incorrect. (For the last two cases, the compiler provides warnings as well.)

Pattern 4.2.6. Reading a field of a header without previously manually setting values to that field or parsing the header [1]. This results in an unspecified value.

A specific instance of this pattern is when a packet lacking an IPv4 header is handled by the `ipv4_lpm` table [1]. In this case, a header field would be used as a key which has not been parsed.

The following two patterns do not primarily cause errors. Instead, they are related to code maintainability and the appropriate use of language constructs.

Pattern 4.2.7. Repetition of larger code segments.

Pattern 4.2.8. Writing a select statement without a default case.

5. Analyser module

In order to facilitate development process, a new analyser program³ was developed to automatically locate the collected patterns in P4 codes. Static analysis was used to achieve this, since it enables the examination of program behaviour without executing the code and the identification of the potential issues directly from the program structure. For the detection of the patterns, the syntax tree and the control flow graph (CFG) have to be examined. The new analyser module is integrated into P4Query tool, as it is able to provide an internal graph representation which contains the syntax tree together with the associated control-flow edges. This complex internal graph representation contains information derived from both the syntax tree and the control-flow analysis. This data was adequate for the currently implemented examinations. However, other analyses available in P4Query can also be utilized if further information is needed during future development.

The implementation includes several distinct analyser classes, which receive this graph representation as an input parameter. Each class is responsible for detecting one or more interconnected patterns. The Gremlin graph traversal language was primarily used for the analysis.

Table 1 contains which class is responsible for detecting which pattern. As shown in the table, the analyser is currently not able to identify all patterns. Some of the missing patterns cannot be examined using the static analysis approach applied in this work. For instance, Pattern 4.1.4 requires information about the current target architecture. Other omitted samples can be implemented in the future to extend the module. If the detection of a new pattern is required, the implementation can be extended easily by creating a new analyser class.

³The code is accessible from <https://github.com/laczkoszofi/PatternAnalysisInP4>

Table 1. Patterns related to analyser classes.

Analyser class	Connected pattern
Status of headers	4.1.1 , 4.1.5 , 4.2.3 , 4.2.4 , 4.2.6
Select statement	4.2.8
Return statement	4.2.2
Known headers	4.1.2

In the implementation every possible execution path is analysed based on the control flow graph. As a result, all occurrences of the analysed patterns are identified from the code. The purpose of highlighting potential errors is not to assert that a program will definitely behave incorrectly, but rather serves a warning about potential issues that developers should consider. As a consequence of the platform-dependent nature of the P4 language, many types of information are not available from the source code alone. In addition, frequent use of lookup tables makes analysis more difficult, since it is not visible which actions will be executed. Because the actual table entries are not defined in the P4 code, the module has to consider all possible actions. This may cause false positive warnings, for which static analysis cannot offer more detailed information.

As an outcome of the analyses of the classes, a report is generated, which delivers warnings to developers about possible issues and about the found patterns. Based on the results of the examination, feedback number is provided to developers. It is calculated by assigning negative scores to each found undesirable patterns and positive scores to recommended ones. Different templates contribute to the total score with different weights. Currently one total score is calculated. However, for more transparent feedback, the value derived from positive patterns could be reported separately from the negative score. It would emphasize better the warnings for potential issues. Nevertheless, the report already displays the individual scores for each entry.

5.1. Example: Status of headers analysis

For instance, in the context of the Status of headers pattern the module examines *valid* or *invalid* status of headers during the execution of the P4 code. Additionally, it verifies whether the *valid* status of a header really guarantees that all the fields have been assigned values. Our analysis also attends to what happens with a header during parsing and deparsing, as well as when its status is set directly by the `setValid` or `setInvalid` methods. The program issues a warning if a field has not been assigned value when it is used or a header is unnecessarily marked as *valid*. In addition, developers get a report about the deparser block. It contains the expected status of the headers to be deparsed, as well as a list of fields that may not have assigned values.

Listing 5. Ingress block with potential error.

```

1  control MyIngress(...) {
2      ...
3      action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
4          standard_metadata.egress_spec = port;
5          hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
6          hdr.ethernet.dstAddr = dstAddr;
7          hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
8      }
9
10     table ipv4_lpm {
11         key = { hdr.ipv4.dstAddr: lpm; }
12         actions = { ipv4_forward; drop; NoAction; }
13         size = 1024;
14         default_action = drop();
15     }
16
17     apply {
18         if (hdr.ipv4.totalLen == 5) {
19             meta.egress_spec = 1;
20             hdr.ethernet.setInvalid();
21         } else {
22             meta.egress_spec = 2;
23         }
24         hdr.ethernet.setValid();
25
26         if (hdr.ipv4.isValid()) {
27             ipv4_lpm.apply();
28         } else {}
29     }
30 }

```

The `ipv4_lpm` table is applied in the code section presented in Listing 5, in the yellow-boxed line. It has three actions (declared in line 12), and it cannot be determined in advance which one will be performed. During execution, the `apply` block may follow the true branch of the first `if` statement, marking the Ethernet header as *invalid* (highlighted in blue). Next, when the `ipv4_lpm` table is applied, an action is selected depending on the concrete table entries. Assume that action `ipv4_forward` is executed. The destination address field (`dstAddr`) in line 5, highlighted in red would be used without a value, because even though the status of header is reset to *valid* (in line 24, highlighted in green), its fields have not been assigned any values. Our analyser module provides a warning in this case, presented in Figure 3.

WARNING -> action: `ipv4_forward`: `dstAddr` has no value (line: 519)

Score: -2

Figure 3. Output of Status of headers analysis.

5.2. Example: Implementation of an analyser class

To demonstrate the implementation of the analysers, the class connected to the status of headers is briefly introduced. The operation of the analyser class connected to header status examination can be divided into six main steps.

1. The first step is to collect the header types used in the program based on the declarations located at the beginning of the code.
2. The next step is to construct an object for tracking the current status of the headers. This object is continuously updated during the analysis of the execution paths.
3. As the third step, the parser block is analysed to identify the headers that can be parsed during packet processing.
4. The execution paths to be analysed are constructed according to the V1Model architecture. The paths are built from the vertices of the representation graph.
5. The main phase of the analysis is the examination of the controls (the blocks present in the V1Model). For every node, the module checks whether any headers or header fields are accessed and whether the status of any header is modified.
6. The last step is the examination of the deparser block. During this step, the status of all headers to be deparsed is determined and written to the report. Furthermore, for each header, the report contains a list of fields that are not guaranteed to have assigned values.

The implementation of these phases requires numerous Gremlin queries. To demonstrate this process, the first stage is presented in the following. This step is the collection of the header types. According to the language syntax⁴ this operation starts with a Gremlin traversal that selects all `HeaderTypeDeclarationContext` nodes. From these nodes, the header name can be accessed via the `NameContext` child node, and the field names are reached through the `StructFieldListContext` child node.

Listing 6. Collection of the header types.

```

1 List<Object> headerTypeDeclarations =
2     g.V().has("class", "TypeDeclarationContext")
3     .out().has("class", "DerivedTypeDeclarationContext")
4     .out().has("class", "HeaderTypeDeclarationContext")
5     .values("nodeId").toList();
6
7 for (Object decl : headerTypeDeclarations) {
8     List<Object> headerName = g.V().has("nodeId", decl)
9     .out().has("class", "NameContext")
10    .repeat(__.out()).until(__.not(__.out()))

```

⁴<https://p4.org/wp-content/uploads/sites/53/2024/10/P4-16-spec-v1.2.5.html#sec-header-types>

```

11         .values("value").toList();
12
13     List<Object> headerFieldsNames = g.V().has("nodeId", decl)
14         .out().has("class", "StructFieldListContext")
15         .repeat(__.outE("syn").inV())
16         .emit().dedup().has("class", "NameContext")
17         .repeat(__.out()).until(__.not(__.out()))
18         .values("value").toList();
19     ...
20 }

```

Listing 6 illustrates the Gremlin queries needed for this traversal. At the beginning, the nodes representing header declarations are selected, which are then processed in a for loop. In the first query, the process begins by locating the current header declaration node, and the traversal continues to the node that determines the name. From there, the value attribute of the reachable leaf node contains the required information. The second query follows a similar approach, except that it moves to the node declaring the list of fields instead of the name node. It selects all nodes representing field names. The necessary data is likewise contained in the reachable leaf nodes.

6. Conclusion and Future Work

The main purpose of the study is to uncover frequently occurring programming patterns in P4 programs using static analysis, including both positive and negative aspects. On one hand, the collection of recommended patterns supports the creation of more maintainable and effective P4 programs, as well as facilitating the work of developers. On the other hand, recognizing negative patterns allows for the early detection of potential problems without program execution. This reduces the complexity of later testing activities using other approaches.

Our analysis identified 5 positive and 8 negative patterns in P4 programs. The implemented static analyser successfully detects some of these patterns, highlighting common mistakes and best practices in P4 coding, as well as providing feedback for developers about found patterns in their code.

In future work, our plan is to expand the collection of identified P4 patterns, including both new positive and negative examples. In addition, our goal is to improve the implemented analyser module so it can recognize more and more patterns automatically, further supporting code quality and correctness. Further research could explore whether there is a demonstrable relationship between the patterns and code quality.

Our research group is currently conducting research on using LLMs (Large Language Models) for refactoring P4 codes. If this approach is successful, it may also be applied to error detection similar to the analyser presented in this paper. A limiting factor in this type of research is that P4 is not a widely used language, and therefore only a limited number of examples are available.

References

- [1] K. BIRNFELD, D. C. DA SILVA, W. CORDEIRO, B. B. N. DE FRANÇA: *P4 Switch Code Data Flow Analysis: Towards Stronger Verification of Forwarding Plane Software*, in: NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1–8, DOI: [10.1109/NOMS47738.2020.9110307](https://doi.org/10.1109/NOMS47738.2020.9110307).
- [2] P. BOSSHART, D. DALY, G. GIBB, M. IZZARD, N. MCKEOWN, J. REXFORD, C. SCHLESINGER, D. TALAYCO, A. VAHDAT, G. VARGHESE, D. WALKER: *P4: programming protocol-independent packet processors*, SIGCOMM Comput. Commun. Rev. 44.3 (July 2014), pp. 87–95, ISSN: 0146-4833, DOI: [10.1145/2656877.2656890](https://doi.org/10.1145/2656877.2656890).
- [3] D. DUMITRESCU, R. STOENESCU, L. NEGREANU, C. RAICIU: *bf4: towards bug-free P4 programs*, in: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM '20, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 571–585, ISBN: 9781450379557, DOI: [10.1145/3387514.3405888](https://doi.org/10.1145/3387514.3405888), URL: <https://doi.org/10.1145/3387514.3405888>.
- [4] L. FREIRE, M. NEVES, L. LEAL, K. LEVCHENKO, A. SCHAEFFER-FILHO, M. BARCELLOS: *Uncovering Bugs in P4 Programs with Assertion-based Verification*, in: Proceedings of the Symposium on SDN Research, SOSR '18, Los Angeles, CA, USA: Association for Computing Machinery, 2018, ISBN: 9781450356640, DOI: [10.1145/3185467.3185499](https://doi.org/10.1145/3185467.3185499).
- [5] H. GULABOVSKA, Z. PORKOLÁB: *Survey on Static Analysis Tools of Python Programs*, in: SQAMIA, 2019, URL: <https://api.semanticscholar.org/CorpusID:209149918>.
- [6] J. LIU, W. HALLAHAN, C. SCHLESINGER, M. SHARIF, J. LEE, R. SOULÉ, H. WANG, C. CAŞCAVAL, N. MCKEOWN, N. FOSTER: *p4v: practical verification for programmable data planes*, in: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '18, Budapest, Hungary: Association for Computing Machinery, 2018, pp. 490–503, ISBN: 9781450355674, DOI: [10.1145/3230543.3230582](https://doi.org/10.1145/3230543.3230582), URL: <https://doi.org/10.1145/3230543.3230582>.
- [7] D. LUKÁCS, G. TÓTH, M. TEJFEL: *P4Query: Static analyser framework for P4*, Annales Mathematicae et Informaticae 57 (2023), pp. 49–64, DOI: [10.33039/ami.2023.03.002](https://doi.org/10.33039/ami.2023.03.002).
- [8] P4 LANGUAGE CONSORTIUM: *P4 Specifications*, URL: <https://p4.org/specifications/> (visited on 12/30/2025).
- [9] M. A. RODRIGUEZ: *The Gremlin graph traversal machine and language (invited talk)*, in: Proceedings of the 15th Symposium on Database Programming Languages, DBPL 2015, Pittsburgh, PA, USA: Association for Computing Machinery, 2015, pp. 1–10, ISBN: 9781450339025, DOI: [10.1145/2815072.2815073](https://doi.org/10.1145/2815072.2815073).
- [10] R. STOENESCU, D. DUMITRESCU, M. POPOVICI, L. NEGREANU, C. RAICIU: *Debugging P4 programs with vera*, in: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '18, Budapest, Hungary: Association for Computing Machinery, 2018, pp. 518–532, ISBN: 9781450355674, DOI: [10.1145/3230543.3230548](https://doi.org/10.1145/3230543.3230548), URL: <https://doi.org/10.1145/3230543.3230548>.



Experimental Study of Building a Nest by an Automaton

Tamás Lukovszki, Gábor Pusztai

Eötvös Loránd University (ELTE),
Faculty of Informatics,
Hungary lukovszki@inf.elte.hu
pusztai.gabor@inf.elte.hu

Abstract. Building upon the theoretical framework established by Czyzowicz et al. [3, 4], this study investigates the problem of constructing a *nest* from a set of bricks arranged in a two-dimensional grid by a robot modeled as a deterministic finite automaton. The subgraph of the grid induced by cells containing bricks is called the shape, which is initially connected. The *span* of the shape is defined as the Manhattan distance between its furthest cells. The robot starts at a cell with a brick. At each step, it can pick up a brick from its current cell if the current cell contains a brick, move to an adjacent cell, and drop a brick at an empty cell. The task of the robot is to reorganize the initial shape into the most compact shape – called the nest – with minimum possible span. In [3, 4] a finite automaton has been designed that accomplishes this task in $O(s \cdot n)$ time, where s is the span of the initial shape and n is the number of bricks. This running time is asymptotically optimal in the worst case. However, the original work omits some implementation details, and a straight-forward implementation results in inefficient runtime.

We present two algorithmic refinements of the original procedures and evaluate their performance using our interactive Java-based environment, the *NestBuilder simulator*, which facilitates visualization and enables comprehensive evaluation. Experimental results show a significant reduction in movement overhead while preserving the theoretical worst-case guarantees of the original algorithm.

Keywords: Mobile robots, Nest building, Finite automaton

2020 Mathematics Subject Classification: 68T40, 68U05, 68M14, 68W15

1. Introduction

The study of *mobile autonomous robot* with limited sensing and memory represents a rapidly advancing field in distributed systems, particularly regarding the design of effective algorithms that function without centralized control, prior environmental knowledge, or extensive data storage, leveraging local sensory data for decision-making. Such decentralized system can provide significant advantages in scalability, cost-effectiveness, and fault tolerance, making them indispensable for the robust execution of *construction tasks* where simple agents must systematically gather and reorganize passive objects into predefined or compact shape geometries under strict localized sensing and memory constraints. A core requirement for these algorithms is the rigorous preservation of structural connectivity throughout the entire process. The practical implications of such tasks are vast, ranging from operations in unfavorable or inaccessible environments like in microscopic environments [1, 10], industrial logistics for hazardous material decontamination, and autonomous outer space assembly [2, 11]. Prior research has extensively investigated the reconfiguration of connected initial configurations on a grid using robots modeled as deterministic finite automata in [5–9, 12].

Czyzowicz et al. [4] showed that a deterministic finite automaton can organize an initially connected shape of bricks in the two-dimensional grid into a compact shape, called a nest, in $O(s \cdot n)$ time, where s is the span of the initial shape and n is the number of bricks, i.e. the Manhattan distance between the furthest cells in the shape. They also showed that this running time is asymptotically worst-case optimal. Despite the proven asymptotic optimality, there is a significant gap between the theoretical bound and the empirical performance. Many implementation details were omitted in the original work for simplicity, and a straight-forward implementation results in inefficient runtime.

This paper addresses these challenges by introducing the following:

- The *NestBuilder simulator*, an interactive Java-based environment for visualization and running rigorous evaluations, and additionally a Unity-based graphical player dedicated to visualization only.
- We propose two refinements of the original algorithm.
 - First, the *Optimized Rough Disc Extension* method, which is an improvement of the original extension procedure, which utilizes a localized geometric property to identify the next placement position of a new brick on the boundary of the shape.
 - Second, the *Distance-Aware Sweep* method, which is an improvement of the original sweeping procedure that separates the constructed nest from other bricks. Instead of checking $O(\sqrt{n})$ cells of the entire boundary of the nest after adding a new brick, our method only needs to check $O(1)$ cells.

- Experimental evaluation across 100 randomly generated initial configurations indicates that these enhancements yield an improvement of 18.75% in expansion and a 92% in sweeping procedures.

2. Model and Algorithmic Framework

Adopting the formal model and terminology by Czyzowicz et al. [4], the problem is defined within an infinite oriented $\mathbb{Z} \times \mathbb{Z}$ grid, represented as a set of unit square cells in the two-dimensional plane, with all cell sides vertical or horizontal. Each cell has 4 adjacent cells, North, East, South, and West of it. A cell is either *full*, i.e. contains a single brick, or *empty*. For any two cells $c = (x, y)$ and $c' = (x', y')$, the distance $d(c, c')$ is defined as the Manhattan distance (L_1 distance) $d(c, c') = |x - x'| + |y - y'|$. The subgraph of the grid induced by full cells is initially connected. The subgraph induced by the full cells is called the *current shape*. The Manhattan distance between the furthest cells of the shape is called its *span* s .

We are given a mobile robot starting in an arbitrary cell of the initial shape. The robot has no knowledge about the shape, of its size or its span. At each step, it can pick up a brick from its current cell (if it does not carry any brick at this time), move to an adjacent cell, and it can drop a brick at the current cell if it carries a brick and the cell is empty. The robot's objective is to rearrange the initial shape into a *nest*. The nest is a *disc* or a *rough disc*, see subsection 3.2 for details. A *nest* of size n is a shape that has the minimum span among all shapes of size n . While the grid is theoretically infinite, the operational area of the robot is limited by the initial configuration. Given that the algorithm operates through searching for a suitable brick and extending the nest with that brick, the robot remains within a limited distance of the initial shape.

In [4], the robot knows whether each cell at distance at most $r = 8$ from its current cell is full or empty. During the nest building, the robot maintains a special, one-cell component M , called the *marker*. The robot also maintains a *swept region* around the nest. This region is a gap where all cells within a distance of 7 from the nest are empty except for the cell of the marker. The marker signals the robot that it got back in the vicinity of the rough disc. Using the marker allows the robot to avoid storing the nest's coordinates. During the execution of the nest building algorithm, the robot does not ensure that the full cells outside of the rough disc and of the marker form one component. Any component that is different from the rough disc and from the marker is called a *free component*. The robot must identify and retrieve a *free brick*, which is defined as a brick of a free component whose removal does not disconnect that component.

The robot is formally modeled as a deterministic automaton. It may remember a constant number of bits by encoding them in its states (see [4] for a detailed description). The high-level idea of the nest building algorithm is the following.

First, the robot performs some preprocessing by establishing the marker, the initial rough disc, and the swept region. Then it performs the main loop. In each iteration, the robot performs three procedures:

1. **FINDNEXTBRICK**: Finding the next free brick b in a free component. This procedure is not discussed in this article; see [4] for details.
2. **RETURN TOMARKER**: Returns to the marker with b . See [4] for details.
3. **EXTENDROUGHDISC**: Extending the rough disc with b and by calling **SWEEP**, restoring the property that there are no full cells within a Manhattan distance of 7 from the rough disc, except for the marker. See subsection 3.2 and subsection 3.3 for details and improvements.

The authors show that this algorithm can be implemented with a finite-state robot that builds a nest in $O(s \cdot n)$ time steps.

3. Implementation and Practical Refinements

While the original algorithm proposed by Czyzowicz et al. [4] provides a worst-case optimal complexity of $O(s \cdot n)$, the authors noted that many specific implementation details were omitted for the sake of simplicity. Our work bridges this gap by developing a functional simulator and introducing refinements that improve the performance without sacrificing the theoretical guarantees of the finite-state automaton model.

3.1. The NestBuilder Simulator

To evaluate the construction algorithm, we developed the *NestBuilder* simulator using the Java programming language. A primary challenge in simulating a robot navigating an infinite grid $\mathbb{Z} \times \mathbb{Z}$ is the management of memory and coordinates. Our environment manages this infinite grid through a dynamic $n \times n$ collection that expands elastically as the robot moves further away from the origin.

Furthermore, to ensure rigorous stress-testing of the robot's ability to recover connectivity, we utilize *Kruskal's* algorithm with randomly generated weights during the setup phase. This allows us to generate complex, randomized, yet initially connected starting shapes of size z , ensuring the robot must navigate a challenging configuration of bricks to gather all the bricks into a nest.

Beyond the core Java-based simulation engine, this research incorporates a supplementary visualization environment developed in Unity. While the Java simulator is primarily used to execute the algorithm and generate statistical data, the Unity environment serves as a specialized tool for visualising and replaying pre-recorded simulation sequences. The program is deployed via *WebGL*, which provides an extra perspective on the robot's movement and the creation of the nest. The architecture of this visualization tool is built upon a modular simulation framework previously developed and published by us [13], which was specifically designed for modeling autonomous agent behaviors and construction tasks in grid-based environments.

3.2. Optimized Rough Disc Extension

A *disc* D of radius $r \geq 0$ centered at c is defined as the set of cells whose distance from c is at most r . The set of cells of D that are at a distance of i , $0 \leq i \leq r$ from c is called the i -th *layer* of D . A disc of radius $r \geq 0$ contains $n_r = 1 + 4 \cdot \sum_{i=1}^r i = 2r^2 + 2r + 1$ cells and has a span $2r$. When the total number of bricks n satisfies $n_r < n < n_{r+1}$, the structure is classified as a *rough disc*. This structure is formed by extending D with a set of cells F of $n - n_r$ bricks, which are placed in the set of cells adjacent to D starting at the northern neighbor of the easternmost cell and proceeding counterclockwise. Let R be a connected shape formed by the full cells. The *boundary* ∂R of R is defined as the set of cells of R that are adjacent to at least one empty cell.

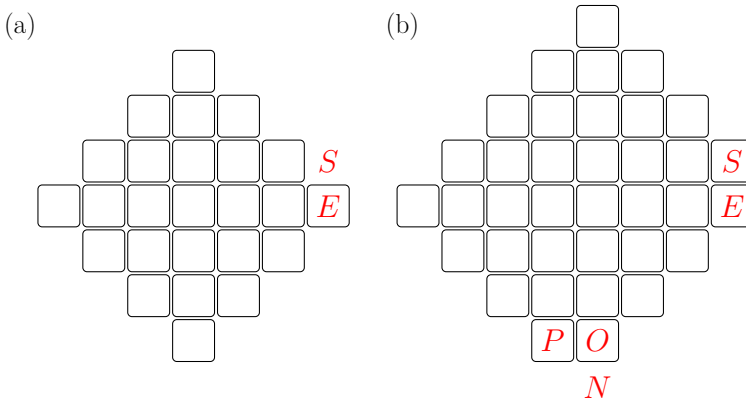


Figure 1. Visual representation of the localized geometric heuristic used to identify the target placement position for new bricks. (a) Disc: The robot identifies the easternmost cell (E) and checks its northern neighbor (S); if S is empty, the nest is recognized as a completed disc, and a new layer begins at S . (b) Rough Disc: For an incomplete layer, the robot identifies the connected pair of bricks P (the previously placed brick) and O (the brick to which the new one will connect) on the boundary to locate the target placement position (N).

In [4], in the original EXTENDROUGHDISC procedure, the robot remembers in its state whether the current nest is a disc or a rough disc. Adding a brick to a disc, it walks to the starting position S of a new layer, which is defined as the northern neighbor of the easternmost cell E of the nest. For a rough disc, it is required to perform a traversal along the boundary ∂R to find the correct location for the brick. The time required for this operation is the size of the boundary of a rough disc, which is $O(\sqrt{n})$, which is $O(s)$.

The original work provides limited specification regarding the precise localization of the placement position. We provide a method to identify this position based on a simple geometric property of the boundary of the nest. During a counterclock-

wise traversal of the nest, the robot evaluates the connectivity of the bricks on the boundary ∂R . This approach makes it unnecessary to remember whether the nest is a disc or a rough disc.

- When the robot reaches the easternmost cell E of the nest, determinable during counterclockwise traversal, when the next ∂R brick would have increased y but non-increased x coordinate. The robot checks the northern neighbor cell S of E . If S is an empty cell, then the robot recognizes the nest as a disc and starts a new layer at the position S .
- Otherwise, the nest is a rough disc. Then there are exactly two positions where bricks of ∂R are connected to each other (the cells share a common side). One occurs at the starting location S , and its southern neighbor E . The other is at P (the previously placed brick of set ∂R) and its neighbor O . Let N be the empty cell (where the next brick will be placed) adjacent to O , such that N is on the right side of $\overrightarrow{PO}$. Note that $P = S$ or $O = E$ are possible. If $P = S$ then we started a new layer with the previous brick. If $O = E$ then we complete a disc layer after placing the brick at N . The pairs E, S and P, O of bricks on the boundary of the nest are identified during the boundary traversal by moving on top of the bricks. When the robot encounters a connected pair, it checks whether it corresponds to the starting position S, E . If so, the robot continues traversing until it finds the pair P, O . Otherwise, the pair P, O is localized.

3.3. Distance-Aware Sweeping

The SWEEP procedure maintains the gap invariant: it ensures that all bricks belonging to a free component C are at a Manhattan distance of at least 7 from the rough disc D . In [4], the original algorithm maintains the gap by requiring the robot to perform a full counterclockwise traversal of the boundary of the nest after every brick placement, checking every cell $c \notin D \cup M$ (M is the marker) within a radius of 7 from its current position.

Our optimization is based on the geometric properties of the free components. Let D be the current nest, which is a rough disc. For D , the region D_{+7} is defined by a Manhattan distance of ≤ 7 from D . $D_{+7} \setminus D$ forms a swept region, which must not contain any bricks. Upon extending the nest by placing a new brick at the cell b , new cells will be included in the swept region, which might contain bricks that have to be relocated. Mathematically, if the nest is extended to $D' = D \cup \{b\}$, then the new distance from any cell c to the nest is defined as $d(c, D') = \min(d(c, D), d(c, b))$. This implies that the invariant $d(c, D') > 7$, for any brick $c \notin D'$, can only be violated if $d(c, b) \leq 7$. All such bricks are visible from b . This means that it is sufficient to check a constant number of cells around b instead of traversing the whole boundary for maintaining the invariant. This significantly reduces the time for each sweep operation.

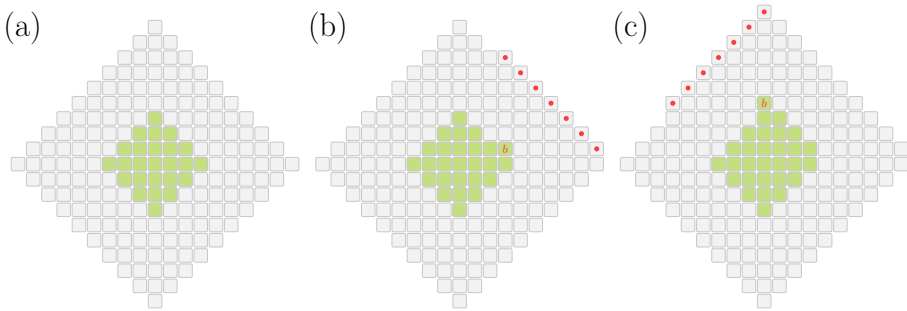


Figure 2. Illustration of the optimized sweeping process, which maintains the gap around nest D , such that $d(c, D) > 7$ for any brick $c \notin D$. (a) The initial nest is surrounded by a swept region where no bricks are permitted. (b) and (c) The nest is extended by placing a new brick at the cell b . Only a constant number of cells (indicated by dots) need to be checked to see whether they contain bricks.

When a new brick is placed at b , only cells c with $d(c, b) = 7$ need to be checked to see if they contain a brick, since any brick which is not in D is of a distance at least 8 from D before extending the nest with b .

4. Experimental Evaluation and Conclusion

To empirically validate the proposed optimizations, we conducted a series of systematic tests using the *NestBuilder* simulator across a diverse range of initially connected fields. The experimental setup involved 100 randomly generated connected configurations, where starting shapes were generated via Kruskal’s algorithm with randomized weights to facilitate a rigorous stress test of the ability of the robot to navigate complex configurations while maintaining connectivity. To ensure that our results are independent of the specific shape and scale of the initial configuration, we evaluated several types of initial shapes (tree, line, spiral) within a grid of three sizes: 10, 20, and 40. The total number of bricks varied depending on the types of the shapes. For sparse configurations, such as tree and line, the number of bricks ranged from 21 to 47 for the 10×10 grid, from 48 to 95 for the 20×20 grid, and from 111 to 165 for the 40×40 grid. The dense spiral (more precisely, an Archimedean spiral according to the L_∞ metric, consisting of vertical and horizontal line segments, with a line width of 1 and a gap between the lines of 1) contained 55, 231, and 861 bricks in the corresponding grids. The data presented reflect the average performance of all experimental runs.

4.1. Evaluation of Optimized Rough Disc Extension

The algorithm described in subsection 3.2 was tested against the original boundary-traversal method in [4].

Our method achieved an improvement of 18.75% in the number of steps during the nest extension phase, see Figure 3a. Specifically, the average number of movements in the EXTENDROUGHDISC procedure was reduced from an average of 48 to an average 39 steps.

4.2. Evaluation of Distance-Aware Sweeping

The Distance-Aware Sweeping method provided the most substantial performance improvement by utilizing the geometric property that the swept region around the nest D is only affected by the cell b , where the new brick is placed. While the original algorithm in [4] required a full boundary traversal of the nest D , which takes $O(n)$ steps, and checking the cells within a distance of 7 cells in every step of the boundary traversal, our method only needs to check $O(1)$ cells around the new brick placed at b .

This improvement resulted in a massive 92% reduction in the number of steps, see Figure 3b. In the sweep phase, the average number of movement steps decreased from an average of 1082 steps to an average of 83. This result shows that our approach effectively eliminates redundant examination of cells that could not have changed since the last iteration.

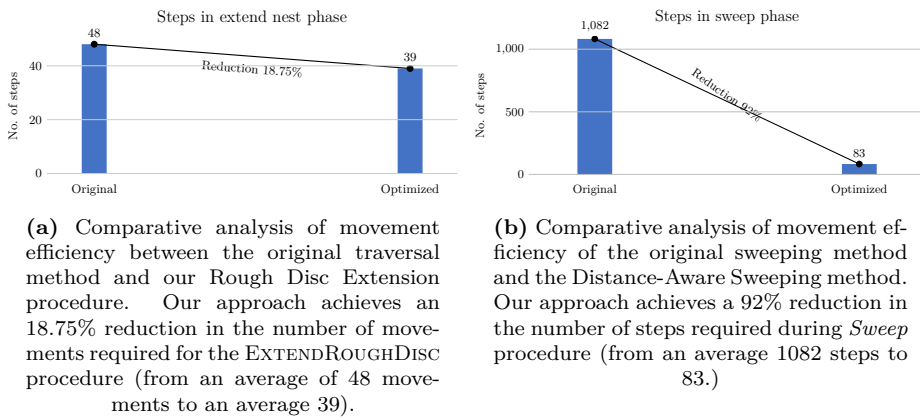


Figure 3. Experimental Evaluation.

5. Conclusion and Future Work

This work presents empirical practical improvements of the algorithm by Czyzowicz et al. [4] for the construction of a nest by an autonomous robot modeled as a finite

automaton. The original algorithm has a running time of $O(s \cdot n)$, which has been shown to be asymptotically optimal for this model. Through the development of the *NestBuilder* simulator and modifications of the original algorithm, we have demonstrated that significant improvements can be achieved in runtime without sacrificing the memory requirements of a deterministic finite automaton.

Future research:

- **Target Structure Diversity:** We aim to characterize which other classes of target shapes can be built starting from any connected shape.
- **Parallelization Challenges:** The model is highly effective for single-robot nest construction, but has a high limitation for the marker-based system for multi-robot coordination. It is a challenging task to develop algorithms for multiple robots that enable collaborative nest building without destroying the connectivity of the free components.

References

- [1] A. T. BECKER, S. P. FEKETE, P. KELDENICH, D. KRUPKE, C. RIECK, C. SCHEFFER, A. SCHMIDT: *Tilt Assembly: Algorithms for Micro-factories That Build Objects with Uniform External Forces*, Algorithmica 82.2 (2020), pp. 165–187, doi: [10.1007/S00453-018-0483-9](https://doi.org/10.1007/S00453-018-0483-9).
- [2] M. K. BEN-LARBI, K. F. POZO, T. HAYLOK, M. CHOI, B. GRZESIK, A. HAAS, D. KRUPKE, H. KONSTANSKI, V. SCHAUS, S. P. FEKETE, C. SCHURIG, E. STOLL: *Towards the automated operations of large distributed satellite systems. Part 1: Review and paradigm shifts*, Advances in Space Research 67.11 (2021), pp. 3598–3619, doi: [10.1016/j.asr.2020.08.009](https://doi.org/10.1016/j.asr.2020.08.009).
- [3] J. CZYZOWICZ, D. DERENIOWSKI, A. PELC: *Building a Nest by an Automaton*, in: 27th Annual European Symposium on Algorithms, ESA 2019, 2019, 35:1–35:14, doi: [10.4230/LIPICs.ESA.2019.35](https://doi.org/10.4230/LIPICs.ESA.2019.35).
- [4] J. CZYZOWICZ, D. DERENIOWSKI, A. PELC: *Building a Nest by an Automaton*, Algorithmica 83.1 (2021), pp. 144–176, doi: [10.1007/S00453-020-00752-0](https://doi.org/10.1007/S00453-020-00752-0).
- [5] S. P. FEKETE, E. NIEHS, C. SCHEFFER, A. SCHMIDT: *Connected assembly and reconfiguration by finite automata*, arXiv preprint arXiv:1909.03880 (2019), doi: [10.1007/s00453-022-00995-z](https://doi.org/10.1007/s00453-022-00995-z).
- [6] S. P. FEKETE, R. GMYR, S. HUGO, P. KELDENICH, C. SCHEFFER, A. SCHMIDT: *CADbots: Algorithmic Aspects of Manipulating Programmable Matter with Finite Automata*, Algorithmica 83.1 (2021), pp. 387–412, doi: [10.1007/S00453-020-00761-Z](https://doi.org/10.1007/S00453-020-00761-Z).
- [7] S. P. FEKETE, E. NIEHS, C. SCHEFFER, A. SCHMIDT: *Connected Reconfiguration of Lattice-Based Cellular Structures by Finite-Memory Robots*, Algorithmica 84.10 (2022), pp. 2954–2986, doi: [10.1007/S00453-022-00995-Z](https://doi.org/10.1007/S00453-022-00995-Z).
- [8] R. GMYR, K. HINNENTHAL, I. KOSTITSYNA, F. KUHN, D. RUDOLPH, C. SCHEIDELER: *Shape recognition by a finite automaton robot*, in: 43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018, pp. 52–1, doi: [10.4230/LIPICs.MFCS.2018.52](https://doi.org/10.4230/LIPICs.MFCS.2018.52).
- [9] R. GMYR, K. HINNENTHAL, I. KOSTITSYNA, F. KUHN, D. RUDOLPH, C. SCHEIDELER, T. STROTHMANN: *Forming tile shapes with simple robots*, Natural Computing 19.2 (2020), pp. 375–390, doi: [10.1007/s11047-019-09774-2](https://doi.org/10.1007/s11047-019-09774-2).

- [10] W. HU, G. Z. LUM, M. MASTRANGELI, M. SITTI: *Small-scale soft-bodied robot with multi-modal locomotion*, Nature 554.7690 (2018), pp. 81–85, DOI: [10.1038/nature25443](https://doi.org/10.1038/nature25443).
- [11] B. JENETT, C. GREGG, D. CELLUCCI, K. CHEUNG: *Design of multifunctional hierarchical space structures*, in: Aerospace Conference, IEEE, 2024, pp. 1–10, DOI: [10.1109/AERO.2017.7943913](https://doi.org/10.1109/AERO.2017.7943913).
- [12] E. NIEHS, A. SCHMIDT, C. SCHEFFER, D. E. BIEDIGER, M. YANNUZZI, B. JENETT, A. ABDEL-RAHMAN, K. C. CHEUNG, A. T. BECKER, S. P. FEKETE: *Recognition and Reconfiguration of Lattice-Based Cellular Structures by Simple Robots*, in: International Conference on Robotics and Automation, (ICRA), IEEE, 2020, pp. 8252–8259, DOI: [10.1109/ICRA40945.2020.9196700](https://doi.org/10.1109/ICRA40945.2020.9196700).
- [13] G. PUSZTAI, T. LUKOVSKI: *Multi-Robot Simulation Framework for Construction Problems*, in: Proceedings of the Intelligent Robotics FAIR 2025, IntRob '25, Association for Computing Machinery, 2025, pp. 124–131, ISBN: 9798400715891, DOI: [10.1145/3759355.3759631](https://doi.org/10.1145/3759355.3759631).



An Improved Measurement and Visualization System for High Precision Positioning with GPS-RTK

Péter Ákos Makár^a, Gergely Kocsis^{a*}

^aUniversity of Debrecen, Faculty of Informatics,
Department of IT Systems and Networks
maker.peter98@gmail.com
kocsis.gergely@inf.unideb.hu

Abstract. In this work we present a GPS measurement and visualization system that enables real-time tracking, saving, and analysis of measurement data. We performed measurements using a u-blox GNSS receiver (Global Navigation Satellite System) module connected to a **Raspberry Pi 5** micro-computer, to determine whether the out-of-the-box solution can be used for high precision GPS positioning that usually means centimeter level in RTK (Real-Time Kinematic) applications. Besides using the built-in RTK correction of the module we have also applied our own algorithm to further increase accuracy based on repeated measurements. Our test measurements showed that by the use of our algorithm in an optimal setup positioning precision can be increased up to 2-3 cm. We provided a modular user interface for the device supporting graphical visualization, database management and point separation. Based on repeated measurements, the system reproduces the coordinates of measurement locations. Overall, the software assists in handling GPS data and analyzing information collected during experiments. The primary goal is to maximize measurement accuracy and simplify the experimental workflow.

Keywords: GPS, RTK, Positioning software

2020 Mathematics Subject Classification: 00A69, 00A79

*Gergely Kocsis was supported by the project TKP 2021 NKTA of the University of Debrecen Project no TKP 2021 NKTA 34 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the TKP 2021 NKTA funding scheme.

1. Introduction

Real-Time Kinematic (RTK) high precision positioning is widely used in several practical areas [6, 10]. However, its current applications are not viable for low-cost home usage. As a result, its applications focus more on high level agricultural usage. As correction data becomes increasingly accessible, everyday usage of high-precision positioning becomes available. Recently, collaborative low-cost RTK networks such as the Centipede-RTK network have gained attention in the research community [9]. These networks consist of distributed low-cost GNSS base stations providing open RTK correction data and enabling centimeter-level positioning for a wide range of applications. Previous studies investigated the reliability of low-cost GNSS receivers using the Centipede network and demonstrated that such systems can be suitable for land surveying and scientific applications.

The rapid development of low-cost multi-frequency GNSS receivers also contributed to the spread of RTK technology in recent years. Modern receivers are capable of tracking multiple satellite constellations such as GPS, GLONASS, Galileo and BeiDou [4], which improves positioning availability and reliability. This makes low-cost RTK positioning a viable solution not only for agriculture but also for robotics, autonomous vehicles, environmental monitoring and various scientific measurements.

Our first objective is to show how a relatively low-cost RTK device can do positioning while we also try to further improve the positioning accuracy this way. Our results may be used for validating the results of [2] and [9]. Besides we also visualize, store, and analyze data obtained from an RTK GNSS measurement device. As a result of our contribution one can use similar devices as ours out-of-the-box for high level accuracy positioning with even better results than the original software solutions would make possible.

In addition to positioning accuracy, measurement repeatability and data logging are also important aspects of RTK measurements. Therefore, in this work we also focus on measurement data storage, visualization and basic statistical analysis in order to better understand positioning errors and system performance.

2. Methodology

2.1. Measurement Workflow and Software

GNSS positioning has proven to be effective in applications where meter-level accuracy is sufficient. On its own, however, it cannot compensate for atmospheric disturbances, multipath effects, and other error sources, which fundamentally limit its achievable precision.

RTK (Real-Time Kinematic) improves GNSS accuracy by incorporating real-time correction data from a continuously operating reference station [5]. The basic idea behind the technology is that the above mentioned effects limiting the accuracy are almost similar in between distances of 30-50 km. Thus correction data

calculated at a fixed base station can effectively be used to tune a moving rover's location data in this radius. In this configuration, the rover receives both the raw satellite observations and the correction stream, enabling it to resolve carrier-phase ambiguities and achieve centimeter-level positioning as described by [11].

In our measurement system (see Figure 1) all computations are performed directly on the rover-side **Raspberry Pi 5**. The Raspberry Pi microcomputer communicates with the **u-blox ZED-F9P-04B-01** receiver, obtains raw GNSS data, and simultaneously retrieves correction messages from the appropriate RTK base station (e.g., **fmtp** in Debrecen or **jrm1** in Mátészalka). The Raspberry Pi fuses these data streams in real time and produces the corrected position solution.

External devices such as a smartphone or laptop connect only for control and monitoring purposes, for example to start or stop measurements, adjust configuration parameters, or inspect quality indicators. They do not participate in the positioning computation itself.

The workflow diagram reflects this architecture: satellite signals reach the RTK receiver, the receiver forwards raw observations to the Raspberry Pi, the Raspberry Pi receives correction data over the Internet, and the corrected position is computed locally on the rover.

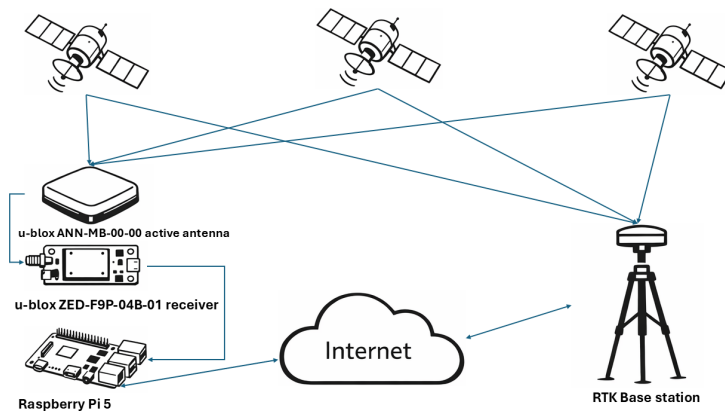


Figure 1. RTK GNSS correction flow diagram in minimalist technical style.

The possible weak point of the system naturally is the availability of RTK correction base stations. The Centipede RTK network is a community-driven GNSS correction network that provides low-cost RTK positioning. It was founded in France around 2019, but today it operates through a distributed network of privately operated base stations that share correction data over the internet, offering an open alternative to commercial RTK services for applications such as precision agriculture, surveying, and robotics [1]. Hungary has a very good coverage of base stations resulting almost 100% availability (see Figure 2). For our measurements in most of the cases we used the base station maintained by University of Debrecen,

Faculty of Agricultural and Food Sciences and Environmental Management that has been started operating at March 2025.

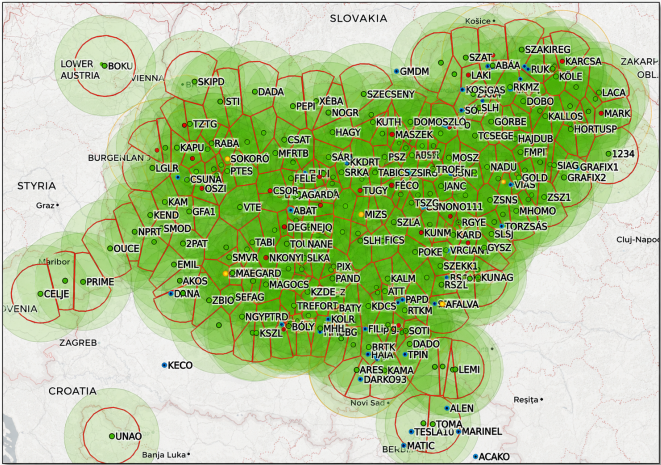


Figure 2. Hungary’s Centipede-RTK base station coverage. Red lines show the max. 30 km radii of base stations.
(Source: <https://www.centipede-rtk.org/maps>)

2.2. Hardware setup

Our measurement system is built on reliable, industry-grade GNSS hardware. The RTK processing is performed by the *u-blox* ZED-F9P-04B-01 receiver, a high-precision, multi-frequency GNSS module capable of centimeter-level positioning according to its documentation. The *u-blox* ANN-MB-00-00 active antenna provides stable multi-band reception, ensuring consistent signal quality even in challenging environments [7].

The antenna connects to the RTK receiver through a standard coaxial RF cable, which minimizes signal loss and preserves the integrity of the multi-band GNSS signals. The RTK module itself interfaces with the *Raspberry Pi 5* via a USB–micro-USB cable, enabling both power delivery and high-speed data transfer without requiring additional interface hardware. The *Raspberry Pi* provides sufficient computational performance for real-time visualization, database operations, and communication with the GNSS hardware, making the entire setup compact and easily deployable.

For network-based RTK corrections, the choice of reference station depends on the measurement location. In Debrecen, the appropriate correction source is the “fmtp” station, while in Mátészalka the “jrmi” station must be used. Although many services offer a “nearest” virtual reference option, this mode relies on a synthetic baseline and cannot deliver the centimeter-level accuracy required for high-precision geodetic measurements.

2.3. Data correction algorithm

As we will present below, according to our measurements the initial results have not reached the aimed centimeter level accuracy. Thus, we have used a simple correction algorithm to make them more accurate.

Our measurement algorithm selects the coordinates with the smallest error from a set of pre-collected GPS points noted by lat_{best} and lon_{best} respectively for the latitude and longitude. Since in our current measurements they are not needed, height and velocity values are simply averaged. We formalize the above as:

$$lat_{best} = lat + \min_i \sigma_{lat}(i), \quad lon_{best} = lon + \min_i \sigma_{lon}(i), \quad (2.1)$$

$$alt_{avg} = \frac{1}{N} \sum_{i=1}^N alt(i), \quad v_{avg} = \frac{1}{N} \sum_{i=1}^N v(i),$$

where $\sigma_{lat}(i)$ and $\sigma_{lon}(i)$ are the accuracy errors in meters of the latitude and the longitude for the measurement i . Since the device remains stationary during a measurement session, the reference point is the same for all collected data. These values are directly received from our measurement device. The total 3D error H_{3D} is:

$$H_{3D} = \sqrt{(\sigma_{lat})^2 + (\sigma_{lon})^2 + (\sigma_{alt})^2}.$$

Note that according to Eq. (2.1) the latitude and longitude errors are optimized separately. This method simplifies the correction process much compared to trying to find a measurement point with minimal global error. On the other hand, since the error values are strongly correlated in most cases, the same measurement point will typically be selected for both coordinates. To prove the above we ran a test measurement with 1043 measurement points (see Figure 3). As a result of the strong correlation and the fact that the final values of σ_{lat} and σ_{lon} are chosen from 5 independent measurements the probability of choosing the same measurement point for both values is above 99%.

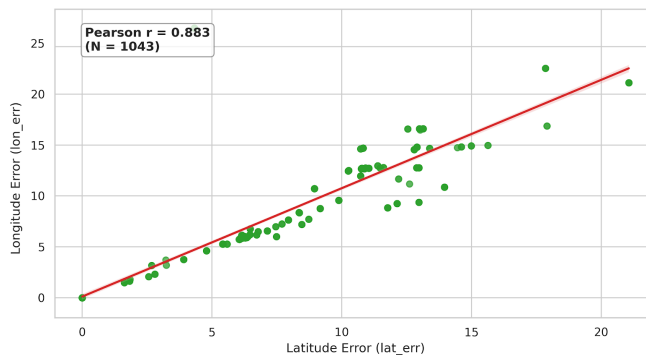


Figure 3. Correlation of σ_{lat} and σ_{lon} .

The above simple correction can indeed increase positioning accuracy, as repeated measurements confirmed that the actual positions lie very close to each other, although the estimated error is typically exaggerated (the real deviation is approximately 5 cm).

2.4. Indoor and Outdoor Accuracy

Similarly to simple GNSS positioning, RTK is also based on satellite data received from one or more supported systems. The problem with this is that the environment highly affects the final results. Weather conditions, signal obstruction and multipath propagation can significantly degrade positioning accuracy, particularly in urban and forested environments [3]. In order to demonstrate this, we performed measurements in different scenarios.

During field tests, ten reference points were marked one meter apart. For each round, 50–100 samples were collected while moving between the points. The results were then summarized to evaluate accuracy under varying conditions.

2.5. Graphical User Interface

In order to easily be able to perform bulk measurements, we have developed a tool in Python with a graphical user interface. A screenshot of the interface is presented on Figure 4.

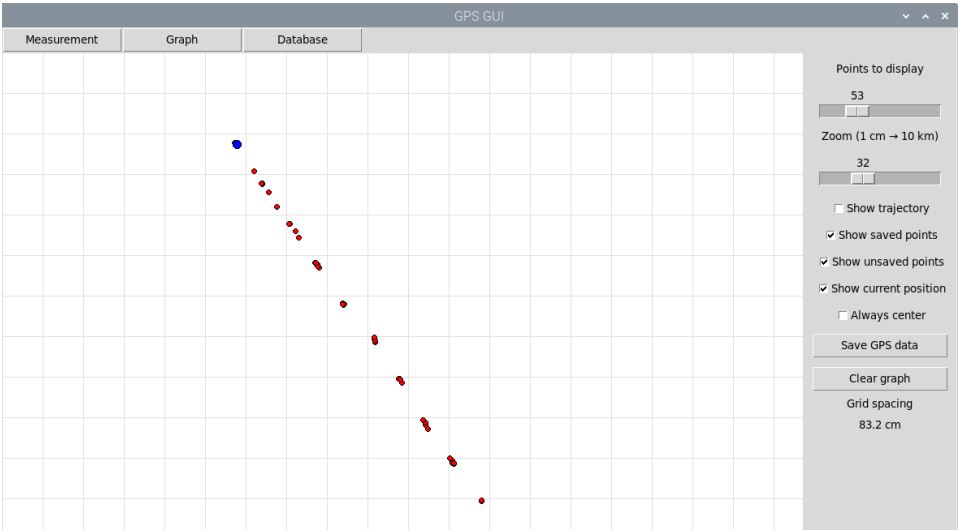


Figure 4. GUI showing measured points, trajectory, and real-time tracking.

The software automatically registers measurement data and persists it using a database. In the database each item describes a measurement registering the below

attributes:

```
(id, timestamp, latitude, longitude,  
latitude_error, longitude_error,  
speed, mode, measurement iteration)
```

The graphical interface displays the measurement points collected during the experiment. The red points represent the measured GPS samples, placed exactly one meter apart on the ground. The isolated point is the control point used for reference. The software correctly visualizes the spatial layout of the points, confirming that the measurement grid was reconstructed accurately.

All data presented in the below section are measured using this tool and the previously described hardware setup.

3. Results

To empirically show the functionality of our system we have performed measurements in the following three environments:

1. Indoor measurements were performed in a flat room in Debrecen residential area.
2. A suboptimal outdoor setup took place at Nagyerdő (Debrecen) that is a classical recreation area of Debrecen city with several high trees and other obstacles.
3. Optimal outdoor measurements were performed in the park of the Faculty of Informatics, University of Debrecen. This relatively new park does not have too many high trees, moreover the measurements were registered in late autumn. In most of the cases the sky was clear.

We have repeated the measurements with simple GNSS positioning, core RTK correction and RTK correction extended with our correction algorithm. The results are summarized on Figures 5, 6 and 7. On the figures it is easy to see how RTK can increase accuracy of GPS positioning in outdoor scenarios. Interestingly, our simple correction algorithm can further increase this accuracy besides really low computational cost. The indoor scenario however proved to be unfeasible for all three positioning variants since no correction can solve the issue of unsatisfactory amount of arriving location data.

Due to the above mentioned properties of GPS-based positioning, pure GPS indoor measurements without any correction showed large deviations (up to 4 meters), however RTK correction reduced this to approximately 50 cm. With our filtering method, accuracy could be improved to around 35 cm.

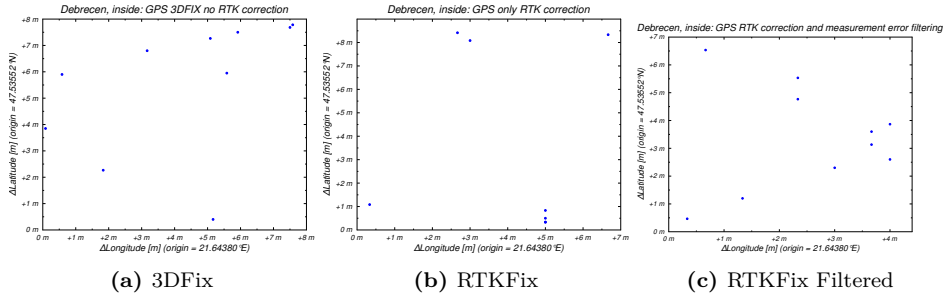


Figure 5. GPS positioning results in indoor scenario.

In the Nagyerdő park area, despite unstable signals, the pure GPS positioning showed around 70 cm accuracy that coincides with literature values [8]. RTK correction improved this to around 20 cm, while the full system reached 10 cm accuracy with filtering.

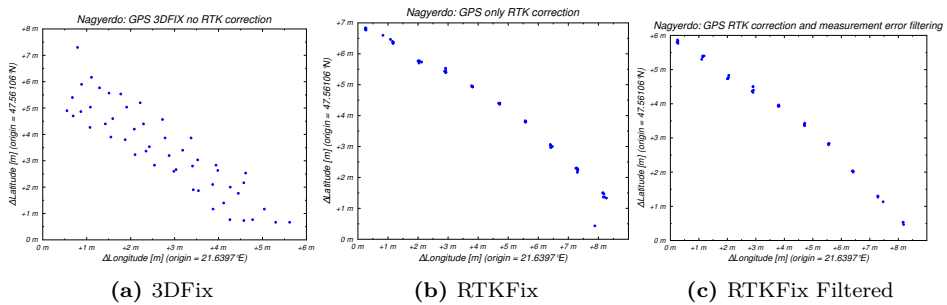


Figure 6. GPS positioning results in suboptimal outdoor scenario.

In optimal outdoor environment measurements yielded significantly better results. At the Faculty of Informatics University of Debrecen, even the pure GPS reached around 30 cm accuracy. RTK correction improved this to ~ 10 cm. Finally the corrected and filtered method achieved 2–3 cm accuracy.

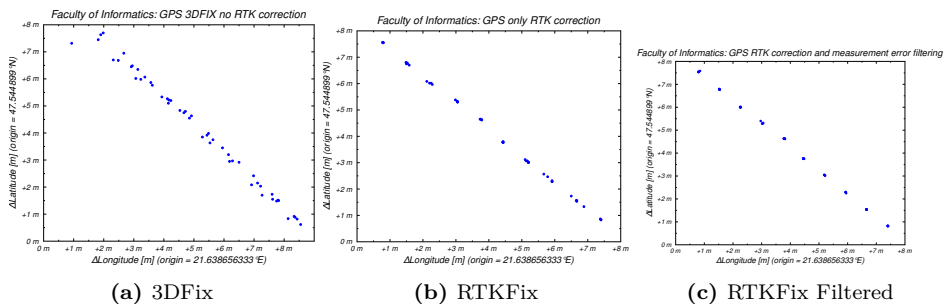


Figure 7. GPS positioning results in optimal outdoor scenario.

Figure 8 sums up our measured accuracy results. The bars indicate the average accuracy.

Initial measurements showed errors of 100-170 cm, while the improved algorithm typically achieves around 70 cm, with best cases reaching 30-40 cm. Measurements were performed in Debrecen under various environmental conditions (forest, open sky, cloudy and clear weather). In order to further improve accuracy we plan refinement of the algorithm, particularly the averaging of coordinates while incorporating error metrics.

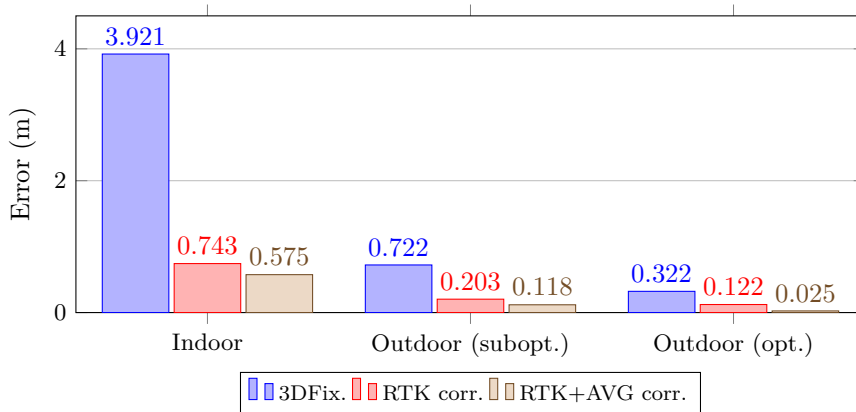


Figure 8. Comparison of GPS measurement errors across locations and correction methods. Our simple correction algorithm can further increase RTK accuracy.

3.1. Conclusion

This paper presented a measurement and visualization system for high-precision positioning based on a low-cost RTK GNSS setup. The system integrates GNSS hardware, real-time data acquisition, visualization, and database management into a unified software environment. The primary objective was to evaluate the achievable positioning accuracy of an out-of-the-box RTK solution and to investigate whether simple post-processing methods can further improve accuracy.

Experimental results demonstrated that RTK correction significantly improves positioning accuracy compared to standalone GNSS positioning, particularly in outdoor environments with good satellite visibility. Furthermore, the proposed filtering and measurement selection method provided additional accuracy improvement with minimal computational overhead. In optimal outdoor conditions, the system achieved centimeter-level positioning accuracy, while suboptimal environments resulted in decimeter-level accuracy. Indoor measurements confirmed the known limitations of satellite-based positioning due to signal attenuation and multipath effects.

The results indicate that reliable high-precision positioning can be achieved

using relatively low-cost hardware when combined with appropriate measurement strategies and data processing methods. Future work will focus on advanced filtering techniques, statistical error modeling, and long-term measurements under varying environmental conditions.

References

- [1] P. BOSSER, J. ANCELIN, M. MÉTOIS, L. ROLLAND, M. VIDAL: *Evaluation of tropospheric estimates from CentipedeRTK, a collaborative network of low-cost GNSS stations*, GPS Solutions 28(158) (2024), DOI: [10.1007/s10291-024-01699-3](https://doi.org/10.1007/s10291-024-01699-3).
- [2] P. DABOVE, M. BAGHERI: *Enhancing Atmospheric Monitoring Capabilities: A Comparison of Low- and High-Cost GNSS Networks for Tropospheric Estimations*, Remote Sensing 16(12): 2223 (2024), DOI: [10.3390/rs16122223](https://doi.org/10.3390/rs16122223).
- [3] V. HAMZA, B. STOPAR, O. STERLE, P. PAVLOVČIČ-PREŠEREN: *Observations and positioning quality of low-cost GNSS receivers: a review*, GPS Solut. 28: 149 (2024), DOI: [10.1007/s10291-024-01686-8](https://doi.org/10.1007/s10291-024-01686-8).
- [4] B. HOFMANN-WELLENHOF, H. LICHTENEGGER, E. WASLE: *GNSS – Global Navigation Satellite Systems, GPS, GLONASS, Galileo, and More*, 1st ed., Vienna, Austria: Springer Vienna, 2008, ISBN: 978-3-211-73012-6, DOI: [10.1007/978-3-211-73017-1](https://doi.org/10.1007/978-3-211-73017-1).
- [5] P. MISRA, P. ENGE: *Global Positioning System: Signals, Measurements, and Performance*, Ganga-Jamuna Press, 2011, ISBN: 978-0-970-95442-8.
- [6] L. RADÓCZ, C. JUHÁSZ, A. TAMÁS, Á. ILLÉS, P. RAGÁN, L. RADÓCZ: *Multispectral UAV-Based Disease Identification Using Vegetation Indices for Maize Hybrids*, Agriculture 14(11): 2002 (2024), DOI: [10.3390/agriculture14112002](https://doi.org/10.3390/agriculture14112002).
- [7] U. ROBUSTELLI, M. CUTUGNO, G. PUGLIANO: *Low-Cost GNSS and PPP-RTK: Investigating the Capabilities of the u-blox ZED-F9P Module*, Sensors 23(13): 6074 (2023), DOI: [10.3390/s23136074](https://doi.org/10.3390/s23136074).
- [8] M. H. SAKA, T. KAVZOGLU, O. C., R. M. ALKAN: *Sub-Metre Accuracy for Stand Alone GPS Positioning in Hydrographic Surveying*, Journal of Navigation 57(1) (2004), pp. 135–144, DOI: [10.1017/S0373463303002571](https://doi.org/10.1017/S0373463303002571).
- [9] M. A. SAMMUNEH, M. ESLAHI, R. E. MEUCHE, E. FARAZDAGHI: *Low-Cost GNSS Receivers Reliability Using Centipede RTK Network for Land Surveying*, in: Proceedings of the 11th International Conference on Geographical Information Systems Theory, Applications and Management - GISTAM, 2025, pp. 211–218, DOI: [10.5220/0013444800003935](https://doi.org/10.5220/0013444800003935).
- [10] A. SUKHENKO, N. MEIRAMBEKULY, A. SYZDYKOV, A. MUKHAMEDGALI, Y. MELLATOVA: *GNSS for High-Precision and Reliable Positioning: A Review of Correction Techniques and System Architectures*, Applied Sciences 15(22): 12304 (2025), DOI: [10.3390/app152212304](https://doi.org/10.3390/app152212304).
- [11] P. J. G. TEUNISSEN, P. J. DE JONGE, C. C. J. M. TIBERIUS: *The least-squares ambiguity decorrelation adjustment: its performance on short GPS baselines and short observation spans*, Journal of Geodesy 71 (1997), pp. 589–602, DOI: [10.1007/s001900050127](https://doi.org/10.1007/s001900050127).



AI-Based Estimation of the Zone of Proximal Development in Programming Education

Bendegúz Nagy, Győző Horváth

Department of Media & Educational Informatics,
Eötvös Loránd University, Budapest, Hungary

brandy@inf.elte.hu

gyozke@inf.elte.hu

Abstract. The *Zone of Proximal Development* (ZPD), introduced by Vygotsky [11], describes the range of learning tasks that learners can successfully complete with appropriate support. Closely related to this concept, the notion of flow experience emphasizes the balance between task challenge and learner skill as a key condition for sustained engagement and intrinsic motivation [1].

Recent research suggests that artificial intelligence (AI) can support ZPD-oriented adaptation in modern educational environments by dynamically adjusting task difficulty and feedback [10]. However, practical and lightweight implementations of such approaches in programming education remain limited.

This paper presents a proof-of-concept AI-based system for estimating learners' ZPD using a minimal set of background information and task interaction data. The approach records simplified learner profiles from task interaction data – task success, error frequency, and completion time – and derives a skill estimate from task success and difficulty; this estimate defines a difficulty band that a large language model uses to generate subsequent programming tasks at an adjusted difficulty level.

The system is developed as part of the continued evolution of the *Codia* gamified programming education platform, building on earlier findings related to student motivation in secondary school programming education [6]. The design focuses on feasibility and pedagogical plausibility rather than predictive accuracy. The study examines whether a simplified AI-supported approach can approximate ZPD-aligned task selection in introductory programming education.

Keywords: zone of proximal development, adaptive learning, ELO rating, large language models, programming education, task generation

2020 Mathematics Subject Classification: Primary 97P30; Secondary 68T50

1. Introduction

Programming education at the secondary school level faces a structural challenge: teachers must design tasks for an entire class, yet individual students differ substantially in prior experience, cognitive pace, and intrinsic motivation. Assigning identical tasks to all learners inevitably places some students below their challenge threshold – risking boredom and disengagement – and others above it – risking frustration and dropout. Addressing this mismatch requires both a theoretical model of optimal challenge and a practical mechanism for adapting task difficulty to individual learners in real time.

Vygotsky’s concept of the *Zone of Proximal Development* (ZPD) provides exactly such a model [11]. The ZPD denotes the gap between what a learner can achieve independently and what they can achieve with guidance; instructional tasks that fall within this zone maximise learning efficiency. Closely related is Csikszentmihalyi’s theory of *flow*: sustained engagement arises when both task difficulty and individual skill are simultaneously high [1]. Together, ZPD and flow theory motivate an educational design in which each student receives tasks calibrated to their evolving competence.

Recent developments in large language models (LLMs) open a practical path towards such personalisation: LLMs can generate contextually appropriate programming tasks conditioned on skill level, topic preferences, and prior performance [9], yet lightweight end-to-end implementations in introductory programming remain scarce. The proof-of-concept system described here, *Codia Path*, is built using the T3 Stack (Next.js, tRPC, Prisma, PostgreSQL) with React Flow for graph visualisation and supports OpenAI (o4-mini), Anthropic Claude, and Google Gemini. Throughout, we distinguish two systems: *Codia* is the deployed platform used in the classroom pilot (Section 3), whereas *Codia Path* is the separate proof-of-concept adaptive system evaluated here on simulated data (Section 5); the pilot motivates but does not validate *Codia Path*.

This paper makes the following contributions:

- We report the results of a real-world pilot study with the *Codia* gamified programming platform, motivating a shift from extrinsic-reward-based gamification to ZPD-centred task design [6].
- We describe the design of *Codia Path*, a proof-of-concept system that combines a graph-based curriculum model, an ELO-inspired ZPD estimator, and LLM-based task generation to produce personalised programming tasks.
- We illustrate the system using three simulated learner trajectories and qualitatively assess the pedagogical plausibility of the generated tasks; we make no claim of improved learning outcomes.

2. Background and Related Work

Vygotsky's *Zone of Proximal Development* (ZPD) denotes the distance between what a learner can achieve independently and what they can achieve with guidance [11]. In a software-based learning environment the ZPD can be operationalised through measurable proxies – response accuracy, error count, and time-on-task – all of which are naturally observable in a programming IDE. Closely related is Csikszentmihalyi's *flow* theory: optimal experience arises when both perceived challenge and perceived skill are high [1], a finding confirmed by meta-analytic evidence [3]; when this balance is absent, the result is either boredom (low challenge) or anxiety (excessive challenge) [7].

Gamification has been widely studied as a motivational tool in education. Hamari et al. [4] report generally positive effects on engagement, but note that outcomes are greatly dependent on context. Koivisto and Hamari [5] confirm that mixed results are remarkably common, underscoring the importance of design quality and situational fit. Palmquist [8] further shows that the success of gamification in education is mediated by factors such as teacher buy-in and implementation quality.

AI-supported adaptive learning ranges from classical intelligent tutoring systems [10] to modern LLM-based approaches. Sarsa et al. [9] demonstrate automatic generation of programming exercises and code explanations using LLMs, though rigorous classroom evaluation of such methods remains limited. Yang et al. [12] combine LLM-derived difficulty features with deep knowledge tracing, improving difficulty prediction accuracy in programming education. Our work focuses on *lightweight* operationalisation: Codia Path records per-attempt interaction data (success, error count, and completion time) and derives a self-calibrating skill estimate from task success and difficulty using an ELO-inspired rating rule [2]; the resulting difficulty band is then supplied to an LLM prompt that generates the next task.

3. The Codia Platform and Pilot Study

3.1. Platform Description

Codia (available at <https://codia.hu>) is a web-based, mobile-optimised programming education platform for introductory Python programming at the secondary school level. Its main features include:

- **Online IDE:** A browser-based Python interpreter optimised for mobile and desktop use.
- **AI code feedback (Codie):** An LLM-based assistant that detects errors and provides hints (Figure 1).

- **Gamification:** A diamond-based reward system where correct solutions earn virtual currency exchangeable for rewards.
- **Teacher dashboard:** Task assignment creation, progress monitoring, and group management.

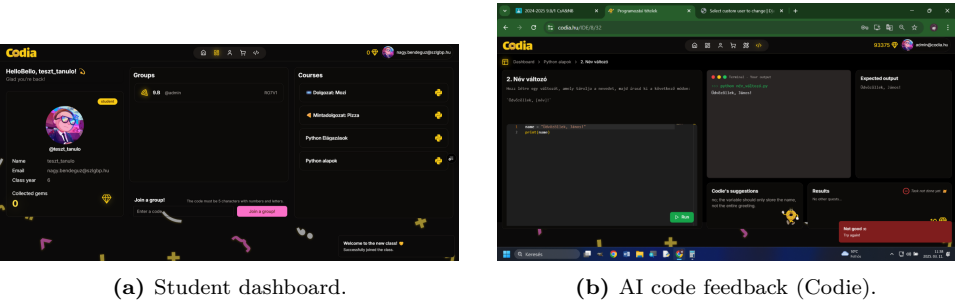


Figure 1. The Codia programming education platform.

3.2. Pilot Study

We conducted a five-week deployment of Codia at *Kőbányai Szent László Gimnázium*, Budapest, with 17 ninth-grade students [6]. Students solved teacher-assigned tasks in the online IDE and received AI feedback from Codie. At the end of the study, 12 students completed a Likert-scale questionnaire (Figure 2).

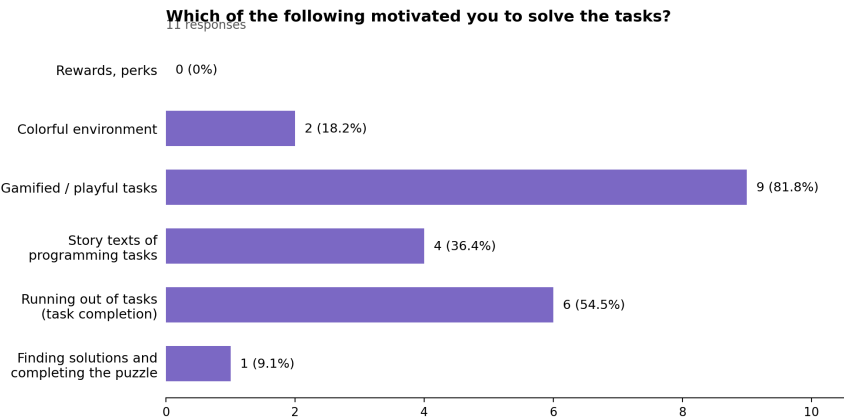


Figure 2. Student attitudes toward programming and Codia (n = 12).

Key findings: 10/11 students preferred Codia over prior coding environments; Codie’s feedback helped students self-correct code independently; the reward/dia-

mond system was rarely used; and not a single student selected rewards as their primary motivator – the dominant motivators were *problem-solving challenge* and *task completion*. These results, consistent with flow theory [7], directly motivated the design of Codia Path.

4. System Design: Codia Path

Codia Path is a proof-of-concept web application built on top of the data infrastructure of Codia. It implements three interconnected subsystems: (1) a graph-based curriculum model, (2) an ELO-based ZPD estimator, and (3) an LLM-driven task generator.

4.1. Graph-Based Curriculum Model

Rather than presenting courses as a flat list, Codia Path models the curriculum as a *directed acyclic graph* (DAG) of courses. A directed edge from course *A* to course *B* indicates that *A* is a prerequisite for *B*. The current curriculum graph covers eight courses, from *Variables & Types* to *Recursion & Algorithms*. In the future, the topics and competence areas defined in the Hungarian National Core Curriculum (NAT) could serve as a solid basis for structuring these courses.

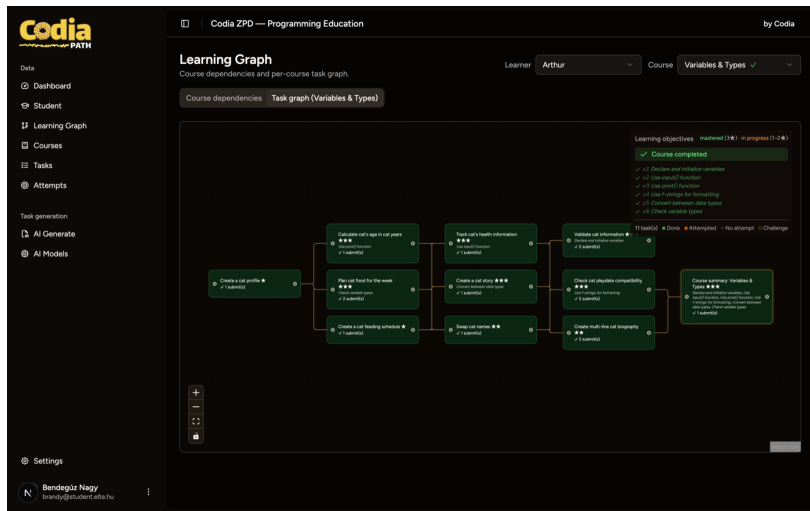
Each course additionally contains a *task dependency graph*: a second DAG in which nodes represent individual programming tasks and edges indicate which earlier tasks must be attempted before a given task becomes available to a learner. Figure 3 shows an example: nodes represent tasks, edges encode prerequisites, and colours indicate attempt outcome. A learner’s personal instance of this graph is annotated with the outcome of each prior attempt, enabling the visualisation of their individual learning trajectory within the course.

4.2. Learning Objectives

Each course is associated with a set of *learning objectives* (e.g., “Declare and initialise variables”, “Use basic data types”, “Read user input with `input()`”). The objectives and topics prescribed by the NAT could provide a useful foundation for defining these in a real deployment. They serve as a progress checklist for learners (Figure 4) and are included verbatim in the LLM prompt during task generation, ensuring that generated tasks target specific learning goals.

4.3. ELO-Inspired ZPD Estimator

After each recorded attempt, Codia Path recomputes the learner’s skill estimate from their attempt history using a bounded, rating-style rule inspired by ELO systems [2]: instead of a fixed reference scale, competence is estimated by comparing attempt outcomes against the difficulty of the attempted tasks.

Figure 3. Task dependency graph for *Variables & Types*.

Title	Description	Learning objectives (checklist)	Tasks
Variables & Types ✓	Basic variables, data types, user input, console output	- v1 — Declare and initialize variables - v2 — Use input() function - v3 — Use print() function - v4 — Use f-strings for formatting - v5 — Convert between data types - v6 — Check variable types	33 View tasks
Operators & Expressions ✓	Arithmetic, comparison, logical operators, expressions	- e1 — Arithmetic operators: +, -, *, /, % - e2 — Comparison operators: ==, !=, >, <, >=, <= - e3 — Logical operators: and, or, not - e4 — Compound expressions with parentheses - e5 — Ternary operator (conditional expression) - e6 — Increment and decrement operators (++ , --)	26 View tasks
Conditionals ✓	if/else, if-elif-else, nested conditionals	- c1 — if else - c2 — if-elif-else chain - c3 — Nested conditionals - c4 — Truthy and falsy values - c5 — Early return in functions - c6 — Compound conditions	24 View tasks
Loops - for	for loops, iteration, range()	- l1 — for loop basics with range() - l2 — Loop variable - l3 — Count/accumulate in loop - l4 — Countdown loops - l5 — Iterate with steps	11 View tasks
Loops - while	while loops, do-while patterns, condition-based iteration	- w1 — while loop - w2 — Condition-based iteration - w3 — Break/continue statements	9 View tasks

Figure 4. Learning objectives and mastery tracking.

Let a learner have n recorded attempts, where attempt i has difficulty $d_i \in \{1, 2, 3\}$ and outcome $o_i \in \{0, 1\}$ ($1 = \text{success}$). Each attempt contributes a difficulty-anchored target t_i and a weight w_i ,

$$t_i = \begin{cases} d_i & \text{if } o_i = 1, \\ d_i - \beta & \text{if } o_i = 0, \end{cases} \quad w_i = \begin{cases} w^+ & \text{if } o_i = 1, \\ w^- & \text{if } o_i = 0, \end{cases}$$

so that a success anchors the estimate to the task’s difficulty while a failure pulls it slightly below. The skill estimate is the weighted mean of these targets, clamped to the difficulty scale:

$$S = \text{clip}\left(\frac{\sum_{i=1}^n w_i t_i}{\sum_{i=1}^n w_i}, 1, 3\right),$$

with $w^+ = 1.2$, $w^- = 0.6$, and $\beta = 0.4$ in the current implementation. Successful attempts thus carry more weight than failures, and the estimate remains within the $[1, 3]$ difficulty range by construction, avoiding the unbounded drift and scale-calibration problems that a raw ELO K -factor update would introduce on such a short scale. The estimate depends only on the outcome o_i and the difficulty d_i ; per-attempt error count and completion time are stored in the learner profile as descriptive statistics (average error count, average completion time) but do *not* enter the skill estimate in the current prototype. The prototype maintains a single skill estimate per learner, aggregated over all attempted courses; per-course and per-objective estimates are natural extensions discussed in Section 7.

ZPD band. The ZPD band for a learner is the interval $[S - \delta, S + \delta]$ intersected with the difficulty scale $[1, 3]$, where $\delta = 0.5$ by default. This band is passed to the LLM as a *soft* constraint on the difficulty of the generated task; Section 4.4 describes how an explicit target difficulty can override it.

Worked example. A learner who has succeeded on two difficulty-2 tasks and failed one difficulty-3 task obtains

$$S = \frac{1.2 \cdot 2 + 1.2 \cdot 2 + 0.6 \cdot (3 - 0.4)}{1.2 + 1.2 + 0.6} = \frac{6.36}{3.0} \approx 2.12,$$

yielding a ZPD band of $[1.62, 2.62]$.

Parameter choices. Table 1 lists the estimator and generation parameters with their default values and rationale.

4.4. AI Task Generation

The system assembles a structured prompt containing:

1. **Learner trajectory:** The learner’s recent attempt history, each entry listing the task title, its difficulty level, and the success/failure outcome.

Table 1. Estimator and generation parameters (current implementation).

Parameter	Value	Rationale
Success weight w^+	1.2	Successes weigh more than failures.
Failure weight w^-	0.6	Failures still inform the estimate, but less.
Failure offset β	0.4	A failed task pulls the estimate slightly below its difficulty.
Band half-width δ	0.5	Width of the ZPD band around the skill estimate.
Difficulty scale	1–3	Manually assigned per-task difficulty labels.
Skill range (clip)	[1, 3]	Keeps the estimate on the difficulty scale.

2. **ZPD band:** The numerical difficulty interval $[zpdMin, zpdMax]$ computed as described in Section 4.3.
3. **Course learning objectives:** The full list of objectives for the selected course, annotated with the learner’s current mastery status.
4. **Interest keywords:** Thematic tags provided by (or assigned to) the learner, e.g., **space**, **cooking**, **sports**. The LLM incorporates these as context and narrative elements, personalising the task scenario without changing its computational content.
5. **Generation history:** Previous AI-generated tasks for this learner, with any recorded feedback, so that the model avoids repetition.

The LLM is instructed to return a structured JSON object containing: task title, description, starter code, sample solution, difficulty estimate (1–3), targeted learning objectives, and a brief *pedagogical rationale* explaining why this task is appropriate for the learner’s current ZPD.

Target difficulty and scaffolding. The ZPD band acts as a *soft* constraint. Alongside it, the generator accepts an explicit target-difficulty control (a 1–3 rating together with a challenge preference such as *easier*, *steady*, or *challenge*), and the LLM is instructed to match the requested target difficulty. When a learner first enters a new course, the system requests a difficulty-1 scaffolding task to introduce the topic gently, even if the learner’s skill-derived band lies higher; in this case the requested target difficulty deliberately overrides the lower bound of the band. This mechanism explains the difficulty-1 task generated for Arthur in Section 5 despite a band of [2.09, 3.00].

5. Evaluation with Simulated Learners

The classroom pilot in Section 3 evaluated the *Codia* platform, not Codia Path; the two systems are distinct, and the pilot motivates but does not validate the adaptive method proposed here. In the absence of real-world classroom data from Codia Path (the system is not yet integrated with the production Codia platform), we report an *illustrative simulation study* – not a controlled experiment – using three *simulated learner trajectories* constructed with the assistance of Claude Opus 4.

5.1. Simulated Trajectories

Three learner personas were designed to represent distinct learning profiles typical of secondary school programming classes:

Thomas Motivated beginner, no prior experience; progresses slowly with frequent errors. Seeded: dog-themed tasks, *Variables & Types* through *For Loops* (42 attempts).

Arthur Moderate prior experience (Scratch, Micro:bit); average pace, occasionally revisits earlier concepts. Seeded: cat-themed tasks across six courses up to *Functions – Basics* (54 attempts).

John Fast learner with strong Python background; few errors, consistently targets higher difficulty. Seeded: skateboard-themed tasks, *Variables & Types* and *Operators & Expressions* (16 attempts).

Success rates, error counts, and completion times were set consistently with each persona’s profile. Interest keywords are supplied at generation time; for the three demonstrated tasks we used **Dune** for Thomas, **M&M’s** for Arthur, and **watermelon** for John.

5.2. Generated Tasks and Results

For each simulated learner, we triggered the AI Generate module after the full trajectory had been recorded, targeting courses at or near the learner’s current frontier. Table 2 summarises the key parameters and the resulting generated tasks.

5.3. Detailed Task Descriptions

Below we reproduce the AI-generated task descriptions for each generation in Table 2, assessed along three dimensions: difficulty appropriateness, objective alignment, and narrative coherence.

Table 2. Generation inputs and AI-generated tasks for the three illustrated examples. All generated with `o4-mini` (OpenAI). Column **D** is the requested target difficulty; Arthur’s $D = 1$ is a deliberate new-course scaffolding override of the ZPD band (Section 4.4), not a constraint violation.

Learner	Course	Skill	ZPD band	D	Generated task
Thomas	For Loops	2.47	[1.97, 2.97]	2	<i>Spice Harvest Accumulator</i>
Arthur	Arrays & Iter.	2.59	[2.09, 3.00]	1	<i>Accessing M&M Colors in a List</i>
John	Oper. & Expr.	2.73	[2.23, 3.00]	3	<i>Watermelon Juice Estimator</i>

Thomas: *Spice Harvest Accumulator* (D2, keywords: “Dune (the movie)”).

House Atreides needs your help on Arrakis to keep track of the melange (spice) they collect. Write a program that: (1) asks the user for the number of days of harvesting (an integer); (2) for each day from 1 to that number, prompts the user to enter the pounds of spice harvested on that day and adds that amount to a running total; (3) after all days have been processed, prints the total in the format: `"Total spice harvested: X pounds"`.

The task targets “count and accumulate values in loops” at difficulty 2, within Thomas’s ZPD band [1.97, 2.97]. The Dune narrative reflects the supplied keyword, and the accumulation-loop structure suits a persistent beginner entering the loops course (Figure 5).

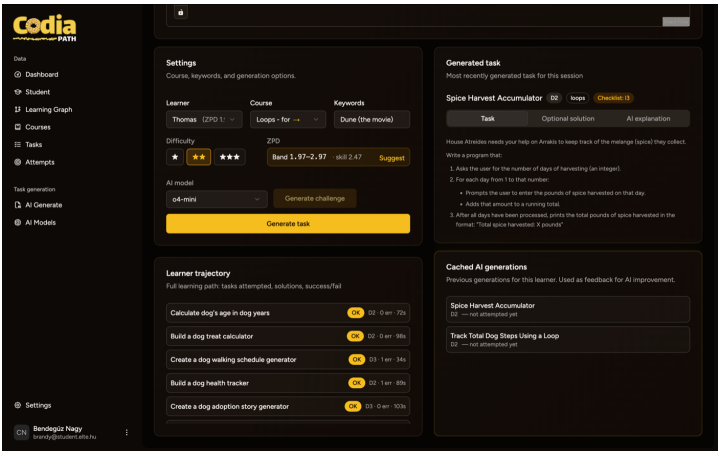


Figure 5. Thomas’s generated task, illustrating how a beginner’s ZPD band and interest keywords produce a contextually appropriate accumulation-loop exercise at difficulty 2.

Arthur: Accessing M&M Colors in a List (D1, keywords: “M&M’s”).

Create a list named `mm_colors` with these M&M colors in this exact order: red, blue, green, yellow, orange. Then: (1) print the first color; (2) print the third color; (3) print the fifth color. Use list indexing (with indices 0, 2, and 4) and f-strings to format your output.

Arthur had completed six courses and was entering *Arrays & Iteration*. Although his skill-derived ZPD band was [2.09,3.00], the generator was asked for a difficulty-1 task because this was his first attempt in a new course: entering a new topic triggers a scaffolding override that introduces the concept with a low-pressure task (Figure 6), regardless of the higher global skill band. The M&M theme personalises the exercise without altering its core requirement – list creation and index access.

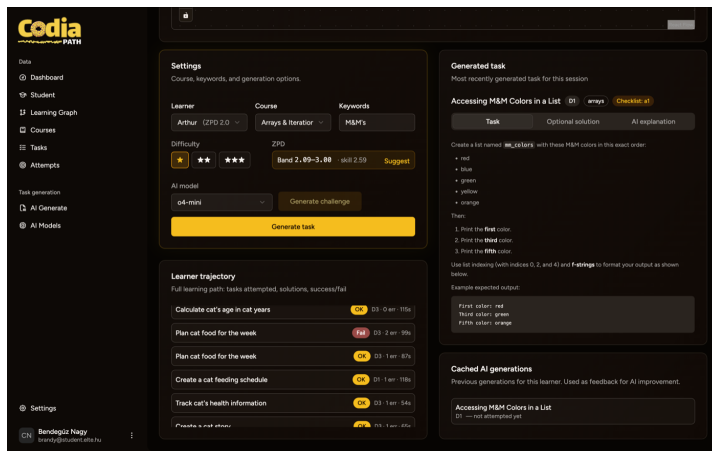


Figure 6. Arthur’s generated task, showing that the system selects difficulty 1 when a learner enters a new course, producing a gentle, interest-themed introduction to list indexing.

John: Watermelon Juice Estimator with Parentheses (D3, keywords: “watermelon”).

Write a program that helps a watermelon vendor estimate how much juice they can get: (1) prompt the user to enter the weights (kg) of three sample watermelons; (2) prompt for the total number of watermelons; (3) compute the **average weight** – use parentheses to group the sum before dividing by 3; (4) compute the **total juice** in litres (1 kg yields 0.9l) – use parentheses for correct operator precedence; (5) print both values formatted to two decimal places.

This targets “build compound expressions using parentheses” at difficulty 3, matching John’s skill of 2.73 (Figure 7). The multi-step arithmetic with explicit parenthesisation challenges an advanced learner.

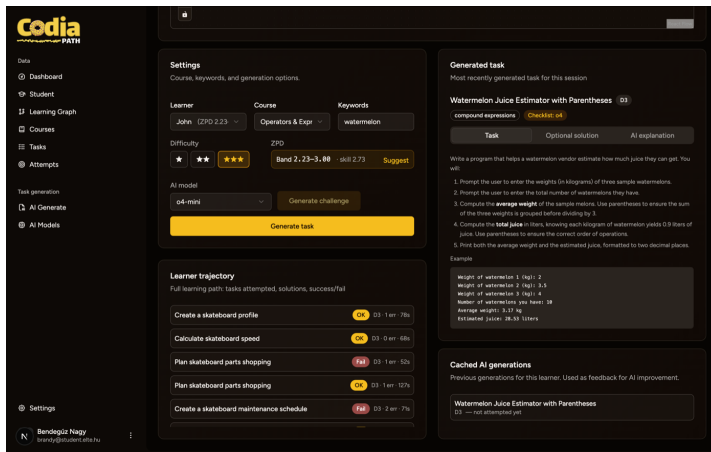


Figure 7. John’s generated task, demonstrating that a high skill score shifts the ZPD band upward, prompting the LLM to produce a difficulty 3 compound-expression task.

5.4. Discussion

Two of the three generated tasks (Thomas and John) fell strictly within the computed ZPD band; Arthur’s difficulty-1 task was a deliberate new-course scaffolding override rather than a constraint violation (Section 4.4). Each task incorporated the supplied interest keywords naturally. The three trajectories produced differentiated skill values (2.47, 2.59, 2.73), and the LLM responded with correspondingly differentiated structures – from a scaffolded list-indexing exercise (D1) for a learner entering a new course to a multi-step arithmetic task with explicit parenthesisation (D3) for an advanced learner. These illustrative results are consistent with the working hypothesis that even a lightweight ZPD estimator can guide differentiated task personalisation; they do not, however, constitute evidence of improved learning outcomes, which would require a controlled study with real learners.

6. Conclusions

We have presented Codia Path, a proof-of-concept system that combines a graph-based curriculum model, an ELO-inspired ZPD estimator, and LLM-based task generation into a lightweight pipeline for personalised programming education. A pilot study with 17 students of the *Codia* platform indicated that intrinsic challenge – rather than gamification rewards – drives motivation, which motivated the design of Codia Path; an illustrative evaluation of Codia Path with three simulated learner profiles produced pedagogically plausible tasks. We emphasise that these are proof-of-concept results: we do not yet claim improved learning outcomes relative to a baseline, which remains future work.

7. Limitations and Future Work

The evaluation has several important limitations:

- **Simulated, LLM-assisted data.** The system has been exercised only with three simulated learner trajectories, themselves constructed with LLM assistance, rather than with real classroom data; this carries a risk of circularity and cannot demonstrate learning gains.
- **No baseline comparison.** We do not compare ZPD-personalised sequences against teacher-assigned tasks or any other baseline.
- **No automated correctness verification.** Generated starter code and reference solutions were not checked by an automated test harness, and LLM-generated tasks occasionally required minor syntactic corrections.
- **Author-only task review.** Generated tasks were assessed by the authors along difficulty appropriateness, objective alignment, and narrative coherence; independent expert (teacher) review is planned but not yet performed.
- **Coarse, single-scalar skill model.** The estimator collapses multidimensional competence into a single skill estimate per learner on a discretised 1–3 difficulty scale, with manually assigned difficulty labels.
- **Unused interaction signals.** Error count and completion time are recorded but do not yet influence the skill estimate or task generation.

More broadly, LLM-based generation carries inherent risks – including factual inaccuracies (hallucinations), implicit biases in task framing, and potential over-reliance that may weaken learners’ independent problem-solving – all of which necessitate expert review of generated content before classroom deployment.

The immediate priority is integration into the production Codia platform and a controlled classroom experiment comparing ZPD-personalised sequences against teacher-assigned tasks [7]. Further directions include per-course and per-objective skill estimates, incorporating error count and completion time into the estimator, expanded course coverage, and collaborative ZPD estimation through peer comparison.

References

- [1] M. CSIKSZENTMIHALYI: *Flow: The Psychology of Optimal Experience*, New York: Harper & Row, 1990.
- [2] A. E. ELO: *The Rating of Chessplayers, Past and Present*, New York: Arco, 1978.
- [3] C. J. FONG, D. J. ZALESKI, J. K. LEACH: *The Challenge–Skill Balance and Antecedents of Flow: A Meta-Analytic Investigation*, *The Journal of Positive Psychology* 10.5 (2015), pp. 425–446, DOI: [10.1080/17439760.2014.967799](https://doi.org/10.1080/17439760.2014.967799).

- [4] J. HAMARI, J. KOIVISTO, H. SARSA: *Does Gamification Work? – A Literature Review of Empirical Studies on Gamification*, in: Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS), IEEE, 2014, pp. 3025–3034, DOI: [10.1109/HICSS.2014.377](https://doi.org/10.1109/HICSS.2014.377).
- [5] J. KOIVISTO, J. HAMARI: *The Rise of Motivational Information Systems: A Review of Gamification Research*, International Journal of Information Management 45 (2019), pp. 191–210, DOI: [10.1016/j.ijinfomgt.2018.10.013](https://doi.org/10.1016/j.ijinfomgt.2018.10.013).
- [6] B. NAGY, G. HORVÁTH: *A gamifikáció hatása a tanulói motivációra egy középiskolai csoport programozásóráin*, in: InfoDidact 2025 – Az informatika tanításának módszertana, Tanulmánykötet, Budapest, Hungary: Eötvös Loránd University, 2025, ISBN: 978-963-489-851-1, URL: https://infodidact.elte.hu/2025/infodidact2025_tanulmanykotet.pdf.
- [7] J. NAKAMURA, M. CSIKSZENTMIHALYI: *The Concept of Flow*, in: Handbook of Positive Psychology, ed. by C. R. SNYDER, S. J. LOPEZ, Oxford: Oxford University Press, 2002, pp. 89–105, DOI: [10.1093/oxfordhb/9780195187243.013.0018](https://doi.org/10.1093/oxfordhb/9780195187243.013.0018).
- [8] A. PALMQUIST: “Gamification was not the problem”: A Case Study Exploring Factors Affecting Teachers’ Approval of Gamification, in: Proceedings of the 24th International Academic Mindtrek Conference (Mindtrek ’21), ACM, 2021, pp. 106–116, DOI: [10.1145/3464327.3464347](https://doi.org/10.1145/3464327.3464347).
- [9] S. SARSA, P. DENNY, A. HELLAS, J. LEINONEN: *Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models*, in: Proceedings of the 2022 ACM Conference on International Computing Education Research (ICER ’22), ACM, 2022, pp. 27–43, DOI: [10.1145/3501385.3543957](https://doi.org/10.1145/3501385.3543957).
- [10] K. VANLEHN: *The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems*, Educational Psychologist 46.4 (2011), pp. 197–221, DOI: [10.1080/00461520.2011.611369](https://doi.org/10.1080/00461520.2011.611369).
- [11] L. S. VYGOTSKY: *Mind in Society: The Development of Higher Psychological Processes*, Cambridge, MA: Harvard University Press, 1978, DOI: [10.2307/j.ctvjf9vz4](https://doi.org/10.2307/j.ctvjf9vz4).
- [12] L. YANG, X. SUN, H. LI, R. XU, X. WEI: *Difficulty Aware Programming Knowledge Tracing via Large Language Models*, Scientific Reports 15 (2025), p. 11475, DOI: [10.1038/s41598-025-96540-3](https://doi.org/10.1038/s41598-025-96540-3).



Acoustic Traffic Monitoring with MEMS Microphones, Concatenated log-Bark Spectrograms and CNN

István Pintér, Lóránt Kovács

John von Neumann University,
GAMF Faculty of Engineering and Computer Science
pinter.istvan@nje.hu, kovacs.lorant@nje.hu

Abstract. The paper presents our results in algorithm-development and solutions for two ATM problems using MEMS microphone-signals. The aim was that the algorithms have to be realised in the sensor itself, thus we followed the edge-AI system-concept. The IDMT-Traffic dataset was used in algorithm development. The paper proposes the concatenated log-Bark spectrogram as feature extraction, followed by the custom CNN classifier. In the motion's direction determination (from left/from right/no vehicle) the overall accuracy was $97.70 \pm 0.64\%$. The detection of passing vehicle in background noise was considered as a two-class classification problem (vehicle/no vehicle). The accuracy, F1-score and Matthew's Correlation Coefficient (MCC) performance parameters achieved were 99.75%, 0.9975, 0.9949, respectively. The conclusion, based on experimental evidence, was that the MEMS-microphones could be used to develop edge-AI type sensors for motion's direction determination and detection of passing vehicle in background noise using the concatenated log-Bark spectrograms converted to grayscale images as input to a CNN-classifier.

Keywords: MEMS-microphone, log-Bark-spectrogram, CNN, motion-direction, passing vehicle detection, accuracy, F1-score, MCC

2020 Mathematics Subject Classification: Primary 68T07; Secondary 94A12

1. Introduction

Recent advances in info-communication technologies highlight the concept of smart cities. One objective is the optimal control of urban and suburban vehicle traffic,

which requires traffic monitoring. It's main problems are the following: 1) detection and counting of passing vehicles, 2) determination of vehicle's category, 3) determination of motion's direction, and 4) vehicle's speed estimation. The real-time data necessary for traffic monitoring are supplied by appropriate sensors. Inductive loops and pneumatic pipes are highly reliable ones for solving problem 1) and 3). Unfortunately, these are not suitable for real-time traffic monitoring: inductive loops must be built into the road during road-building, and pneumatic pipes must be laid across the road, thus affecting the traffic itself. Another family of sensors are based on electromagnetic operating principles (e.g. radar, lidar, and cameras). However, their – otherwise excellent – detection accuracy is highly dependent on meteorological conditions and time of day [8]. Besides, the high cost of installation and maintenance also calls for another or additional sensor-domain, namely acoustic sensors. This leads to the concept of Acoustic Traffic Monitoring (ATM). Most frequently, pressure-sensing microphones are used, and a promising goal is developing sensors with low-cost, medium-quality omnidirectional MEMS (Micro-ElectroMechanical Systems) microphones. The main efforts in this sensor-development field can be summarized as: (I) minimizing the cost of sensors, (II) decreasing the installation and maintenance costs, (III) decreasing the electrical power consumption of the sensor and signal processing units, (IV) optimizing the suitable sensor network architecture for covering a large geographic area with many cooperating sensor units, and (V) guaranteeing the continuous operation even in adverse weather conditions.

In this paper we present our results in algorithm-development for ATM problems 1) and 3) using MEMS microphone-signals. The problem 2) was also investigated. Our aim was that the algorithms have to be realised in the sensor itself, thus we followed the edge-AI system-concept. Several freely available audio datasets exist for this purpose [1, 7, 9]. We used the IDMT-Traffic dataset, because it consists of two-channel audio recordings both with high-quality sE8 and MEMS microphones. Therefore, the performance of MEMS-based algorithms can be compared with sE8-based ones. We considered [1] as reference, because the authors presented possible solutions for problems 1), 2) and 3) in case of sE8 microphones. In solving the problem 3) their neural network was a variant of MobileNet, adapted to the problem at hand. The input was the cross-correlogram, computed as a series of cross-correlations between the two microphone-channels. For problem 2) their classifier was a pretrained CNN (Convolutional Neural Network), which was adapted to the concrete problem using transfer learning. The input was the log-mel spectrogram computed from the average of the two channels' signals.

The structure of the paper is the following. Section 2 summarizes the main properties of the dataset used for algorithm development. Section 3 introduces the concept of concatenated log-Bark features and describes the CNN structure. In Section 4 the numerical results achieved in vehicle category classification can be found, which is followed by in Section 5 and 6 with results of direction of movement classification and performance parameters of passing vehicle detection. The paper ends with conclusions.

2. The IDMT-Traffic dataset

In the work we used the IDMT-Traffic dataset [1], which consists of 17506 two-channel (stereo) wav-files, recorded both with high quality sE8 and medium quality, low cost MEMS microphones. The duration of a particular recording is $t_d = 2$ s, the sampling frequency is $f_s = 48$ kHz. The dataset contains two main sound categories: background noise (8144 recordings) and sound of passing vehicles (9362). In the latter case there are four sub-categories: car (7804), bus (106), truck (1022) and motorcycle (430). The sounds were recorded in four places (three urban and one country-road) with different weather conditions (dry and wet road) and with different speed limits (30 km/h, 50 km/h and 70 km/h). The direction of motion was also recorded. The recordings are annotated, the audio file-name contains the labels: date, location, speed limit, daytime, weather condition, vehicle category/background noise, motion's direction and microphone type.

3. Concatenated log-Bark features and the CNN

In the recent, relevant literature the input of the classifier is the so-called log-mel spectrogram [1, 7, 9]. The log-mel spectrogram is computed from linear spectrogram using the mel-frequency scale and triangle-shaped, overlapped filters [5]. The mel frequency-scale is a characteristics of human hearing, and its non-linear function is the result of function-fitting to experimental, psychoacoustical data [10]. Another psychoacoustical frequency scale is the Bark-scale, with 24 critical bands of human hearing [12]. Therefore, in an earlier preliminary study – with the subject of detecting passing vehicles in background noise – we compared the effect of the mel/Bark scale on detection accuracy. Beyond that, in that comparison we used both triangular-shaped filters and raised cosine filters, too. The final result of the study was, that in case of sE8 microphone the best combination is mel-scale with raised-cosine filters, and in case of MEMS microphone the Bark-scale with raised cosine filters gave the best accuracy. Therefore, in this work we used the log-Bark spectrogram computed with raised cosine filters. Moreover, in order to preserve the time-delay information between the two microphone channels, we concatenated the log-Bark spectrograms of the two channels ensuring the proper time-alignment. Thus, the concatenated log-Bark spectrogram was the result of feature extraction.

3.1. The concatenated log-Bark spectrogram

The signal was processed using sliding windows with fixed window length ($t_w = 100$ ms) and hop ($t_h = 25$ ms). Thus the number of samples in a window was $N = f_s \cdot t_w = 4800$, and the number consecutive of windows was $M = t_d/t_h = 80$ (not 77, because the last three windows were zero-padded at the end). In the m th window the signal's one-sided power spectrum $|S_m(l)|^2$, $m = 1..M, l = 1..N/2$ was estimated using Hamming-window and discrete Fourier-transform (The frequency

resolution was $\Delta f = 1/t_w = 10$ Hz). The linear spectrogram was a series of the one-sided power spectra.

We processed the linear spectrogram further using a psychoacoustical model of human hearing. From the available non-linear functions (scales) we used the Bark-scale, mentioned above, with the invertible function below [11] :

$$\eta(f) = 6 \cdot \operatorname{asinh}\left(\frac{f[H\text{z}]}{600[H\text{z}]}\right)$$

We covered the $[0; \eta(f_{\max})]$ interval with K overlapped, raised-cosine-type weighting functions (sub-band filters). In these bands we summed up the weighted power values and converted the result into dB. Thus finally we got the log-Bark spectrogram as signal representation of the channel's signal.

The sub-band filters were shifted versions of the raised-cosine prototype function with finite support of $2\Delta\eta$, where $\Delta\eta = \eta(F_s/2)/(K+1)$. The k th sub-band filter therefore:

$$\Psi_k(\eta) = \begin{cases} \frac{1 + \cos\left[\frac{\pi \cdot (\eta - k\Delta\eta)}{\Delta\eta}\right]}{2} & \eta_1 \leq \eta \leq \eta_2 \\ 0 & \text{else} \end{cases}$$

where $\eta_1 = (k-1)\Delta\eta$ and $\eta_2 = (k+1)\Delta\eta$. With the inverse η^{-1} of $\eta(f)$ the corresponding frequency values were $f_1 = \eta^{-1}(\eta_1)$ and $f_2 = \eta^{-1}(\eta_2)$ and the frequency band was $f_1 \leq f \leq f_2$. The k th sub-band filter in the frequency domain was $H_k(f) = \Psi_k(\eta^{-1})$, $f \in [f_1; f_2]$. With the frequency resolution above finally we got $H_k(l) = H_k(l \cdot \Delta f)$, $l \in [l_1; l_2]$ where $l_1 = \lfloor f_1/\Delta f \rfloor$, $l_2 = \lceil f_2/\Delta f \rceil$, the sub-band filter in the linear frequency domain for discrete-time processing.

For better visibility Figure 1 shows a raised-cosine filterbank with $K = 3$ filters transformed back to linear frequency scale ($f_s = 8$ kHz, $f_{\max} = f_s/2$)

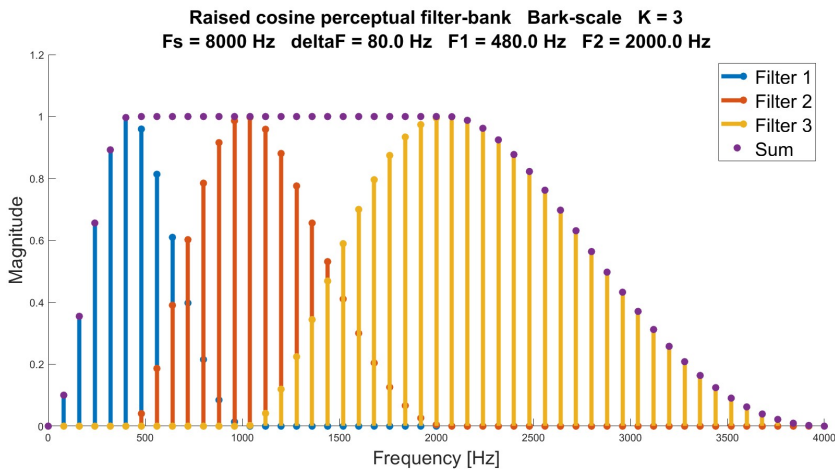


Figure 1. Raised-cosine filterbank example.

The final step in feature extraction was computing the weighted sums from power spectrum in the m th signal window:

$$P_{k,m} = 10 \lg \left(\sum_{l=l_1}^{l_2} |H_k(l)|^2 \cdot |S_m(l)|^2 \right), \quad k = 1..K, \quad m = 1..M$$

The $P_{k,m}$ is the log-Bark spectrogram, which is an $K \times M$ matrix of real values.

In Figure 2 an example can be seen using a recording from the IDMT-Traffic dataset. The upper trace is trace 1, which shows the Hamming-windowed 100 ms long signal segment (of 4800 samples). Trace 2 is the one-sided power spectrum (of 2400 components). Trace 3 shows the 24 raised-cosine-type filters, designed in the Bark-scale and warped back to linear frequency scale. Finally in trace 4 the results of weighted sum of power values of each filters can be seen expressed in dB. These computations are resulted in a 24 dimensional vector, which is a column of the log-Bark spectrogram matrix. It is important to note, that changing over to log-Bark spectrogram from linear spectrogram results in a 100-times data reduction.

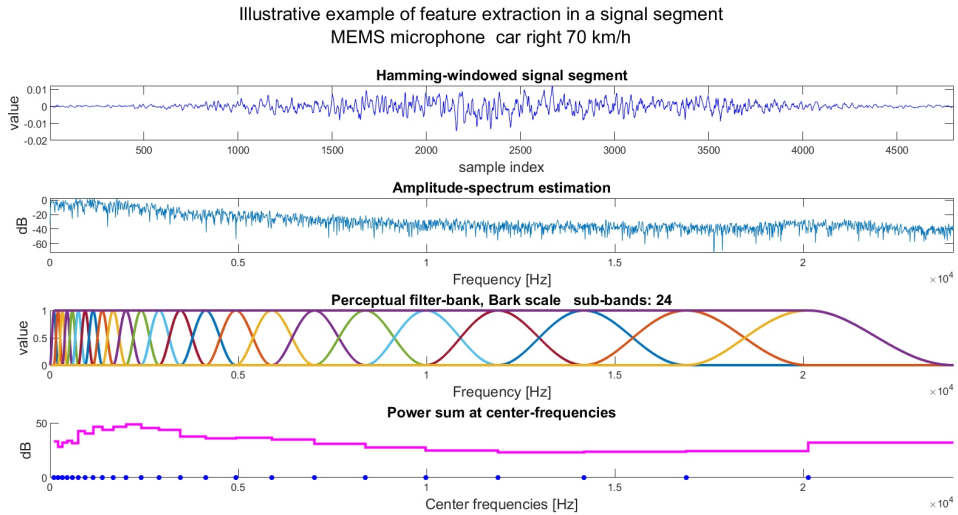


Figure 2. Log-Bark power spectrum of a signal segment.

Figure 3 illustrates that the log-Bark spectrogram representation enhances the weighted power sums in the lower frequency bands. In the figure the log-Bark spectrogram (left) and the linear spectrogram (right) can be seen in case of 50 km/h bus recording with MEMS microphones. The concatenation operation means that the two channels' log-Bark spectrograms are stacked above each other. It ensures the proper time-alignment of the channels' samples. The final result of the feature extraction is thus a 48×80 matrix of real values.

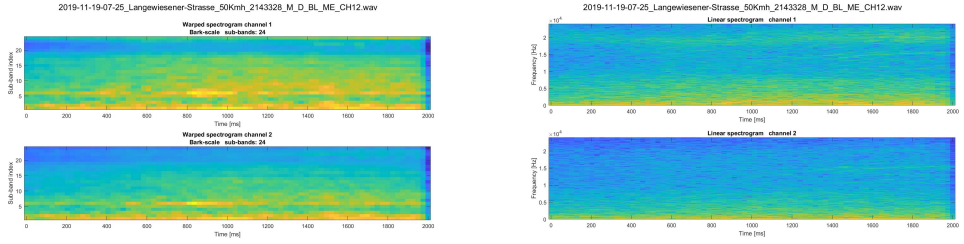


Figure 3. Log-Bark (left) and linear spectrograms (right).

3.2. The CNN structure

The concatenated log-Bark spectrogram representation, described above, can easily be converted to a 48×80 gray-scale image (PNG file-format was used). The interval of the lowest and greatest matrix-elements was scaled to $[0;255]$ of real values linearly, then uniformly quantized to $[0;255]$ interval of nonnegative integers. So it was obvious to select CNN [2] as the classifier part of the system, because it is well-proven and widely-used in ATM problems [1, 7, 9]. However, we intentionally did not use a pre-trained version. It makes it possible to optimize further the network structure if necessary when implementing the CNN classifier in the sensor. In designing the CNN structure we used the MATLAB Deep Learning Toolbox. Interestingly, the CNN processing pipeline was the same in both vehicle category classification and direction of movement determination. The network structure was the following (ReLU denotes the $f(x) = \max(0, x)$ function). There were three main building block types. We denote here with type-1 a sequence of <2D convolution layer, batch normalization, ReLU activations, max-pooling> operations, with type-2 the <2D convolution layer, batch normalization, ReLU activations> sequence and type-3 the <fully connected layer, softmax activations, classification layer>. Using these notations the image input layer was followed by two type-1 layers. The outputs the second were further processed with a type-2 block. It was followed by a type-3 block for outputting the predicted class labels. The size of convolution layers' filters were 3×3 in each blocks, the number of filters were 8, 16, 32, respectively. Not only the weights, but biases were trained as well. The total number of learnable parameters were 26011. The network was trained using the cross-entropy loss-function and Adam optimizer, with minibatch-size of 128 and initial learning rate of 0.001.

4. Vehicle category classification

From the IDMT dataset we created the “commercial vehicle” class from recordings of trucks and buses in order to distinguishing them from passenger cars. When creating the training/validation and test subsets of recordings we followed the policy of [1]: recordings labelled with 30 km/h and 50 km/h were the training/validation

sets and recordings with 70 km/h speed limit was the test set. As a consequence, we excluded all motorcycle recordings, because there are 5 recordings with 30 km/h labels, there is no with 50 km/h and 70 km/h and there are 123 with “Unknown” speed labels. The vehicle category classification was a 3-class classification problem: passenger car (car), commercial vehicle (bus or truck), no vehicle (labelled as BG, background noise in the dataset). We used the `splitEachLabel` function of MATLAB Deep Learning Toolbox with parameters of 0.9 and ‘randomize’ with default seed. The number of recordings of each set can be found in Table 1 (with 90% training, 10% validation ratios).

Table 1. Number of recordings in datasets.

vehicle category	training	validation	test
car	2471	274	1157
bus or truck	332	37	195
no vehicle	2395	266	1412

The structure of CNN classifier was described in the Subection 3.2. We did 10 training experiments, evaluated the confusion matrix, selected the best network with greatest overall accuracy and computed the average and standard deviation of experiments’ accuracies. Figure 4 shows randomly selected training grayscale image examples.

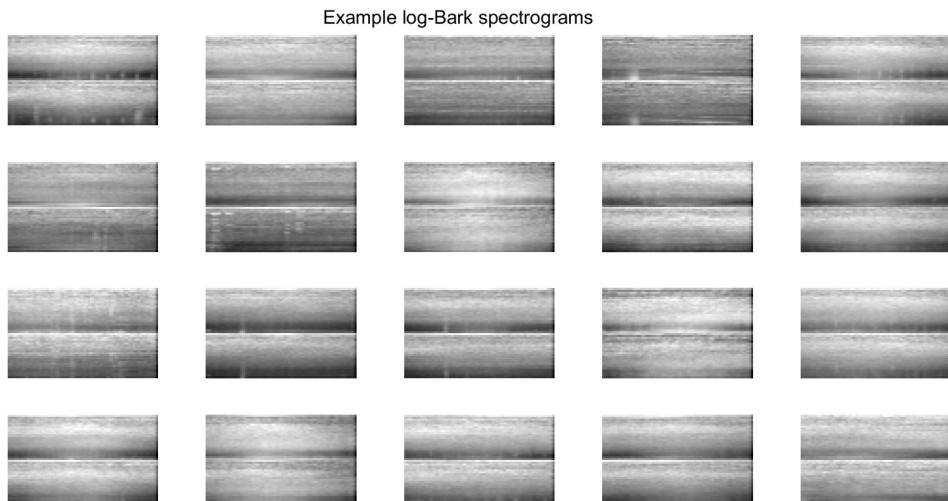


Figure 4. Randomly selected training images for vehicle category classification.

The best confusion matrix in case of 10 experiments and 10 epoch/experiments can be seen in Table 2. The best overall accuracy was 94.79%, the average accuracy was $94.04 \pm 0.48\%$.

Table 2. Vehicle category classification: best confusion matrix.

True label	bus or truck	87 (44.62%)	107 (54.87%)	1 (0.51%)
	car	27 (2.33%)	1128 (97.49%)	2 (0.17%)
	no vehicle	4 (0.28%)	3 (0.21%)	1405 (99.5%)
		bus or truck	car	no vehicle
		Predicted label		

In the reference paper [1] high quality sE8 microphone recordings were used for training a 4-class vehicle category classifier (car/truck/motorcycle/no vehicle). The corresponding classification ratios are the following: a) car/car decision 97.29% (sE8); 97.49% (MEMS), b) no vehicle/no vehicle decision 99.65% (sE8); 99.5% (MEMS). Let’s mention, that the truck/truck ratio was 38.84% (sE8) and the bus or truck/bus or truck ratio was 44.62% (MEMS). The greatest misclassification ratio in case of sE8 was the truck/car with 60.21%, and in case of MEMS the bus or truck/car ratio was 54.87%. To our opinion the primary cause of these high misclassification rates was the underrepresentation of training recordings of trucks (322, sE8) and bus or truck (369, MEMS) compared with cars’ (2745). We concluded, that both in case of sE8 and MEMS microphones further data collection is necessary to get a more balanced datasets.

5. Direction of movement classification

In organizing the computations in this case we followed similar methodology as in the vehicle category classification: the motorcycle recordings were excluded, the recordings labelled with 30 km/h or 50 km/h speed limits formed the trainig/validation substes with 90%/10% splits, and the recordings labelled with 70 km/h were the test set. We used the `splitEachLabel` function of MATLAB Deep Learning Toolbox with parameters of 0.9 and 'randomize' with default seed in this case also. Thus, the direction of movement classification was also a 3-class classification problem: from left to right/from right to left/no vehicle. The number of recordings of each set can be found in Table 3.

Table 3. Number of recordings in datasets.

direction of movement	training	validation	test
from left to right	1391	155	635
from right to left	1411	157	717
no vehicle	2395	266	1412

We also did 10 training experiments, selected the best network with greatest overall accuracy and characterized the training process itself with the average and

standard deviation of experiments' accuracies. The best confusion matrix in case of 10 experiments and 10 epoch/experiments can be seen in Table 4. The best overall accuracy was 98.63%, the average accuracy was $97.70 \pm 0.64\%$.

Table 4. Direction of movement: best confusion matrix.

True label	from left to right	616 (97.01%)	16 (2.52%)	3 (0.47%)
	from right to left	15 (2.09%)	701 (97.77%)	1 (0.14%)
	no vehicle	3 (0.21%)	0 (0%)	1409 (99.79%)
		from left to right	from right to left	no vehicle
		Predicted label		

In the reference paper [1] the datasets used for training were slightly different from ours. However, some comparison can be made based on confusion matrices. The corresponding classification ratios are the following: a) from left to right/from left to right 96.61% (sE8); 97.01% (MEMS), b) from right to left/from right to left 98.64% (sE8); 97.77% (MEMS) and c) no vehicle/no vehicle 99.79% (sE8) 97.79% (MEMS). We concluded, that MEMS microphone recordings can be used for direction of movement classification with similar performance of sE8.

6. Detection of passing vehicles

The detection of passing vehicles in a background noise is a 2-class classification problem (vehicle/no vehicle). The confusion matrix CM_2 can be derived from the 3-class one (CM_3) as:

$$CM_2(1,1) = CM_3(1,1) + CM_3(1,2) + CM_3(2,1) + CM_3(2,2),$$

$$CM_2(1,2) = CM_3(1,3) + CM_3(2,3),$$

$$CM_2(2,1) = CM_3(3,1) + CM_3(3,2),$$

$$CM_2(2,2) = CM_3(3,3).$$

Table 5 shows the confusion matrix with conventional notations.

Table 5. Two-class confusion matrix (CM_2) structure.

True label	vehicle	TP	FN
	no vehicle	FP	TN
		vehicle	no vehicle
		Predicted label	

For numerical assessment of detection we used the following performance parameters (for MCC se [3]):

$$\text{accuracy} = (TP + TN) / (TP + FN + FP + TN)$$

$$F1_{\text{score}} = 2TP / (2TP + FP + FN)$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

The numerical values derived from vehicle category classifier and direction of movement estimator can be seen in Table 6.

Table 6. Performance parameters of vehicle detection.

task	accuracy	F1-score	MCC
vehicle category	99.64%	0.9965	0.9928
direction of movement	99.75%	0.9975	0.9949

The accuracy of motion's direction classifier (99.75%) was somewhat greater, than that of the vehicle category classifier.

In the reference paper [1] the high quality sE8 microphone recordings were used, and there is no explicit accuracy data available for vehicle/no vehicle detection. However, we can compare the no vehicle/no vehicle decision ratios of corresponding classification tasks. Their TABLE III is the confusion matrix of vehicle category classifier; the ratio of no vehicle/no vehicle decision was 99.65%. TABLE V is the confusion matrix of vehicle category classifier with 99.79%. Our best ratio of 99.79% (see Table 4) was the same as the best reference data (99.79%). Thus the vehicle/no vehicle classification can be solved using our MEMS-based motion's direction classifier with accuracy of 99.75% (see Table 6).

7. Conclusions

Based on the numerical experimental evidence presented in the paper we concluded, that the MEMS-microphones can be used for vehicle/non vehicle classification and direction of movement estimation with acceptable accuracy. The great data reduction with log-Bark spectrogram signal representation and the small number of learnable parameters of the proposed CNN makes it possible to develop an edge-AI type sensor, too. In this respect we will consider the results published in [4, 6].

Another interesting work planned is the comparison the log-Bark representation-based solution with others (e.g. log-mel, MFCC, Gammatone Frequency Cepstral Coefficients, cross-correlogram and Generalized Cross-Correlation with Phase Transform) using larger datasets of [7, 9] with ablation studies.

However, in the vehicle category determination problem further data collection is needed in case of commercial vehicles in order to get a more balanced training dataset. The poor recognition accuracy caused by the class imbalance is planned to increase using class weighting and data augmentation.

Acknowledgements

The authors would like to thank the anonymous reviewers. Their comments and suggestions significantly improved the quality of the paper.

References

- [1] J. ABESSER, S. GOURISHETTI, ET AL.: *IDMT-Traffic: An Open Benchmark Dataset for Acoustic Traffic Monitoring Research*, in: Proceedings of 29th EUSIPCO European Signal Processing Conference, Dublin, Ireland, 2021, pp. 551–555, DOI: [10.23919/EUSIPCO54536.2021.9616080](https://doi.org/10.23919/EUSIPCO54536.2021.9616080).
- [2] C. M. BISHOP, H. BISHOP: *Deep Learning, Foundations and Concepts*, Springer Cham, 2024, DOI: [10.1007/978-3-031-45468-4](https://doi.org/10.1007/978-3-031-45468-4).
- [3] D. CHICCO, G. JURMAN: *The Matthews correlation coefficient (MCC) should replace the ROC AUC as the standard metric for assessing binary classification*, BioData Mining 16.4 (2023), pp. 1–23, DOI: [10.1186/s13040-023-00322-4](https://doi.org/10.1186/s13040-023-00322-4).
- [4] R. CORDOVA-CARDENAS ET AL.: *Edge AI in Practice: A Survey and Deployment Framework For Neural Networks on Embedded Systems*, Electronics 14.4877 (2025), pp. 1–39, DOI: [10.3390/electronics14244877](https://doi.org/10.3390/electronics14244877).
- [5] S. DAVIS, P. MERMELSTEIN: *Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences*, IEEE Transactions on Acoustics, Speech, and Signal Processing 28.4 (1980), pp. 357–366, DOI: [10.1109/TASSP.1980.1163420](https://doi.org/10.1109/TASSP.1980.1163420).
- [6] V. FRIDOUS ET AL.: *Optimization and Bechmarcking of Lightweight Neural Networks for Efficient Embedded AI Deployment*, Engineering Reports 8.e70818 (2026), pp. 1–14, DOI: [10.1002/eng2.70818](https://doi.org/10.1002/eng2.70818).
- [7] S. GHAFARZADEGAN ET AL.: *Sound of Traffic: A Dataset for Acoustic Traffic Identification and Counting*, in: Proceedings of Interspeech 2024, Kos, Greece, 2024, pp. 117–121, DOI: [10.21437/Interspeech.2024-1205](https://doi.org/10.21437/Interspeech.2024-1205).
- [8] T. KOVÁCS, L. KOVÁCS: *Economical Camera-based Measuring Scheme for Roundabout Traffic*, Periodica Polytechnica Transportation Engineering 53.2 (2025), pp. 235–241, DOI: [10.3311/PPtr.37617](https://doi.org/10.3311/PPtr.37617).
- [9] H. PARINEH, M. SARVI, A. B.: *MELAUDIS: A Large-Scale Benchmark Acoustic Dataset For Intelligent Transportation Systems Research*, Sci Data 12.362 (2025), pp. 1–13, DOI: [10.1038/s41597-025-04689-3](https://doi.org/10.1038/s41597-025-04689-3).
- [10] S. S. STEVENS, J. VOLKMANN, E. B. NEWMAN: *A scale for the measurement of the psychological magnitude pitch*, American Journal of Psychology 8.3 (1937), pp. 185–190, DOI: [10.1121/1.1915893](https://doi.org/10.1121/1.1915893).
- [11] S. WANG, A. SEKEY, A. GERSHO: *An objective measure for predicting subjective quality of speech coders*, IEEE Journal on Selected Areas in Communications 10.5 (1992), pp. 819–829, DOI: [10.1109/49.138987](https://doi.org/10.1109/49.138987).
- [12] E. ZWICKER: *Subdivision of the audible frequency range into critical bands (frequenzgruppen)*, Journal of the Acoustical Society of America 33.2 (1961), p. 248, DOI: [10.1121/1.1908630](https://doi.org/10.1121/1.1908630).



Formal Verification of Questionnaire Logic Using SMT Solvers

Péter SÁghelyi, Péter Bereczky

Eötvös Loránd University, Faculty of Informatics, Budapest, Hungary
psaghelyi@hotmail.com, berpeti@inf.elte.hu

Abstract. Modern computer aided surveys (CAPI/CATI) can contain hundreds of questions with complex conditional logic. In order to guarantee correctness through all possible traversal paths is a computationally demanding challenge. We propose to replace the traditional imperative, command-based questionnaire design with a constraint-based paradigm where correctness is verified statically rather than through path discovery. This work presents a formal, declarative framework for questionnaire specification and SMT-based verification that enables this separation. We formalize questionnaires as tuples of items with preconditions and postconditions inspired by Hoare Logic, classify item reachability and postcondition feasibility through satisfiability checks, and establish a four-level validation hierarchy with five theorems characterizing the relationships between per-item, global, and path-based analysis. The method is evaluated based on real-world questionnaires.

Keywords: questionnaire verification, SMT solvers, formal methods, Z3, preconditions, postconditions

2020 Mathematics Subject Classification: 68V15, 68Q60, 68N30

1. Introduction

The formal study of questionnaire structure spans six decades, from Picard’s foundational *Théorie des questionnaires* [18] through Fellegi and Holt’s systematic approach to edit rules [9], to Willenborg’s proof that checking questionnaire consistency is NP-complete [26]. Graph-based methods [1, 6, 7, 24] have provided visualization and documentation tools, while path enumeration approaches [8] have enabled systematic testing. However, none of these approaches provide formal guarantees about questionnaire correctness—a fundamental limitation of testing

that Dijkstra famously observed applies to all software systems [5].

The proliferation of AI-powered survey generation [14, 17] has amplified this need. AI systems can generate questionnaires with hundreds of interconnected questions whose conditional logic exceeds what human reviewers or even the generating AI can verify.

Our approach. We adopt a declarative, constraint-based paradigm inspired by Hoare Logic [10]. Just as Hoare triplets $\{P\}C\{Q\}$ specify program correctness through preconditions and postconditions, our framework specifies questionnaire correctness through logical constraints. Preconditions become Boolean formulas determining item visibility; postconditions become constraints on valid responses. This declarative specification maps naturally to SMT constraint satisfaction, enabling the Z3 theorem prover [15] to analyze questionnaire properties with mathematical precision.

Contributions. This paper makes four contributions:

1. A formal framework and domain-specific language (called QML [21]) for representing questionnaire logic using preconditions and postconditions.
2. A four-level SMT-based validation hierarchy (per-item, global, cycle detection, path-based) characterized by five theorems with formal proofs.
3. A compiler transforming the questionnaire logic expressed in QML with embedded Python code blocks into Z3 constraints.
4. The evaluation of our framework with real-world questionnaires.

The rest of this paper is structured as follows. Section 2 introduces the formal model for questionnaires followed by Section 3 discussing our approach for validating questionnaires based on the model. Section 4 introduces a concrete implementation, which connects the model to the Z3 SMT solver [15], and evaluates it with real-world questionnaires. Section 5 summarises related work, and Section 6 concludes and points out future directions.

2. Formal Questionnaire Model

We formalize questionnaires by defining their components axiomatically and then composing them into a complete structure.

Definition 2.1 (Item). An **item** I_i is a questionnaire element that presents content and optionally collects responses. Items may be informational (no outcome), scalar questions (single integer outcome), question groups (vector outcome), or matrix questions (matrix outcome).

Definition 2.2 (Outcome Variable). For each item I_i collecting responses, the **outcome variable** S_i represents the respondent's answer.

In this paper, we only discuss scalar outcome variables ($S_i \in \mathbb{Z}$) for clarity; however, real-world questionnaires require more complex answer structures. Our framework is capable of handling *vector* outcome variables $S_i \in \mathbb{Z}^k$ for question groups, or $S_i \in \mathbb{Z}^{m \times n}$ for matrix questions too.

We denote the complete vector of outcome variables as $\mathbf{S} = (S_1, \dots, S_n)$. An *assignment* is a vector $\mathbf{a} = (a_1, \dots, a_n)$ over the corresponding domains.

Definition 2.3 (Domain Constraint). Each outcome variable S_i has a **domain constraint** $D_i(S_i)$ restricting valid values, specified either as a range $\ell_i \leq S_i \leq u_i$ or an enumeration $S_i \in \{v_1, \dots, v_k\}$.

The conjunction of all domain constraints forms the **base constraint**:

$$B := \bigwedge_{i=1}^n D_i(S_i)$$

Let the set of domain-respecting assignments be $\text{Dom} := \{\mathbf{a} \mid \mathbf{a} \models B\}$. Here, we use $\models$ to denote the SMT satisfaction relation¹ used by Z3 [15]: $\mathbf{a} \models \phi$ means assignment $\mathbf{a}$ satisfies formula ϕ .

Definition 2.4 (Precondition). Each item I_i has a **precondition** P_i , a Boolean formula over $\mathbf{S}$ determining whether the item is presented. If $\mathbf{a} \models P_i$ for assignment $\mathbf{a}$, item I_i is displayed; otherwise it is hidden.

Preconditions encode the conditional logic that creates branching paths through the questionnaire based on previous answers.

Definition 2.5 (Postcondition). Each item I_i has a **postcondition** Q_i , a Boolean formula over $\mathbf{S}$ constraining valid responses. Postconditions are only evaluated when the item is accessible, requiring: $P_i \Rightarrow Q_i$.

Postconditions ensure logical consistency between answers and can dynamically restrict acceptable values based on prior responses.

Definition 2.6 (Questionnaire). A **questionnaire** $\mathcal{G}$ is a tuple $(\mathcal{I}, \mathbf{S}, \mathcal{D}, \mathcal{P}, \mathcal{Q}, I_{\text{start}})$ where:

- $\mathcal{I} = \{I_1, \dots, I_n\}$ is a finite set of items
- $\mathbf{S} = (S_1, \dots, S_n)$ is a vector of outcome variables
- $\mathcal{D} = (D_1, \dots, D_n)$ specifies domain constraints
- $\mathcal{P} = (P_1, \dots, P_n)$ specifies preconditions
- $\mathcal{Q} = (Q_1, \dots, Q_n)$ specifies postconditions
- $I_{\text{start}} \in \mathcal{I}$ is the designated starting item

Throughout, we write $\text{SAT}(\phi)$ when formula ϕ has a satisfying assignment, $\text{UNSAT}(\phi)$ when none exists.

¹We essentially rely on a subset of propositional logic with integer and array theory.

2.1. Precondition Reachability Classification

For each item I_i , we classify its precondition P_i relative to B into three categories:

$$\begin{aligned} \text{ALWAYS} : & \quad \text{UNSAT}(B \wedge \neg P_i) \\ \text{NEVER} : & \quad \text{UNSAT}(B \wedge P_i) \\ \text{CONDITIONAL} : & \quad \text{SAT}(B) \end{aligned}$$

An **ALWAYS** item is presented to every respondent; **NEVER** means no valid assignment can satisfy the precondition, indicating a design error (the item is dead code); **CONDITIONAL** is the normal case where the item depends on prior answers.

2.2. Postcondition Classification

We classify postconditions relative to their preconditions into four categories:

$$\begin{aligned} \text{VACUOUS} : & \quad \text{UNSAT}(B \wedge P_i) \\ \text{TAUTOLOGICAL} : & \quad \text{UNSAT}(B \wedge P_i \wedge \neg Q_i) \\ \text{CONSTRAINING} : & \quad \text{SAT}(B \wedge P_i \wedge Q_i) \text{ and } \text{SAT}(B \wedge P_i \wedge \neg Q_i) \\ \text{INFEASIBLE} : & \quad \text{UNSAT}(B \wedge P_i \wedge Q_i) \end{aligned}$$

VACUOUS correlates with **NEVER** reachability—the postcondition is irrelevant. **TAUTOLOGICAL** means the postcondition adds no constraint beyond what domains and the precondition already enforce, suggesting a redundant rule. **CONSTRAINING** is the normal case: the postcondition meaningfully restricts valid responses. **INFEASIBLE** is a design error—no valid response exists when the item is reached.

3. Comprehensive Validation

This section establishes the precise relationship between global and per-item validation, proving what each level can and cannot detect. These results guide the practical choice of validation strategy.

Definition 3.1 (Global Satisfiability Formula). The **global satisfiability formula** for a questionnaire is:

$$\mathcal{F} := B \wedge \bigwedge_{i=1}^n (P_i \Rightarrow Q_i)$$

The questionnaire passes global validation if $\text{SAT}(\mathcal{F})$.

Intuitively, the global formula asks whether there exists *at least one* complete assignment of answers that respects all domain constraints and satisfies all postconditions for every reached item.

Definition 3.2 (Witness Formula). For item I_i , the **witness formula** is $W_i := B \wedge P_i \wedge \neg Q_i$. If $\text{SAT}(W_i)$, there exists an assignment where item I_i is reached but its postcondition is violated. Such an assignment *witnesses* a validation failure.

The diagnostic question is whether W_i is satisfiable: $\text{UNSAT}(W_i)$ means item I_i is *locally valid* (whenever it is reached, its postcondition must hold), while $\text{SAT}(W_i)$ indicates a potential violation. The witness assignment provides a concrete counterexample showing exactly which answer combination causes the failure.

3.1. Key Properties of Global Satisfiability

The following five theorems capture the exact relationships between per-item, global, and path-based validation.

Theorem 3.3 (Soundness of Per-Item Validity). *If $\text{SAT}(B)$ and for every $i \in \{1, \dots, n\}$ we have $\text{UNSAT}(W_i)$, then $\text{SAT}(\mathcal{F})$.*

Proof. Assume $\text{SAT}(B)$ and $\text{UNSAT}(W_i)$ for all i , where $W_i = B \wedge P_i \wedge \neg Q_i$.

Step 1: Transform unsatisfiability to entailment. For each i , $\text{UNSAT}(B \wedge P_i \wedge \neg Q_i)$ means:

$$\nexists \mathbf{a} : \mathbf{a} \models (B \wedge P_i \wedge \neg Q_i)$$

Equivalently, $\forall \mathbf{a} : \mathbf{a} \not\models B$ or $\mathbf{a} \not\models P_i$ or $\mathbf{a} \models Q_i$. Using $(\neg A \vee \neg B \vee C) \equiv ((A \wedge B) \Rightarrow C)$: if $\mathbf{a} \models B \wedge P_i$ then $\mathbf{a} \models Q_i$. By the deduction theorem: $B \models (P_i \Rightarrow Q_i)$.

Step 2: Combine entailments. Since $B \models (P_i \Rightarrow Q_i)$ for each i , by monotonicity of entailment:

$$B \models \bigwedge_{i=1}^n (P_i \Rightarrow Q_i)$$

Step 3: Construct a model for $\mathcal{F}$. By hypothesis, $\text{SAT}(B)$, so there exists $\mathbf{a}$ with $\mathbf{a} \models B$. By Step 2, every model of B is also a model of $\bigwedge_{i=1}^n (P_i \Rightarrow Q_i)$. Therefore:

$$\mathbf{a} \models B \wedge \bigwedge_{i=1}^n (P_i \Rightarrow Q_i) = \mathcal{F}$$

Hence $\text{SAT}(\mathcal{F})$. □

Theorem 3.4 (Local Invalidity Does Not Imply Global Failure). *If $\text{SAT}(W_i)$ for some i , it does not follow that $\text{UNSAT}(\mathcal{F})$.*

The witness W_i only shows that there *exists* a violating assignment; the global check asks whether there *exists* a (possibly different) assignment satisfying all implications simultaneously.

Proof. By counterexample. Consider two items:

- I_1 : age, $S_1 \in [0, 120]$, $P_1 = \text{true}$, $Q_1 = \text{true}$
- I_2 : driving experience, $S_2 \in [0, 100]$, $P_2 = (S_1 \geq 16)$, $Q_2 = (S_2 \leq S_1 - 16)$

The postcondition Q_2 ensures driving experience cannot exceed years since turning 16.

The witness $W_2 = B \wedge (S_1 \geq 16) \wedge (S_2 > S_1 - 16)$ is satisfiable: $S_1 = 20$, $S_2 = 10$ gives 10 years of experience at age 20, violating Q_2 since $10 > 20 - 16 = 4$. Thus $\text{SAT}(W_2)$ —item I_2 is locally invalid.

However, $\mathcal{F}$ is satisfiable: $S_1 = 30$, $S_2 = 5$ satisfies B and $(P_2 \Rightarrow Q_2)$ since $5 \leq 30 - 16 = 14$. Thus $\text{SAT}(\mathcal{F})$ despite $\text{SAT}(W_2)$. The local invalidity shows that *some* respondents could violate the postcondition, but valid completions still exist. $\square$

Theorem 3.5 (Accumulated Constraints Can Cause Global Unsatisfiability). *There exist questionnaires where no item is INFEASIBLE individually (each $\text{SAT}(B \wedge P_i \wedge Q_i)$), yet $\text{UNSAT}(\mathcal{F})$.*

Per-item checks examine each postcondition independently against B , missing conflicts between postconditions that must be satisfied simultaneously during questionnaire execution.

Proof. By counterexample. Consider a satisfaction survey where I_1 asks “Rate your overall experience” with $S_1 \in [1, 100]$, $P_1 = \text{true}$, and postcondition $Q_1 = (S_1 > 50)$ requiring a positive rating. Item I_2 asks “Would you like a follow-up?” with $S_2 \in \{0, 1\}$, $P_2 = \text{true}$, and postcondition $Q_2 = (S_1 < 30)$ intended to trigger follow-up only for dissatisfied respondents. Each item is individually feasible:

- $\text{SAT}(B \wedge P_1 \wedge Q_1)$ with $S_1 = 75$: a positive rating satisfies Q_1
- $\text{SAT}(B \wedge P_2 \wedge Q_2)$ with $S_1 = 20$: a low rating satisfies Q_2

Neither is INFEASIBLE. However, since both preconditions are *true*:

$$\mathcal{F} = B \wedge (\text{true} \Rightarrow S_1 > 50) \wedge (\text{true} \Rightarrow S_1 < 30) = B \wedge (S_1 > 50) \wedge (S_1 < 30)$$

This is unsatisfiable—no S_1 can simultaneously satisfy both constraints. The per-item checks miss this conflict because they evaluate each postcondition against all possible values of S_1 , not accounting for the fact that Q_1 constrains the same variable that Q_2 references. $\square$

3.2. Dependency Graph and Cycle Detection

To ensure questionnaires can be evaluated in a consistent order, we analyze dependencies between items. Construction proceeds in three stages:

Stage 1: Data flow graph. Through static analysis of preconditions, and postconditions (and code blocks after introducing them in Section 4), we construct a directed graph with item vertices and variable vertices. Edges capture data flow: answering item I_j assigns outcome S_j (edge $I_j \rightarrow S_j$).

Stage 2: Item dependency graph. From the data flow graph, we derive $G = (V, E)$ where $V = \{I_1, \dots, I_n\}$ and $(I_j \rightarrow I_i) \in E$ iff there exists a directed path from I_j to I_i in the data flow graph.

Stage 3: Topological sort and cycle detection. We compute a topological ordering of G using Kahn's algorithm [11]: initialize with zero in-degree items, iteratively remove items and their outgoing edges. If edges remain after termination, cycles exist and the questionnaire cannot be evaluated—each item on the cycle depends on others in the cycle.

3.3. Path-Based Validation

Path-based validation addresses a limitation of the global formula and per-item checks: items that appear reachable in isolation but become unreachable when predecessor constraints accumulate.

Definition 3.6 (Execution Path). An **execution path** $\pi = (I_{k_1}, \dots, I_{k_n})$ is a permutation of items respecting the dependency order where each item's precondition is satisfied given the accumulated constraints from preceding items.

Definition 3.7 (Path Constraint Formula). For path $\pi = (I_{k_1}, \dots, I_{k_n})$, the **path constraint formula** is:

$$\Phi_\pi := B \wedge \bigwedge_{j=1}^n (P_{k_j} \wedge Q_{k_j})$$

The path is **feasible** if $\text{SAT}(\Phi_\pi)$.

Note the key difference from the global formula: path constraints use conjunctions $(P_i \wedge Q_i)$ for items on the path, not implications $(P_i \Rightarrow Q_i)$. This ensures that preconditions are actually satisfiable, not just vacuously handled.

Theorem 3.8 (Global Validity is Necessary for Path Validity). *If $\text{UNSAT}(\mathcal{F})$, then for every execution path π , $\text{UNSAT}(\Phi_\pi)$.*

Proof. By contraposition. Suppose $\text{SAT}(\Phi_\pi)$ for some path $\pi = (I_{k_1}, \dots, I_{k_m})$ with satisfying assignment $\mathbf{a}$.

For items on the path: $\mathbf{a} \models P_{k_j}$ and $\mathbf{a} \models Q_{k_j}$, so $\mathbf{a} \models (P_{k_j} \Rightarrow Q_{k_j})$.

For items not on the path: the execution path includes all items whose preconditions hold under $\mathbf{a}$ (by definition, a maximal set respecting the dependency order). Therefore, for items $I_i \notin \pi$, we have $\mathbf{a} \not\models P_i$, making $(P_i \Rightarrow Q_i)$ vacuously true under $\mathbf{a}$.

Combining: $\mathbf{a} \models B \wedge \bigwedge_{i=1}^n (P_i \Rightarrow Q_i) = \mathcal{F}$, giving $\text{SAT}(\mathcal{F})$.

By contraposition: $\text{UNSAT}(\mathcal{F})$ implies $\text{UNSAT}(\Phi_\pi)$ for all π . $\square$

Theorem 3.9 (Global Validity Does Not Guarantee All Items Reachable). *There exist questionnaires where $\text{SAT}(\mathcal{F})$ but some conditionally-reachable item is unreachable on every feasible path.*

Proof. We provide an example. Let $I_1 = (S_1 \in [1, 100], P_1 = \text{true}, Q_1 = (S_1 > 80))$ and $I_2 = (S_2 \in \{0, 1\}, P_2 = (S_1 < 50), Q_2 = \text{true})$. The dependency graph gives $I_1 \rightarrow I_2$ since P_2 references S_1 .

Per-item analysis: $P_2 = (S_1 < 50)$ is **CONDITIONAL**: $\text{SAT}(B \wedge S_1 < 50)$ with $S_1 = 25$. Neither item is **INFEASIBLE**.

Global analysis:

$$\mathcal{F} = B \wedge (\text{true} \Rightarrow S_1 > 80) \wedge ((S_1 < 50) \Rightarrow \text{true}) \equiv B \wedge (S_1 > 80)$$

This is satisfiable with $S_1 = 90$, so $\text{SAT}(\mathcal{F})$.

Path analysis: Any path including I_2 requires:

$$\Phi_\pi = B \wedge (S_1 > 80) \wedge (S_1 < 50)$$

which is unsatisfiable. Item I_2 is classified **CONDITIONAL** by per-item analysis, passes the global check, yet is unreachable on every valid path. This is **dead code**: the global formula masks I_2 's unreachability because $(P_2 \Rightarrow Q_2) = ((S_1 < 50) \Rightarrow \text{true})$ is vacuously true when Q_1 forces $S_1 > 80$. $\square$

3.4. Accumulated Reachability

To detect dead code systematically without enumerating all paths—which would face the same exponential blow-up as testing—we define accumulated reachability, which accounts for constraints imposed by predecessor items in the dependency order.

Definition 3.10 (Accumulated Reachability). For item I_i , let $\text{Pred}(i) = \{j : I_j \prec I_i\}$ be predecessor items in the topological order. The **accumulated constraint formula** is:

$$\mathcal{A}_i := B \wedge \bigwedge_{j \in \text{Pred}(i)} (P_j \Rightarrow Q_j)$$

Item I_i is **accumulated-reachable** if $\text{SAT}(\mathcal{A}_i \wedge P_i)$.

If $\text{UNSAT}(\mathcal{A}_i \wedge P_i)$ for a conditionally-reachable item, it is dead code—its precondition can never be satisfied given accumulated predecessor constraints.

Example 3.11. Consider three items:

$$I_1 = (\text{annual income}, S_1 \in [0, 1000000], P_1 = \text{true}, Q_1 = \text{true})$$

$$I_2 = (\text{tax bracket}, S_2 \in \{1, 2, 3\}, P_2 = \text{true}, Q_2 = (S_1 \geq S_2 \cdot 10000))$$

$$I_3 = (\text{low-income assistance}, S_3 \in \{0, 1\}, P_3 = (S_1 < 10000), Q_3 = \text{true})$$

There are two possible topological orders $I_1 \rightarrow I_2 \rightarrow I_3$ and $I_1 \rightarrow I_3 \rightarrow I_2$ (since both I_2 and I_3 refer to S_1). If we consider the first order, per-item analysis classifies P_3 as **CONDITIONAL**: $\text{SAT}(B \wedge S_1 < 10000)$ holds with $S_1 = 5000$. However, with predecessor set $\text{Pred}(3) = \{1, 2\}$:

$$\mathcal{A}_3 \wedge P_3 = B \wedge \underbrace{(S_1 \geq S_2 \cdot 10000)}_{Q_2} \wedge \underbrace{(S_1 < 10000)}_{P_3}$$

This is unsatisfiable: since $S_2 \geq 1$, Q_2 forces $S_1 \geq 10000$, contradicting P_3 . Item I_3 (low-income assistance) is dead code—the postcondition Q_2 sneaks a lower bound on S_1 through I_2 , making I_3 unreachable even though per-item analysis sees no problem. This order-independent fact is captured exactly by the path constraint formula Φ_π (Definition 3.7), which conjoins $P_i \wedge Q_i$ over all path items and is unsatisfiable here for both orders; in practice we detect such cases by evaluating accumulated reachability under a fixed canonical topological order.

3.5. Validation Hierarchy

The presented validation approaches can be combined into a hierarchy. Furthermore, we can utilise SMT-solvers (in our case, Z3 [15]) for instance-based verification: the presented validation formulas can be automatically checked for concrete questionnaires. Questionnaire validation operates at four levels of increasing thoroughness and computational cost:

1. **Per-item validation** (Sections 2.1 and 2.2) classifies each item independently by testing its precondition reachability and postcondition feasibility. This is the fastest level, as item analyses are parallelizable and each requires only 2–3 SMT queries.
2. **Global validation** (Definition 3.1) performs a single satisfiability check on the global formula $\mathcal{F}$, determining whether at least one valid questionnaire completion exists.
3. **Cycle detection** (Section 3.2) analyzes the dependency graph for circular dependencies, which indicate that the item ordering cannot be resolved.
4. **Path-based validation** (Definition 3.10) verifies accumulated reachability for each conditional item under predecessor postconditions. This is the most thorough level, capable of detecting dead code that earlier levels miss, at the cost of one SMT query per item with accumulated constraints.

Validation outcomes. The hierarchy yields six diagnostic cases: (1) all W_i UNSAT implies all levels pass (Theorem 3.3); (2) some $\text{SAT}(W_i)$ with $\text{UNSAT}(\mathcal{F})$ means conflicting postconditions—no valid path exists (Theorems 3.5 and 3.8); (3) $\text{SAT}(\mathcal{F})$ with all items accumulated-reachable is the ideal outcome; (4) $\text{SAT}(\mathcal{F})$ with some items not accumulated-reachable indicates dead code (Theorem 3.9); (5) any INFEASIBLE item is always a design error; (6) any NEVER-reachable item signals an unsatisfiable precondition.

4. Implementation of the Formal Model

We introduce the Questionnaire Markup Language (QML, specified in [21]), a YAML-based specification language where preconditions and postconditions are expressed as Python boolean expressions. Items may contain imperative code blocks

for intermediate computations. These code blocks must be transformed into Z3 constraints (influencing pre- and postconditions), a non-trivial challenge since Z3 variables are immutable while Python allows mutation, conditional branching, and iteration. We present Example 3.11 in Section A with QML syntax, and refer to [20] for further, more involved examples.

4.1. Compilation Strategy

The compiler [22] transforms Python code blocks into Z3 formulas through three key techniques:

Single Static Assignment (SSA). SSA [4] ensures each variable is assigned exactly once by creating versioned copies at each reassignment point (e.g., $\text{total}_0 = 0$, $\text{total}_1 = \text{total}_0 + x$, $\text{total}_2 = \text{total}_1 + y$). Each SSA variable maps directly to an immutable Z3 expression, eliminating mutation while preserving the semantics.

Phi functions for conditional branching. When control flow paths merge after conditional statements, *phi functions* [4] select the appropriate variable version: $x_{\text{new}} = \phi(\text{cond}, x_{\text{true}}, x_{\text{false}})$, mapping to Z3's $\text{If}(\text{cond}, x_{\text{true}}, x_{\text{false}})$. Nested conditionals produce nested *If* expressions, preserving full semantic equivalence with the original Python code.

Bounded loop unrolling. Since Z3 cannot represent arbitrary loops, we employ bounded loop unrolling [3, 13] for *for* loops with statically determinable bounds (range calls with compile-time constants, container literals). Each iteration is expanded into sequential SSA-transformed constraints. The compiler unrolls up to 20 iterations, issuing a warning (not an error) for larger loops and using partial results for validation.

4.2. Unified Analysis

A distinguishing feature of this approach is the unified analysis of both declarative conditions and imperative code blocks within a single constraint system. Variables introduced in code blocks flow through the dependency graph alongside outcome variables; therefore, we extend the data flow graph construction step to include these variables too (i.e. data flow edges can contain these new variables too), but item dependencies and the topological sort is determined exactly the same way as we discussed in Section 3.2.

Only code portions that ultimately influence a precondition or postcondition are compiled; auxiliary computations are ignored. This selective compilation reduces constraint complexity without affecting verification completeness.

The compiler supports a Python subset covering common questionnaire patterns: arithmetic, comparisons, Boolean operators, *if/elif/else*, bounded *for* loops, variable assignment, list literals, and attribute access [20]. Vector and matrix operations map to Z3's array theory.

4.3. Evaluation

We converted 12 real-world questionnaires (26–303 pages) from their original imperative format into QML and ran the full validation hierarchy on each [19]. In these traditional instruments, branching between questions is expressed imperatively through skip logic or explicit *GOTO* jump instructions that route the respondent to a target question. The corpus spans major national surveys from Canada, the U.S., and Europe, totalling over 4,200 items in 12 QML files. A companion benchmark harness measures how validation time and memory scale with questionnaire size and complexity [23]. We provide a summary extracted from this benchmark in Section B.

Conversion methodology. Each questionnaire was converted through a three-phase LLM-assisted pipeline. First, the source PDF was OCR-extracted and parsed into a structured question inventory cataloguing every item, its response options, and its imperative routing instructions (*GOTO* targets, interviewer checks, filter conditions). Second, the inventory was transformed into declarative QML by systematically converting *GOTO*-based skip patterns into preconditions and interviewer consistency checks into postconditions, following eight identified conversion patterns (e.g., forward *GOTO*s become disjunctive preconditions over skipped items; range-check interviewer instructions become postconditions on the preceding question). Third, the resulting QML files were validated using the full four-level hierarchy described in Section 3.

Structural equivalence. All *GOTO*-based routing was successfully expressed as preconditions, confirming that traditional imperative questionnaires can be faithfully translated into the declarative model without loss of semantics or structure. All 12 questionnaires passed the full validation hierarchy. This is expected—these are mature, production instruments that have been refined over many survey cycles.

Postconditions and data consistency. Despite their structural soundness, the questionnaires reveal a systematic gap in data quality assurance. Half of the 12 questionnaires contain zero postconditions—no range checks, no cross-field edits, no logical constraints of any kind. Across the entire corpus, only 63 constraining postconditions exist (1.5% of items), concentrated almost entirely in a single questionnaire. Traditional practice handles such inconsistencies through post-collection data cleaning rather than preventing them at the point of collection. QML postconditions enable a shift from correction to prevention: at runtime the survey engine enforces them, rejecting inconsistent responses before they enter the dataset.

5. Related Work

Questionnaire logic formalization. The formal study of questionnaires began with Picard [18], who established the mathematical foundations of question-

naire theory as a branch of combinatorics. Fellegi and Holt [9] introduced edit rules—consistency constraints on survey responses—and systematic procedures for detecting and correcting violations. Willenborg [25] formalized routing structures (skip patterns) and edit structures as separate concepts, later proving [26] that checking their combined consistency is NP-complete. We extend this line from post-collection cleaning to pre-deployment verification via SMT solving.

Graph-based approaches. Fagan and Greenberg [7] pioneered graph representations for skip patterns in census questionnaires. Bethlehem and Hundepool [1] developed TADEQ, a comprehensive tool for documentation and analysis of electronic questionnaires, using directed graphs to visualize routing logic. Elliott [6] derived basis path sets from questionnaire graphs for systematic testing. Schiopu-Kratina et al. [24] formalized the relationship between questionnaires and labeled directed graphs, emphasizing that edges must carry conditions. Our dependency graph construction extends this tradition with SMT-based formal verification.

SMT solvers and bounded model checking. SMT solvers [2, 15, 16] combine Boolean satisfiability with decision procedures for background theories (integers, arrays, bitvectors). The DPLL(T) architecture [16] enables efficient handling of constraints mixing Boolean logic with integer arithmetic—exactly the combination questionnaire validation requires. Bounded model checking [3, 13] applies similar techniques to verify software properties by unrolling loops to a bounded depth, analogous to our loop unrolling strategy. Our compilation approach also parallels symbolic execution [12].

6. Conclusion

We have presented a formal framework for questionnaire validation based on pre- and postconditions inspired by Hoare Logic. The framework provides a four-level validation hierarchy (per-item, global, cycle detection, and path-based) with theorems characterizing what each level can and cannot detect. A compiler bridges practical questionnaire authoring in the QML language [22] with formal verification via Z3 SMT constraints. Evaluation on 12 real-world questionnaires (over 4,200 items) confirmed that imperative designs translate faithfully into the declarative model, that mature instruments pass structural validation as expected, and that postconditions address a systematic gap by shifting data consistency enforcement from post-collection cleaning to the point of collection.

Future work. We plan to extend language support (functions, exception handling), develop incremental validation for efficient re-checking after modifications, and apply the framework to domains beyond surveys—compliance audits, medical triage protocols, and safety-critical system checklists—where formally verified conditional logic is essential.

References

- [1] J. G. BETHLEHEM, A. HUNDEPOOL: *TADEQ: A Tool for Documentation and Analysis of Electronic Questionnaires*, Journal of Official Statistics 20.2 (2004), pp. 233–263.
- [2] A. BIERE, M. HEULE, H. VAN MAAREN, T. WALSH: *Handbook of Satisfiability*, vol. 185, Frontiers in Artificial Intelligence and Applications, IOS Press, 2009.
- [3] E. CLARKE, A. BIERE, R. RAIMI, Y. ZHU: *Bounded Model Checking Using Satisfiability Solving*, Formal Methods in System Design 19.1 (2001), pp. 7–34, DOI: [10.1023/A:1011276507260](https://doi.org/10.1023/A:1011276507260).
- [4] R. CYTRON, J. FERRANTE, B. K. ROSEN, M. N. WEGMAN, F. K. ZADECK: *Efficiently Computing Static Single Assignment Form and the Control Dependence Graph*, ACM Transactions on Programming Languages and Systems 13.4 (1991), pp. 451–490, DOI: [10.1145/115372.115320](https://doi.org/10.1145/115372.115320).
- [5] E. W. DIJKSTRA: *Notes on Structured Programming*, in: Structured Programming, ed. by O.-J. DAHL, E. W. DIJKSTRA, C. A. R. HOARE, London: Academic Press, 1972, pp. 1–82.
- [6] D. ELLIOTT: *The Application of Graph Theory to the Development and Testing of Survey Instruments*, Survey Methodology 38.2 (2012), pp. 195–206.
- [7] J. T. FAGAN, B. V. GREENBERG: *Using Graph Theory to Analyze Skip Patterns in Questionnaires*, in: Proceedings of the Section on Survey Research Methods, American Statistical Association, 1988, pp. 193–198.
- [8] G. FEENEY, S. FEENEY: *On the Logical Structure of Census and Survey Questionnaires*, Genus 75.1 (2019), pp. 1–23, DOI: [10.1186/s41118-019-0065-y](https://doi.org/10.1186/s41118-019-0065-y).
- [9] I. P. FELLEGI, D. HOLT: *A Systematic Approach to Automatic Edit and Imputation*, Journal of the American Statistical Association 71.353 (1976), pp. 17–35, DOI: [10.1080/01621459.1976.10481472](https://doi.org/10.1080/01621459.1976.10481472).
- [10] C. A. R. HOARE: *An Axiomatic Basis for Computer Programming*, Communications of the ACM 12.10 (1969), pp. 576–580, DOI: [10.1145/363235.363259](https://doi.org/10.1145/363235.363259).
- [11] A. B. KAHN: *Topological Sorting of Large Networks*, Communications of the ACM 5.11 (1962), pp. 558–562, DOI: [10.1145/368996.369025](https://doi.org/10.1145/368996.369025).
- [12] J. C. KING: *Symbolic Execution and Program Testing*, Communications of the ACM 19.7 (1976), pp. 385–394, DOI: [10.1145/360248.360252](https://doi.org/10.1145/360248.360252).
- [13] D. KROENING, M. TAUTSCHNIG: *CBMC—C Bounded Model Checker*, in: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Lecture Notes in Computer Science, Springer, 2014, pp. 389–391, DOI: [10.1007/978-3-642-54862-8_26](https://doi.org/10.1007/978-3-642-54862-8_26).
- [14] J. LERNER: *The Promise and Pitfalls of AI-Augmented Survey Research*, tech. rep., NORC at the University of Chicago, 2024.
- [15] L. DE MOURA, N. BJØRNER: *Z3: An Efficient SMT Solver*, in: Tools and Algorithms for the Construction and Analysis of Systems (TACAS), vol. 4963, Lecture Notes in Computer Science, Springer, 2008, pp. 337–340, DOI: [10.1007/978-3-540-78800-3_24](https://doi.org/10.1007/978-3-540-78800-3_24).
- [16] R. NIEUWENHUIS, A. OLIVERAS, C. TINELLI: *Solving SAT and SAT Modulo Theories: From an Abstract Davis–Putnam–Logemann–Loveland Procedure to DPLL(T)*, Journal of the ACM 53.6 (2006), pp. 937–977, DOI: [10.1145/1217856.1217859](https://doi.org/10.1145/1217856.1217859).
- [17] F. OLIVOS, M. LIU: *ChatGPTTest: Opportunities and Cautionary Tales of Utilizing AI for Questionnaire Pretesting*, Field Methods (2024), DOI: [10.1177/1525822X241280574](https://doi.org/10.1177/1525822X241280574).
- [18] C. PICARD: *Théorie des questionnaires*, Paris: Gauthier-Villars, 1965.
- [19] P. SÁGHELYI: *Questionnaire Markup Language (QML): Evaluation Corpus of Real-World Questionnaires*, Accessed: 2026-06-18, 2026, URL: <https://github.com/askalot-io/QML/tree/main/evaluation>.

- [20] P. SÁGHELYI: *Questionnaire Markup Language (QML): Example Specifications*, Accessed: 2026-06-18, 2026, URL: <https://github.com/askalot-io/QML/tree/main/tests/fixtures>.
- [21] P. SÁGHELYI: *Questionnaire Markup Language (QML): Language Specification (JSON Schema and EBNF Grammar)*, Accessed: 2026-06-18, 2026, URL: https://github.com/askalot-io/QML/tree/main/askalot_qml/schema.
- [22] P. SÁGHELYI: *Questionnaire Markup Language (QML): Reference Implementation and Compiler*, Accessed: 2026-03-30, 2026, URL: <https://github.com/askalot-io/QML>.
- [23] P. SÁGHELYI: *Questionnaire Markup Language (QML): Validator Scaling Benchmark Harness*, Accessed: 2026-06-18, 2026, URL: <https://github.com/askalot-io/QML/tree/main/benchmark>.
- [24] I. SCHIOPU-KRATINA, C. M. ZAMFIRESCU, K. TRÉPANIÉ, L. MARQUES: *Survey Questionnaires and Graphs*, Electronic Journal of Statistics 9.2 (2015), pp. 2005–2035, DOI: [10.1214/15-EJS1067](https://doi.org/10.1214/15-EJS1067).
- [25] L. WILLENBORG: *Computational Aspects of Survey Data Processing*, PhD thesis, University of Amsterdam, 1988.
- [26] L. WILLENBORG: *Testing and Checking Questionnaires*, in: Proceedings of the ASA Section on Survey Research Methods, American Statistical Association, 1995, pp. 1032–1037.

A. QML Source for the Running Example

Listing 1. QML specification for the income survey of Example 3.11.

```

1 qmlVersion: "2.0"
2
3 questionnaire:
4   title: "Income Survey with Dead Code"
5
6   blocks:
7     - id: block1
8       kind: Group
9       title: "Financial Information"
10      items:
11        - id: q_income
12          kind: Question
13          title: "What is your annual income?"
14          input:
15            control: Editbox
16            min: 0
17            max: 1000000
18
19        - id: q_tax_bracket
20          kind: Question
21          title: "Select your tax bracket"
22          input:
23            control: Radio
24            labels:
25              1: "Low"
26              2: "Medium"
27              3: "High"
28          postcondition:
29            - predicate: q_income.outcome >= q_tax_bracket.outcome * 10000
30              hint: "Income must cover the tax bracket's lower bound"
31
32        - id: q_low_income_assist
33          kind: Question
34          title: "Would you like information about low-income assistance programs?"
35          input:
36            control: Radio
37            labels:
38              0: "No"
39              1: "Yes"
40          precondition:
41            - predicate: q_income.outcome < 10000

```

B. QML Benchmark Data

To characterise how validation cost scales, a companion harness [23] generates synthetic QML questionnaires that vary one structural axis at a time – item count, number of preconditions, number of postconditions, and dependency depth – and times the full per-item validation pipeline (Section 3.5). Each questionnaire is validated in a fresh subprocess so that peak memory reflects Z3's native allocations;

reported values are medians of five repetitions after one warm-up run. The measurements were collected with Z3 4.15.3 and the QML compiler [22] on a single thread of an x86-64 Linux host.

Wall-clock time is split into three phases: YAML *parse* and AST construction, constraint construction, and the Z3 *solve*; we plot parse, Z3 solve, and their sum (*total*; constraint construction is the small remainder). Figure 1 shows that the Z3 solve dominates at scale and is by far most sensitive to the number of preconditions, whereas dependency depth has negligible effect. Figure 2 shows peak memory remains within a narrow $\sim 49\text{--}61$ MiB band throughout, so validation is compute-bound rather than memory-bound at these sizes.

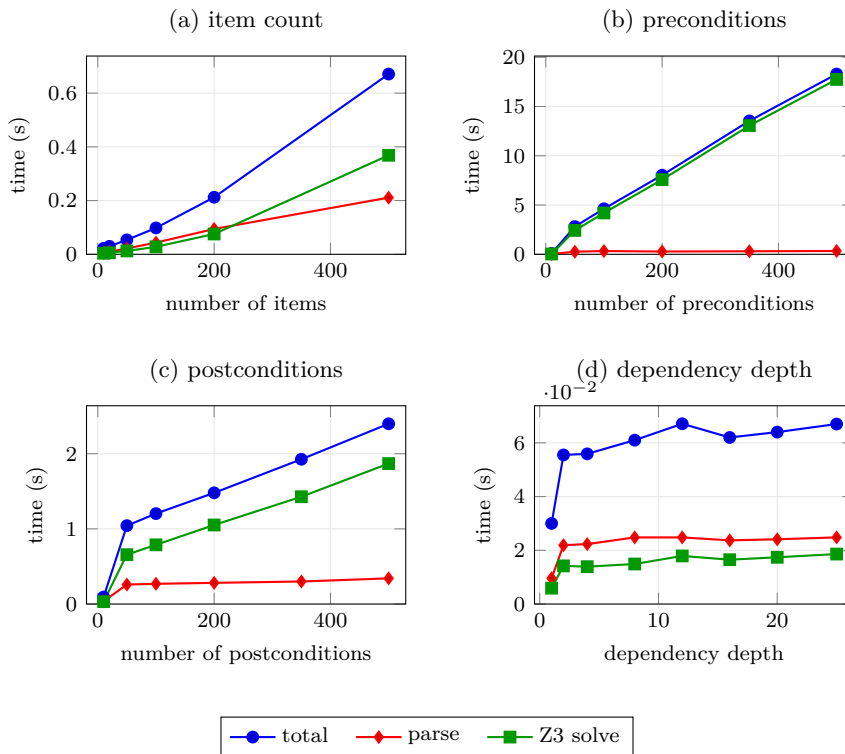


Figure 1. QML validation wall-clock time vs. questionnaire size and complexity, decomposed into end-to-end *total*, YAML *parse*/AST build, and the isolated Z3 *solve*. Each marker is the median of five repetitions. The Z3 solve dominates and grows steeply with the number of preconditions (panel b); item count grows roughly linearly (a); postconditions grow mildly (c); dependency depth is essentially free (d, note the 10^{-2} scale).

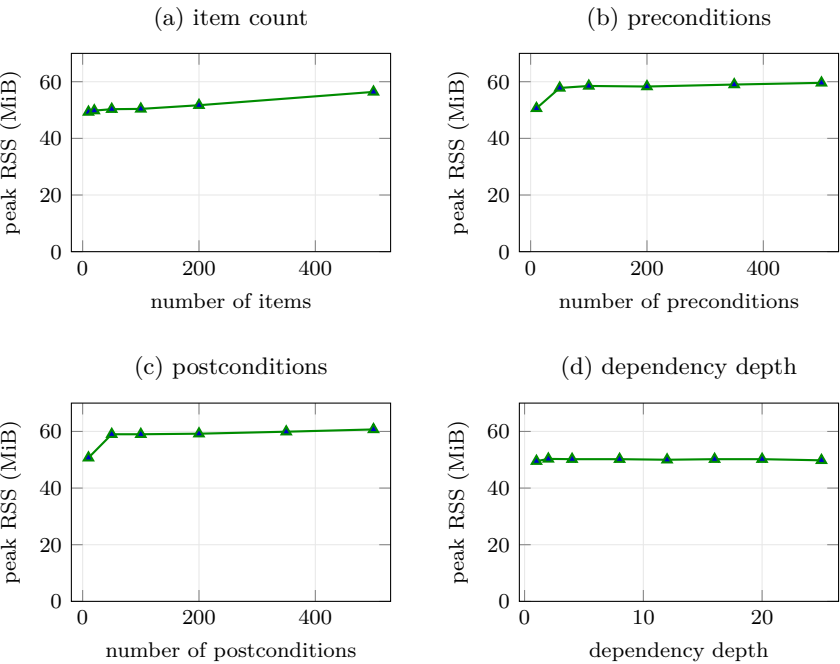


Figure 2. Peak resident set size (whole-process RSS, including Z3’s native allocations) for the same sweeps. Memory stays within a narrow ~49–61 MiB band across every configuration (axes start at 0 to make the flatness explicit), confirming that validation is compute-bound, not memory-bound, at these scales.



An Empirical Study of HomePlug AV2 PowerLine Performance in Real-World Residential Electrical Networks

Szabolcs Szilágyi

Faculty of Informatics, University of Debrecen
szilagyi.szabolcs@inf.unideb.hu

Abstract. PowerLine Communication (PLC) represents a class of wired communication technologies that leverage existing low- and medium-voltage electrical wiring to transmit high-frequency modulated data signals, enabling network connectivity without the need for dedicated cabling. While PLC has gained significant traction as a practical alternative to Wi-Fi extenders and Ethernet cabling for in-home broadband distribution, empirical evidence on its real-world performance – particularly under diverse electrical topologies and noise conditions—remains limited. This paper presents a controlled measurement study of a commercial HomePlug AV2 1000 Mbps kit (TP-Link TL-WPA8631P + TL-PA8010P), operated exclusively in wired Ethernet mode with Wi-Fi disabled. We evaluate throughput and jitter across multiple residential scenarios, including same-phase baseline measurements, the use of power strips, and the presence of electromagnetic interference (EMI) sources. Our findings show that although the data throughput provided by PLC technology falls short of the rates advertised by vendors, it is nevertheless capable of maintaining stable communication over distances of several tens of meters. Jitter remains at a negligible level, which makes the technology suitable for multimedia transmission. Furthermore, neither the use of power-strip extensions nor the presence of electromagnetic interference introduces any substantial degradation in PLC-based data delivery. These results highlight both the progress and the constraints inherent to PLC systems, providing useful insights for smart-home integrators and emphasizing how strongly PLC performance depends on the characteristics of a home’s electrical network.

Keywords: PLC, HomePlug AV2, in-home network, performance, smart home

1. Introduction

PowerLine Communication (PLC) has emerged as a significant networking trend over the past decade. While initially introduced in foundational networking curricula as an alternative connectivity method, its practical deployment has grown substantially. PLC technology offers a viable solution for extending network coverage across multiple floors or rooms without the need for invasive cabling, drilling through reinforced concrete, or installing aesthetically intrusive cable ducts. After observing the practical benefits of PLC deployments over a two-year period, and noting the lack of empirical performance data in recent literature, this study aims to examine the advantages and limitations of PLC technology through detailed, industry-standard performance measurements [1–5, 7, 10].

PLC utilizes the existing electrical wiring infrastructure for data transmission [8, 9]. Deployment requires only a PLC kit containing at least two adapters. The first adapter is connected to a wall socket and to the wired Ethernet network. A second adapter is plugged into another outlet on the same electrical phase elsewhere in the building, allowing it to receive the network signals injected by the first adapter and provide connectivity—either wired or wireless, depending on the model.

Recent studies have highlighted the potential of PLC in smart grid and IoT applications [2, 3], while others have focused on physical layer security and broadband service limitations [9, 10]. However, as noted by [8], real-world performance can vary significantly based on local electrical topology. Our empirical results align with the observations of Ribeiro et al. [6] regarding the complementary nature of PLC and wireless systems. Specifically, our measurements demonstrate that while throughput drops compared to nominal values, latency (jitter) remains highly stable, making PLC a viable and reliable option for specific multimedia and smart-home applications where traditional cabling is impractical.

2. Operating Principle of PLC Technology

PowerLine Communication operates by superimposing a high-frequency data signal onto the standard 230 V / 50 Hz AC mains voltage. Although these two signals propagate along the same physical conductor, they occupy distinct frequency domains and therefore do not interfere with one another. Mathematically, the combined waveform is expressed as the sum of v_{mains} , representing the 220 V AC voltage, and v_{data} , representing the approximately 5 V data signal.

$$v_{\text{mains}}(t) = V_{\text{peak}} \sin(\omega t)$$

where $V_{\text{peak}} = 325$ V (because $230 \cdot \sqrt{2} \approx 325$ V) and $\omega = 2\pi \cdot 50$ rad/s

$$v_{\text{mains}}(t) = 325 \sin(2\pi \cdot 50t)$$

$$v_{\text{data}}(t) = V_{\text{data}} \sin(\omega_{\text{data}} t)$$

where $V_{\text{data}} = 5 \text{ V}$ and $\omega_{\text{data}} = 2\pi \cdot 10^6 \text{ rad/s}$

$$v_{\text{data}}(t) = 5 \sin(2\pi \cdot 10^6 t)$$

In a PLC-based communication:

$$v_{\text{line}}(t) = v_{\text{mains}}(t) + v_{\text{data}}(t)$$

$$v_{\text{line}}(t) = 325 \sin(2\pi \cdot 50t) + 5 \sin(2\pi \cdot 10^6 t)$$

They do not interfere because, in a linear system:

- signals at different frequencies are orthogonal, and
- they can be effectively separated using filters (e.g., OFDM – Orthogonal Frequency Division Multiplexing).

Mathematically:

$$\int_0^T \sin(2\pi 50t) \cdot \sin(2\pi 10^6 t) dt = 0$$

While the mathematical orthogonality described above holds true in ideal linear systems, it is important to note that real-world electrical networks do not perfectly meet these conditions. Practical power grids introduce non-linearities, impedance variations, and background noise from various connected devices. Consequently, real-world PLC systems must employ robust modulation techniques (such as OFDM) and advanced error correction to mitigate these interferences. This discrepancy between ideal theoretical models and practical network behavior is the primary motivation for the empirical measurement approach presented in this paper.

Figure 1 depicts the composite signal on the electrical network: the red component has high amplitude and low frequency, whereas the blue component exhibits low amplitude and high frequency.

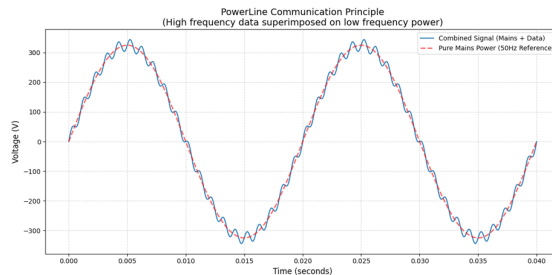


Figure 1. PLC composite signal.

Figure 2 presents the separation of the two signals on a logarithmic scale, highlighting the substantial spectral distance that enables concurrent, interference-free transmission. On a linear scale, this distance would span several kilometers.

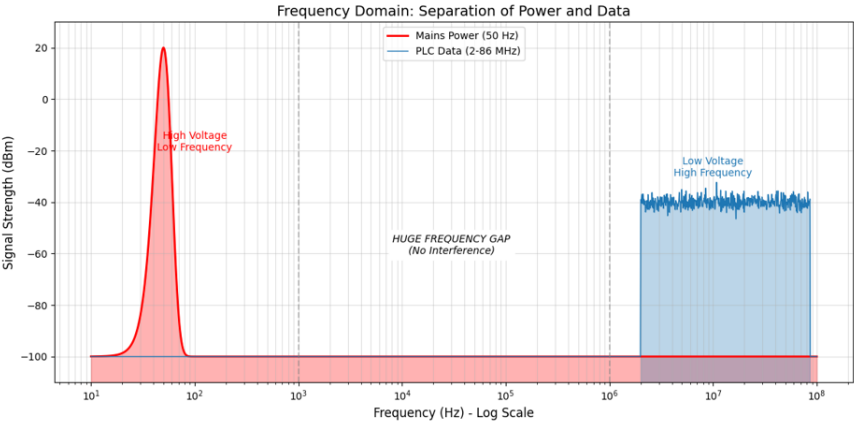


Figure 2. Separation of electrical and data signals during PLC transmission.

3. Types of PLC Technologies

In recent years, several PLC standards have been introduced. Table 1 summarizes the most relevant technologies. For our measurements, we utilized PLC adapters based on the HomePlug standard.

Table 1. Types of PLC standards.¹

Category	Technology	Frequency	Range	Data Rate	Standards
Narrowband	G3-PLC	< 148.5 kHz (EU) < 490 kHz (FCC)	Long	< 280 kbps	ITU-T G.9901 ITU-T G.9903 IEEE 1901.2
Mid-band	HPLC	0.7 - 12 MHz	Middle	150 kbps to 10 Mbps	IEEE 1901.1
Broadband	HomePlug	1.8 - 86 MHz	Short	< 1.3 Gbps	IEEE 1901
Broadband	G.hn	2.0 - 80 MHz	Short	< 1.5 Gbps	ITU-T G.9960 ITU-T G.9961 ITU-T G.9962 ITU-T G.9964

¹Source: <https://nessum.org>

4. Measurement Environment

We conducted all PLC-based measurements in the building of the Faculty of Informatics at the University of Debrecen (see Figure 4). Our measurement environment consisted of two computers, two PLC adapters, and a laptop (see Figure 3).

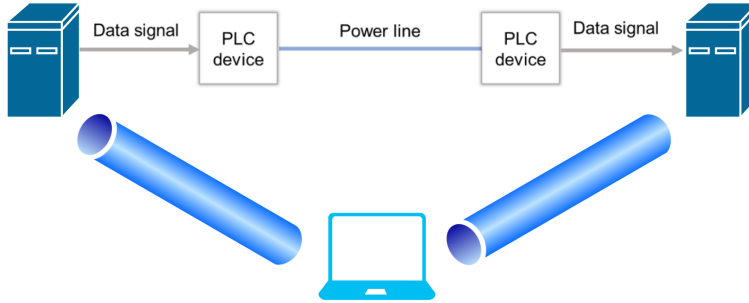


Figure 3. PCL measurement environment.

The computers ran a Linux Server operating system with the following configuration:

- Gigabyte Z77-D3H motherboard with Intel Z77 chipset
- Intel Core i7-3770K 3.50 GHz processor (4 cores / 8 threads)
- 4×4 GB 1600 MHz DDR3 SDRAM
- Intel quad Gigabit Ethernet server adapter
- Ubuntu 24.04 LTS (Noble Numbat), 64-bit
- Linux kernel: 6.8-generic

Initially, the PLC adapters were connected to adjacent electrical outlets. We then progressively increased the distance between outlets to 3 m, 5 m, 10 m, 13 m, and 20 m (see Figure 4).

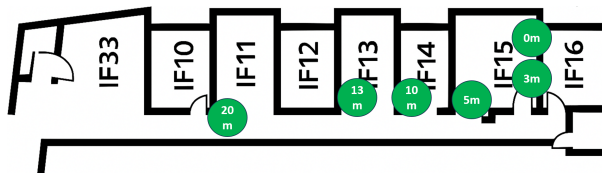


Figure 4. Floor plan of the measurement environment. The numerical values shown indicate the distances between the connectors.



Figure 5. TP-Link HomePlug AV2 1300 kit.

We performed baseline *iperf3* measurements to assess UDP and TCP throughput as well as UDP jitter. The left-side PC was configured as the *iperf3* server, and the right-side PC executed the following commands:

- `iperf3 -u -b 1G -c 192.168.1.1 bidir -t 60`
- `iperf3 -c 192.168.1.1 -t 60 -i 1 -f m`
- `iperf3 -c 192.168.1.1 -u -t 60`

Each measurement lasted 60 seconds and was repeated ten times. The Wi-Fi modules of the PLC adapters were disabled during all tests. The laptop was used for remote SSH management. In addition to baseline measurements, we evaluated the impact of a basic power strip, a surge-protected power strip, and a microwave oven on PLC transmission.

Further environment parameters:

- Location: Faculty of Informatics, University of Debrecen, IF-15 room (see Figure 4)
- Ethernet cables: AMP Netconnect AMPTRAC CAT6 FTP
- Wall outlets: Legrand Valena
- Power strip (basic): no-name
- Surge-protected power strip: Intertek
- Microwave oven: Samsung M182DN, 850 W

5. The TP-Link PLC Kit

For our experiments, we used the TP-Link HomePlug AV2 1300 kit (TL-WPA8631P + TL-PA8010P) (see Figure 5).

Device management can be performed via the *tpPLC* mobile application (see Figure 6)—which requires Wi-Fi—or through a browser-based GUI (see Figure 7). Both interfaces allow convenient configuration of essential parameters such as device naming, LED control, IP configuration, Wi-Fi settings, and firmware updates.

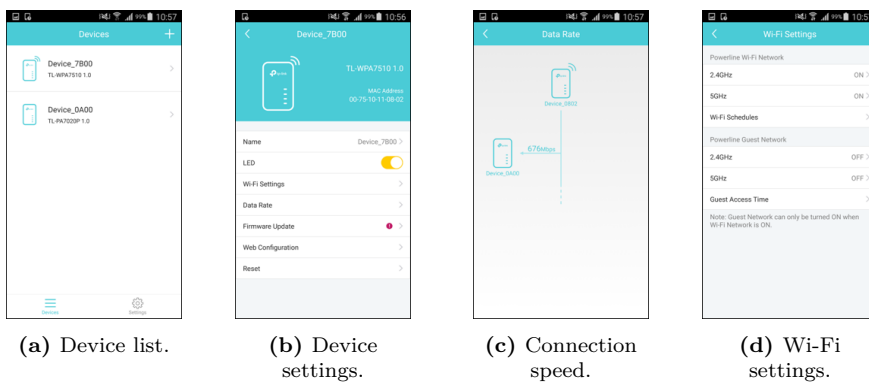


Figure 6. PLC adapter management - *tpPLC*.

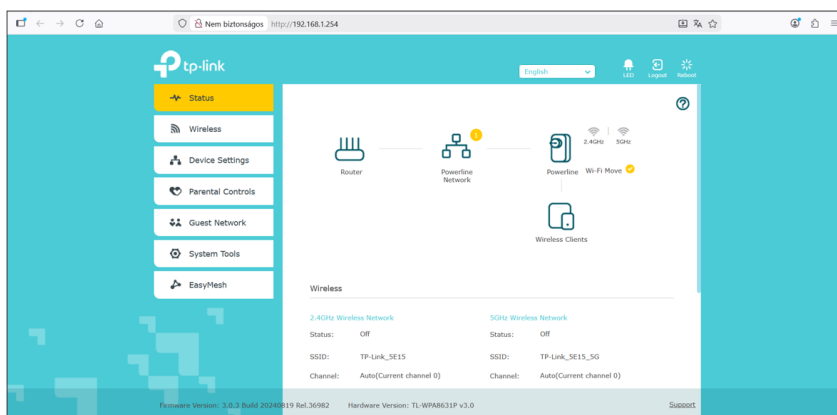


Figure 7. PLC adapter management – tp-link GUI.

6. Measurement Results

6.1. Throughput

We began with UDP-based throughput tests to determine the maximum achievable transmission rate without PLC adapters. Baseline measurements consistently showed 956 Mb/s. TCP throughput, representing real-world file transfer performance, was slightly lower at 941 Mb/s due to TCP header overhead. When PLC adapters were introduced, throughput decreased to roughly half of the baseline and then declined linearly as the distance increased (see Figure 8). The full results are summarized in Table 2).

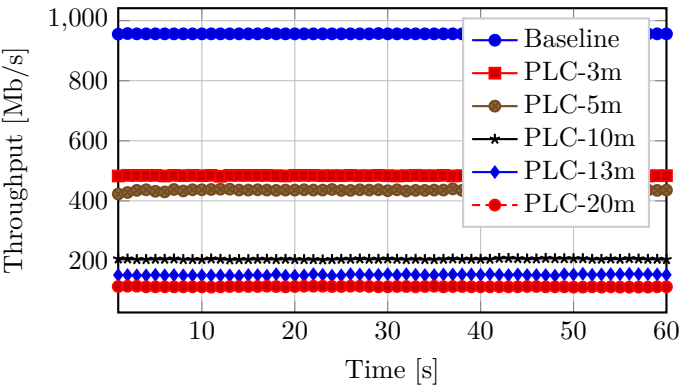


Figure 8. PLC throughput over UDP.

Table 2. PLC throughput over UDP.

Wall socket distance [m]	Baseline Throughput over UDP [Mb/s]	Baseline Throughput over TCP [Mb/s]	PLC Throughput over UDP [Mb/s]	PLC Throughput over TCP [Mb/s]
3	956	941	484	444
5	N/A	N/A	433	378
10	N/A	N/A	203	182
13	N/A	N/A	147	136
20	N/A	N/A	114	104

6.2. Jitter

Jitter measurements were performed exclusively using UDP, as TCP does not support jitter assessment. Baseline jitter averaged 0.021 ms and showed only minimal

degradation at 20 m. Although jitter fluctuations increased with distance, all values remained below 0.1 ms—well within the 30 ms QoS threshold for voice transmission (see Figure 9 and Table 3).

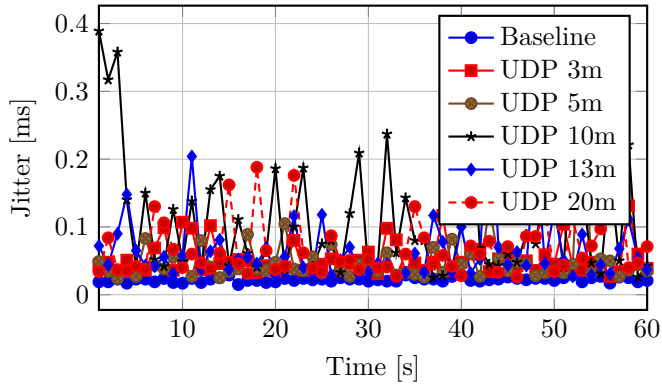


Figure 9. PLC jitter over UDP.

Table 3. PLC jitter over UDP.

Wall socket distance [m]	Baseline Jitter over UDP [ms]	PLC Jitter over UDP [ms]
3	0.021	0.043
5	N/A	0.058
10	N/A	0.075
13	N/A	0.037
20	N/A	0.048

6.3. PLC Through a Basic Power Strip

We then evaluated PLC performance using a conventional no-name power strip. Initially, both PLC adapters were connected to the strip; subsequently, only the second adapter was connected while the strip itself was plugged into a wall outlet 3 m away. Results (see Figure 10, Figure 11 and Table 4) indicate only negligible degradation in both throughput and jitter compared to the baseline.

Table 4. PLC jitter over power strip classic.

Wall socket distance [m]	PLC Throughput over UDP [Mb/s]	PLC Throughput over TCP [Mb/s]	PLC Jitter over UDP [ms]
0	488	464	0.099
3	473	433	0.063

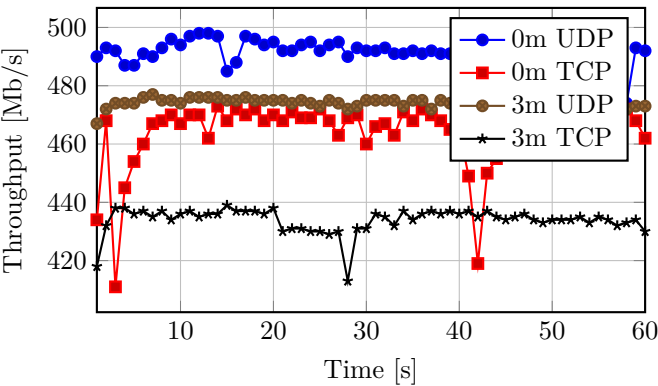


Figure 10. PLC throughput over power strip classic.

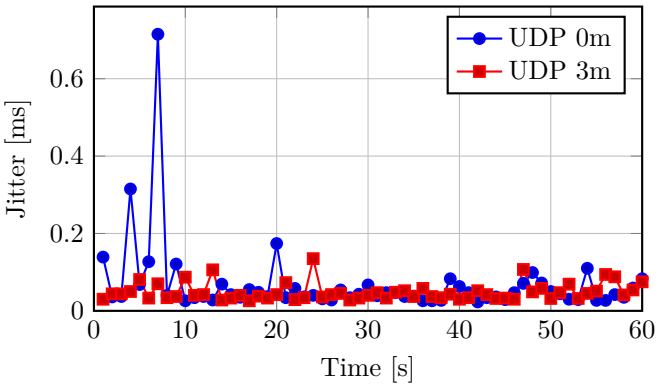


Figure 11. PLC jitter over power strip classic.

6.4. PLC Through a Surge-Protected Power Strip

Next, we repeated the measurement series using a surge-protected power strip. In this case, both throughput and jitter exhibited noticeably greater degradation. Detailed results are provided in Figure 12, Figure 13 and Table 5.

Table 5. PLC throughput and jitter over power strip with protection.

Wall socket distance [m]	PLC Throughput over UDP [Mb/s]	PLC Throughput over TCP [Mb/s]	PLC Jitter over UDP [ms]
0	489	463	0.077
3	396	302	0.103

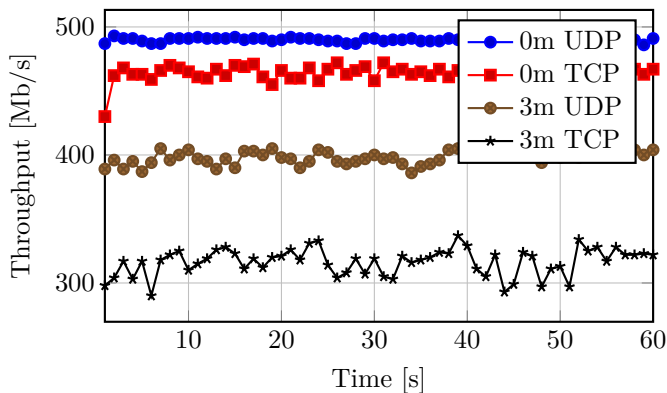


Figure 12. PLC throughput over power strip with protection.

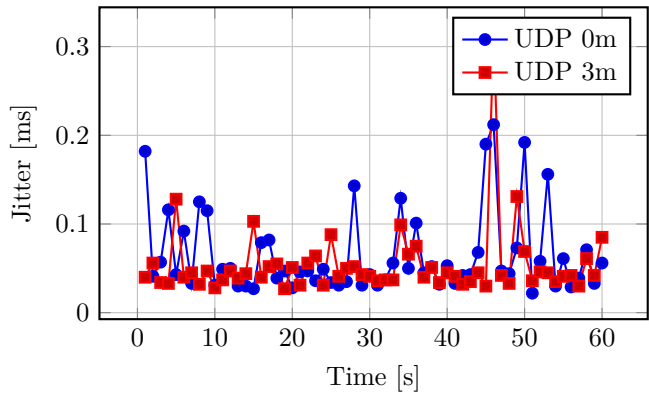


Figure 13. PLC jitter over power strip with protection.

6.5. PLC Operation Near a Microwave Oven

Finally, we tested the influence of a microwave oven on PLC data transmission. Although some articles and anecdotal reports claim that microwave interference can severely degrade PLC communication, our measurements indicate that its effect on wired PLC transmission is minimal. Figure 14, Figure 15 and Table 6 presents the measured results and averaged values.

Table 6. PLC throughput and jitter close to microwave oven.

Wall socket distance [m]	PLC Throughput over UDP [Mb/s]	PLC Throughput over TCP [Mb/s]	PLC Jitter over UDP [ms]
3	468	431	0.051

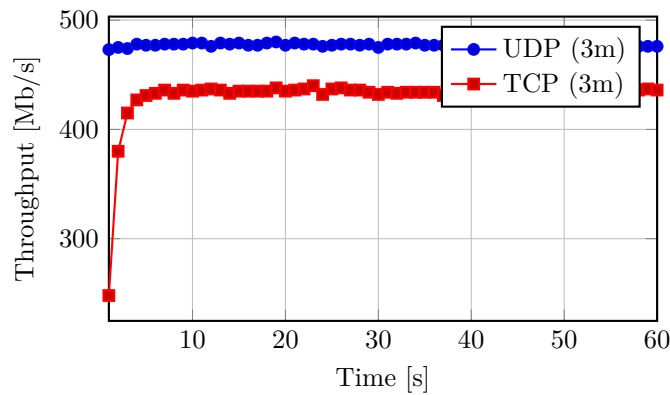


Figure 14. PLC throughput close to microwave oven.

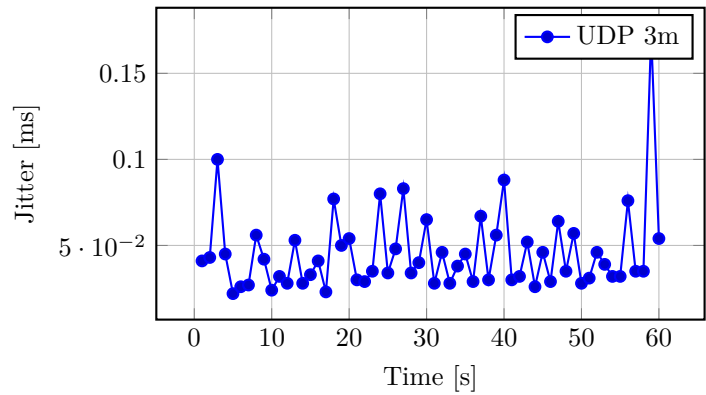


Figure 15. PLC jitter close to microwave oven.

7. Conclusion and Future Work

PLC technology proved highly effective for data communication. Even at 20 meters, it delivered adequate throughput and jitter values. Its performance characteristics place it between wired Ethernet and wireless Wi-Fi [6, 11, 12]. PLC can reliably support both IoT devices and latency-sensitive multimedia applications. The technology is particularly suitable for older multi-story buildings where retrofitting Ethernet cabling is impractical, Wi-Fi performance is suboptimal, drilling through reinforced concrete is undesirable, or wall aesthetics must be preserved.

In future work, we plan to extend our PLC performance analysis to include Wi-Fi environments.

References

- [1] S. M. HASHIM, I. B. AL-MASHHADANI: *Adaptation of powerline communications-based smart metering deployments with IoT cloud platform*, Indonesian Journal of Electrical Engineering and Computer Science 29.2 (2023), pp. 825–837, DOI: [10.11591/ijeecs.v29.i2.pp825-837](https://doi.org/10.11591/ijeecs.v29.i2.pp825-837), URL: <https://doi.org/10.11591/ijeecs.v29.i2.pp825-837>.
- [2] J. HERNANDEZ FERNANDEZ, A. OMRI, R. DI PIETRO: *An Overview of Power Line Communication Networks*, in: Physical Layer Security in Power Line Communications: Fundamentals, Models and Applications, Cham: Springer Nature Switzerland, 2024, pp. 9–20, ISBN: 978-3-031-57349-1, DOI: [10.1007/978-3-031-57349-1_2](https://doi.org/10.1007/978-3-031-57349-1_2), URL: https://doi.org/10.1007/978-3-031-57349-1_2.
- [3] *IEEE Standard for Broadband over Power Line Networks: Medium Access Control and Physical Layer Specifications*, IEEE Std 1901-2010 (2010), pp. 1–1586, DOI: [10.1109/IEEESTD.2010.5678772](https://doi.org/10.1109/IEEESTD.2010.5678772).
- [4] S. H. MAHMOOD, A. M. SALIH, M. I. KHALIL: *Broadband Services on Power Line Communication Systems: A Review*, in: 2019 22nd International Conference on Control Systems and Computer Science (CSCS), 2019, pp. 465–470, DOI: [10.1109/CSCS.2019.00085](https://doi.org/10.1109/CSCS.2019.00085).
- [5] A. NDOLO, İ. H. ÇAVDAR: *Current state of communication systems based on electrical power transmission lines*, Journal of Electrical Systems and Information Technology 8.1 (2021), pp. 1–10, DOI: [10.1186/s43067-021-00028-9](https://doi.org/10.1186/s43067-021-00028-9), URL: <https://doi.org/10.1186/s43067-021-00028-9>.
- [6] M. V. RIBEIRO, M. D. L. FILOMENO, Â. CAMPONOGARA, T. R. OLIVEIRA, T. F. MOREIRA, S. GALLI, H. V. POOR: *Seamless Connectivity: The Power of Integrating Power Line and Wireless Communications*, IEEE Communications Surveys & Tutorials 26.1 (2024), pp. 1–40, DOI: [10.1109/COMST.2023.3327321](https://doi.org/10.1109/COMST.2023.3327321).
- [7] A. M. TONELLO: *Power Line Communication - Generation Next: Vision and Disruptive Directions for Evolution*, in: 2025 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), 2025, pp. 1–7, DOI: [10.1109/SmartGridComm65349.2025.11204612](https://doi.org/10.1109/SmartGridComm65349.2025.11204612).
- [8] S. USTUN ERCAN: *Power line Communication: Revolutionizing data transfer over electrical distribution networks*, Engineering Science and Technology, an International Journal 52 (2024), p. 101680, ISSN: 2215-0986, DOI: <https://doi.org/10.1016/j.jestch.2024.101680>, URL: <https://www.sciencedirect.com/science/article/pii/S2215098624000661>.
- [9] E. UWAEZUOKE, T. G. SWART: *A vulnerability assessment and exploitation analysis of a powerline communication HomePlug AV network adapter*, International Journal of Information Security 24.168 (2025), DOI: [10.1007/s10207-025-01081-5](https://doi.org/10.1007/s10207-025-01081-5), URL: <https://doi.org/10.1007/s10207-025-01081-5>.
- [10] L. YANG, J. FAN, Y. LIU, E. LI, J. PENG, Z. LIANG: *A Review on State-of-the-Art Power Line Inspection Techniques*, IEEE Transactions on Instrumentation and Measurement 69.12 (2020), pp. 9350–9365, DOI: [10.1109/TIM.2020.3031194](https://doi.org/10.1109/TIM.2020.3031194).
- [11] J. ZHANG, C. LONG, H. ZHANG, Q. ZENG, Y. GAO, X. SU, S. LI: *Design and Analysis of Reliable Current Differential Protection Through Improved Power-Line Communication*, IEEE Access 12 (2024), pp. 77421–77428, DOI: [10.1109/ACCESS.2024.3407862](https://doi.org/10.1109/ACCESS.2024.3407862).
- [12] Y. ZHOU, F. GAO, Y. LI, Z. ZHANG, C. ZHOU: *Accurate Impedance Measurement Technology for Power Line Communication Considering Noise Interference*, IET Renewable Power Generation 19.1 (2025), DOI: [10.1049/rpg2.70028](https://doi.org/10.1049/rpg2.70028), URL: <https://doi.org/10.1049/rpg2.70028>.



Structural Node Importance Beyond Classical Centrality: A Comparative Study of Entropy-Based k -Hop Measures

Zsolt Valiskó^a, Tamás Balla^{a,b}, Csaba Biró^{a,b}

^aEszterházy Károly Catholic University

zsoltvaliskojob@gmail.com

balla.tamas@uni-eszterhazy.hu

biro.csaba@uni-eszterhazy.hu

^bEötvös Loránd University

Abstract. This paper investigates whether entropy-based k -hop metrics capture aspects of structural node importance that are not emphasized by classical centrality measures. We compare degree and eigenvector centrality with two entropy-based measures, Entropy-Weighted Redundancy (EWR) and Normalized Entropy Density (NED), on three real-world Twitter-derived networks with different sizes and connectivity patterns. Unlike conventional centrality measures, EWR and NED characterize node importance through the informational diversity and redundancy of the induced k -hop neighborhood. The results show that entropy-based rankings differ substantially from classical rankings and highlight structurally distinctive nodes whose local environments exhibit different structural properties. Furthermore, the observed rankings depend on both the underlying network topology and the neighborhood radius. Overall, the study demonstrates that entropy-based k -hop analysis provides a complementary perspective on structural node importance and motivates further comparative evaluation on other classes of networks.

Keywords: complex networks, social network analysis, structural entropy, k -hop neighborhood, entropy-based metrics, node importance, centrality measures, network topology, structural node analysis

2020 *Mathematics Subject Classification:* 05C82, 94A17, 68R10, 91D30

1. Introduction

Identifying important nodes in complex networks is a fundamental problem in network science, with applications in social network analysis, communication systems, and online platforms [5, 10]. Classical approaches to node importance are mainly based on centrality measures. Degree centrality captures immediate connectivity, whereas eigenvector-based approaches assign greater importance to nodes connected to other important nodes [4, 5, 10]. Although these measures are interpretable and widely used [5, 10], they primarily emphasize direct connectivity or recursively defined prominence and may fail to capture the structural heterogeneity of a node's wider environment [3, 16, 17]. This limitation is especially relevant in large social and communication networks, where structurally distinctive actors are not always the most visible hubs [16, 17]. Nodes with only moderate degree may still occupy distinctive positions if they are embedded in neighborhoods with high diversity, low redundancy, or unusual topology [1–3].

Information theory provides a natural framework for addressing this issue. Since Shannon's foundational work, entropy has been used to quantify uncertainty, diversity, and structural complexity [15]. In network analysis, entropy-based approaches have been used to describe graph organization, quantify structural variability, and reveal properties that remain less visible under conventional connectivity-based summaries [3, 16]. Recent studies also suggest that entropy-oriented methods can highlight structurally distinctive nodes and network regions that may not be emphasized by traditional centrality measures [3, 16, 17].

Within this line of research, k -hop-based entropy formulations are particularly attractive because they extend node analysis beyond immediate adjacency [1, 2]. In our previous work, we introduced two hybrid entropy-based metrics for k -hop environment analysis, Entropy-Weighted Redundancy (EWR) and Normalized Entropy Density (NED), to characterize the structural sensitivity of node neighborhoods in complex networks [1, 2]. These metrics distinguish nodes not only by neighborhood size, but also by the informational arrangement and redundancy profile of that neighborhood [1, 2]. Thus, entropy-based structural analysis complements rather than replaces traditional centrality theory and provides an alternative perspective on structural node importance [1–3, 5, 16, 17].

Although the present study focuses on social networks, similar approaches may be investigated in educational and learning-related settings in future work.

In this paper, we investigate whether entropy-based k -hop metrics capture aspects of structural node importance that are not emphasized by classical centrality measures. Using three Twitter-derived networks with different sizes and connectivity patterns, we compare degree centrality, eigenvector centrality, EWR, and NED in order to examine how classical and entropy-based rankings differ. Our working hypothesis is that entropy-guided measures may emphasize nodes whose surrounding environments exhibit different structural characteristics than those highlighted by degree and eigenvector centrality. [1–3].

The main contribution of this work is twofold. First, we provide an empirical

comparison between classical and entropy-based node-ranking strategies on real-world Twitter-derived networks with different sizes and connectivity patterns [1, 2, 5, 10]. Second, we investigate whether entropy-based k -hop measures emphasize structurally distinctive nodes and structural characteristics that are not reflected by degree and eigenvector centrality alone. Overall, the study provides a comparative perspective on structural node importance and motivates further evaluation of entropy-based measures on other classes of networks and with additional baseline methods.

2. Related Work

The identification of important nodes in networks has long been studied through centrality theory. Foundational work established degree-, closeness-, and betweenness-based interpretations of centrality in social networks [9, 10], while eigenvector-type approaches extended node ranking to recursively defined importance [4, 5]. Related link-structure-based models, such as authority and hub formulations, further showed that node importance depends not only on immediate connectivity but also on broader structural context [13].

Although classical centrality measures remain fundamental, they do not capture every aspect of structural relevance. In many real-world networks, highly connected nodes are not necessarily the only actors with important organizational roles. Nodes with moderate degree may still occupy distinctive positions because they connect heterogeneous neighborhoods or lie in locally non-redundant environments. This limitation has motivated broader work on influence estimation and structural role detection, especially in online and social networks [17].

A natural way to address this limitation is through information-theoretic analysis. Since Shannon's foundational work [15], entropy-based ideas have increasingly been adapted to graphs and networks. General graph-entropy frameworks were introduced to quantify structural complexity [6], and later overviews established entropy as a useful perspective for studying structural information in networks [7]. More recently, entropy-based methods have been applied directly to social-network analysis. The Structural Entropy Index of Community (SEIC) was proposed for Twitter communication analysis [3], community-based entropic methods were developed for identifying influential nodes across multiple social networks [16], and structure-entropy formulations were shown to capture macro-level aspects of social-network organization and stability [14].

The present work is most directly connected to recent k -hop-based entropy formulations. In earlier work, hybrid entropy-based metrics for k -hop environment analysis, including Entropy-Weighted Redundancy (EWR) and Normalized Entropy Density (NED), were introduced to characterize the structural sensitivity of node neighborhoods in complex networks [1, 2]. Unlike classical centrality measures, these metrics characterize how the local environment of a node is organized within a broader structural radius by combining informational diversity and redundancy. This makes them suitable for investigating whether structurally dis-

tinctive nodes can be emphasized even when they are not highly ranked by degree or eigenvector centrality.

Although the present study focuses on Twitter-derived social networks, similar ideas may also be investigated in educational and learning-related settings in future work [8, 11, 12].

In summary, the present work builds on three main lines of research: classical centrality theory, entropy-based graph and social-network analysis, and recent k -hop-based entropy formulations [1–3, 5–7, 9, 10, 16]. Its specific contribution is an empirical comparison of classical centrality measures and entropy-based k -hop metrics on multiple real-world Twitter-derived networks in order to investigate whether these approaches capture different aspects of structural node importance.

3. Preliminaries and Metrics

Let $G = (V, E)$ be a simple undirected graph, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. For $v \in V$, let $d(v)$ denote the degree of v . For a fixed integer $k \geq 1$, the k -hop neighborhood of v is

$$N_k(v) = \{u \in V \setminus \{v\} \mid \text{dist}(v, u) \leq k\},$$

and the induced subgraph on $N_k(v)$ is denoted by $G[N_k(v)]$. We write

$$n_k(v) = |N_k(v)|, \quad m_k(v) = |E(G[N_k(v)])|.$$

3.1. Classical Baseline Measures

As baseline measures, we consider raw degree and eigenvector centrality. The degree-based score of node v is

$$C_D(v) = d(v).$$

Let A be the adjacency matrix of G . The eigenvector centrality score $C_E(v)$ is defined by the dominant eigenvector $\mathbf{x}$ of A ,

$$A\mathbf{x} = \lambda_{\max}\mathbf{x},$$

where $\lambda_{\max}$ is the largest eigenvalue of A , and $C_E(v)$ is the component of $\mathbf{x}$ corresponding to v .

3.2. Entropy in the k -Hop Environment

Following the local-degree variant of the k -hop entropy framework, the entropy-based metrics are computed from the degree distribution inside the induced subgraph $G[N_k(v)]$. For each $u \in N_k(v)$, let $d_{N_k(v)}(u)$ denote the degree of u within $G[N_k(v)]$. If

$$\sum_{u \in N_k(v)} d_{N_k(v)}(u) > 0,$$

we define

$$p_u(v) = \frac{d_{N_k(v)}(u)}{\sum_{w \in N_k(v)} d_{N_k(v)}(w)}, \quad u \in N_k(v),$$

and the Shannon entropy of the k -hop environment of v by

$$H_k(v) = - \sum_{u \in N_k(v)} p_u(v) \log_2 p_u(v).$$

If $N_k(v) = \emptyset$ or $G[N_k(v)]$ has no edges, then $H_k(v) = 0$.

3.3. k -Hop Redundancy / Density

The redundancy component is defined as the edge density of the induced k -hop neighborhood:

$$R_k(v) = \begin{cases} \frac{m_k(v)}{\binom{n_k(v)}{2}}, & \text{if } n_k(v) \geq 2, \\ 0, & \text{if } n_k(v) < 2. \end{cases}$$

3.4. Entropy-Weighted Redundancy (EWR)

The Entropy-Weighted Redundancy of node v at radius k is

$$\text{EWR}_k(v) = H_k(v) R_k(v).$$

3.5. Normalized Entropy Density (NED)

The Normalized Entropy Density of node v at radius k is

$$\text{NED}_k(v) = \begin{cases} \frac{H_k(v)}{n_k(v)} R_k(v), & \text{if } n_k(v) > 0, \\ 0, & \text{if } n_k(v) = 0. \end{cases}$$

Thus, $\text{NED}_k(v)$ normalizes the entropy-redundancy score by the size of the induced k -hop neighborhood.

3.6. Datasets

The empirical evaluation was conducted on three Twitter-derived networks with different sizes and connectivity patterns. The Zandberg Twitter network represents a relatively small political communication network related to the Polish electoral context. In contrast, the Foursquare-based Twitter network and the Twitter network are substantially larger anonymized social networks obtained from the publicly available *Foursquare and Twitter Networks* dataset available on Kaggle.

The Foursquare-based network contains 44,832 nodes and 1,664,402 edges, whereas the Twitter network contains 48,089 nodes and 16,303,641 edges.

In all datasets, nodes correspond to user accounts and edges represent social relationships between users. Following common practice in structural network analysis, edge directions were ignored and the networks were converted into simple undirected graphs. This representation enables direct comparison between connectivity-based centrality measures and entropy-based k -hop metrics without introducing additional asymmetries associated with directed interactions.

The Zandberg Twitter network is not fully connected; its largest connected component contains 463 nodes out of the total 609 nodes. In contrast, the Foursquare-based Twitter network and the Twitter network are connected. The analysis was performed on the complete graphs and was not restricted to the largest connected component.

The three networks exhibit markedly different structural characteristics. The Zandberg network is sparse and relatively small, allowing qualitative interpretation of several publicly identifiable actors. The Foursquare-based network exhibits a highly heterogeneous connectivity pattern, whereas the Twitter network is considerably denser and provides a contrasting environment in which the behavior of entropy-based measures can be examined under different connectivity regimes. These differences make the three datasets suitable for investigating whether entropy-based rankings emphasize structural characteristics that differ from those highlighted by degree and eigenvector centrality.

4. Analysis

This section presents the empirical analysis of the three Twitter-derived datasets. For each network, we compare the rankings obtained from degree centrality, eigenvector centrality, EWR, and NED for $k = 2$ and $k = 3$, with the aim of investigating whether entropy-based measures emphasize structural characteristics differently from classical centrality measures.

4.1. Zandberg Twitter Network

We first consider the *Zandberg Twitter network*, the most interpretable dataset in this study because several accounts are publicly identifiable. The network is related to the Polish political landscape in an election-related context and contains 609 nodes and 843 edges; its largest connected component has 463 nodes, so the graph is not fully connected. Both degree and eigenvector centrality identify @ZandbergRAZEM and @__Lewica as the most prominent actors. By contrast, EWR and NED place @pawel_baka among the highest-ranked nodes despite its lower visibility under degree and eigenvector centrality. Interestingly, this account belonged to the spokesperson of the Razem party. Although the present analysis does not validate functional influence, this observation illustrates that entropy-based rankings may emphasize actors whose structural characteristics differ from

those highlighted by connectivity-based measures. The relatively small differences between the $k = 2$ and $k = 3$ rankings suggest that the entropy-based rankings are fairly stable with respect to the neighborhood radius.

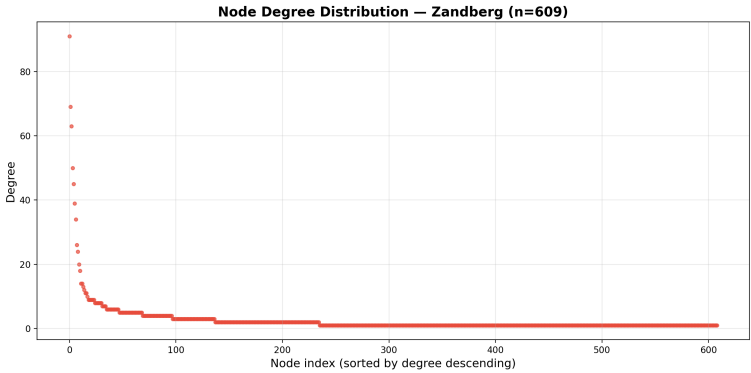


Figure 1. Sorted degree sequence of the Zandberg Twitter network.

Figure 1 shows the sorted degree sequence of the Zandberg Twitter network. Only a few nodes occupy hub-like positions, whereas the majority of nodes have low degree, indicating a heterogeneous interaction structure.

Tables 1–3 summarize the top-10 rankings of the Zandberg Twitter network under the classical and entropy-based measures. Table 1 shows that @ZandbergRAZEM and @Lewica are the most prominent actors according to degree and eigenvector centrality. By contrast, Tables 2 and 3 place @pawel_baka at or near the top of the entropy-based rankings. The strong overlap between Tables 2 and 3 indicates that the entropy-based rankings are relatively stable with respect to the neighborhood radius in this network.

Table 1. Top 10 nodes by classical measures in the Zandberg Twitter network.

Rank	Degree		Eigenvector	
	Node	Value	Node	Value
1	@ZandbergRAZEM	91	@Lewica	1.0000
2	@PolskieRadio24	69	@PiotrNo66983662	0.7954
3	@CzerwonyKat	63	@ZandbergRAZEM	0.7630
4	@Lewica	50	@rorek	0.7513
5	@Amigant	45	@CzerwonyKat	0.5009
6	@rorek	39	@PolskieRadio24	0.4939
7	@partiarazem	34	@wlodekczarzasty	0.4132
8	@PiotrNo66983662	26	@Amigant	0.2780
9	@wlodekczarzasty	24	@AMZukowska	0.2233
10	@marianapiorkows	20	@GiertychRoman	0.1357

Table 2. Top 10 nodes by entropy-based measures ($k = 2$) in the Zandberg Twitter network.

Rank	EWR ($k = 2$)		NED ($k = 2$)	
	Node	Value	Node	Value
1	@pawel_baka	1.5850	@pawel_baka	0.5283
2	@maggiepython	1.3333	@BuurtBelangen	0.5000
3	@SurowKarol	1.2027	@kasiababis	0.5000
4	@Lewica_News	1.1232	@BNDeStem	0.5000
5	@DemSocjalizm	1.0863	@mkljczk	0.5000
6	@adamidmark	1.0610	@Bartek19953	0.5000
7	@BuurtBelangen	1.0000	@21lupus	0.5000
8	@damian_krawczuk	1.0000	@_surdackiH	0.5000
9	@boch_adam	1.0000	@RazemInt	0.5000
10	@jakubkaldunski	1.0000	@BatugWojciech	0.5000

Table 3. Top 10 nodes by entropy-based measures ($k = 3$) in the Zandberg Twitter network.

Rank	EWR ($k = 3$)		NED ($k = 3$)	
	Node	Value	Node	Value
1	@pawel_baka	1.5850	@pawel_baka	0.5283
2	@jakubkaldunski	1.0000	@mkljczk	0.5000
3	@Europiara	1.0000	@Bartek19953	0.5000
4	@SWasilkowski	1.0000	@Strykasia	0.5000
5	@BNDeStem	1.0000	@JablonskiDoktor	0.5000
6	@mkljczk	1.0000	@s_urbanowicz	0.5000
7	@Bartek19953	1.0000	@wziolek_sj	0.5000
8	@RazemInt	1.0000	@1972tomek	0.5000
9	@Strykasia	1.0000	@jzpinski	0.5000
10	@JablonskiDoktor	1.0000	@CDA_Breda	0.5000

4.2. Foursquare

The second dataset is the *Foursquare-based Twitter dataset*. Degree and eigenvector centrality show strong overlap, indicating a relatively consistent notion of prominence. In contrast, the entropy-based rankings differ both from the classical rankings and from one another. EWR ($k = 2$) is notably distinct from NED ($k = 2$) and from the $k = 3$ rankings, whereas NED ($k = 2$), EWR ($k = 3$), and NED ($k = 3$) show substantial overlap. In particular, @Uma appears prominently in the entropy-based rankings but not in the classical ones, while @Tom exhibits the opposite behavior. The presence of @Kevin only in EWR ($k = 2$) further illustrates that different metrics and neighborhood radii may emphasize different structural characteristics.

Figure 2 shows the sorted degree sequence of the Foursquare-based Twitter dataset. The sequence is highly skewed, with a small number of highly connected

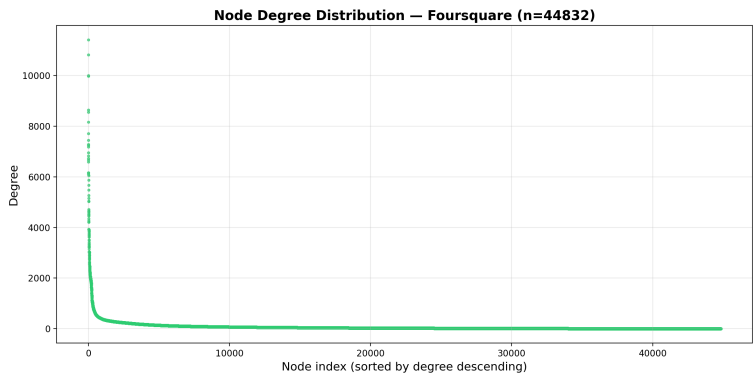


Figure 2. Sorted degree sequence of the Foursquare-based Twitter dataset.

nodes and a long tail of low-degree nodes. This indicates a heterogeneous network structure in which connectivity is concentrated in relatively few actors. Such a pattern provides a useful setting for comparing classical and entropy-based rankings.

Tables 4–6 summarize the top-10 rankings of the Foursquare-based Twitter dataset under the classical and entropy-based measures. The classical rankings in Table 4 exhibit substantial overlap, indicating a relatively stable notion of prominence. In contrast, the entropy-based rankings in Tables 5 and 6 differ considerably from the classical rankings and partly from one another. A notable example is @Uma, which appears prominently in the entropy-based rankings but not in the classical ones, whereas @Tom is highly ranked by degree and eigenvector centrality but absent from the entropy-based top lists. Overall, these results indicate that entropy-based rankings may emphasize structural characteristics that are not highlighted by degree and eigenvector centrality.

Table 4. Top 10 nodes by classical measures in the Foursquare-based Twitter dataset.

Rank	Degree			Eigenvector Centrality		
	Node	ID	Value	Node	ID	Value
1	@Tom	4105	11415	@Tom	4105	1.0000
2	@Alice	4297	10817	@Charlie	4326	0.9731
3	@Bob	4231	10007	@Alice	4297	0.9621
4	@Charlie	4326	9975	@Bob	4231	0.9614
5	@David	48	8639	@David	48	0.9384
6	@Emma	4289	8553	@Frank	4142	0.9297
7	@Frank	4142	8169	@Emma	4289	0.9257
8	@Grace	39	7712	@Grace	39	0.9204
9	@Hannah	4299	7449	@Ivan	4208	0.9005
10	@Ivan	4208	7287	@Jack	4146	0.8899

Table 5. Top 10 nodes by entropy-based measures ($k = 2$) in the Foursquare-based Twitter dataset.

Rank	EWR ($k = 2$)			NED ($k = 2$)		
	Node	ID	Value	Node	ID	Value
1	@Kevin	31106	2.3072	@Uma	40102	0.5000
2	@Laura	44011	2.2980	@Victor	43638	0.5000
3	@Mike	42036	2.0403	@Wendy	42264	0.3333
4	@Nina	21647	1.8982	@Xavier	38481	0.3333
5	@Oscar	36293	1.8683	@Yara	24008	0.3333
6	@Paul	21648	1.7988	@Zane	41146	0.3176
7	@Quinn	39322	1.7869	@Adam	32205	0.2241
8	@Rachel	44064	1.7869	@Bella	36512	0.2241
9	@Steve	38127	1.7184	@Chris	43078	0.2241
10	@Tina	42150	1.6820	@Diana	33438	0.2241

Table 6. Top 10 nodes by entropy-based measures ($k = 3$) in the Foursquare-based Twitter dataset.

Rank	EWR ($k = 3$)			NED ($k = 3$)		
	Node	ID	Value	Node	ID	Value
1	@Uma	40102	0.9756	@Victor	43638	0.0181
2	@Xavier	38481	0.8316	@Uma	40102	0.0127
3	@Ethan	26933	0.8019	@Xavier	38481	0.0121
4	@Yara	24008	0.6838	@Oliver	41184	0.0079
5	@Noah	14627	0.5650	@Liam	41186	0.0079
6	@Victor	43638	0.5618	@Lucas	26935	0.0070
7	@Oliver	41184	0.5509	@Noah	14627	0.0061
8	@Liam	41186	0.5509	@Ethan	26933	0.0060
9	@Lucas	26935	0.5204	@Wendy	42264	0.0039
10	@Wendy	42264	0.4938	@Yara	24008	0.0038

4.3. Twitter

The third dataset is the *Twitter dataset*. In this considerably denser network, degree and eigenvector centrality exhibit relatively little overlap in their top-10 rankings, suggesting that these measures emphasize different aspects of node prominence. Their highest-ranked nodes are also absent from the entropy-based rankings, indicating that the entropy-based measures emphasize different structural characteristics. At the same time, EWR ($k = 2$) and NED ($k = 2$) are highly similar, whereas the $k = 3$ rankings introduce partly different nodes. In particular, nodes such as @Luke and @Caleb appear in EWR ($k = 3$), while NED ($k = 3$) retains nodes already highlighted at $k = 2$, including @Lucas and @Matthew. This illustrates that the rankings produced by EWR and NED are influenced by the choice of neighborhood radius.

Figure 3 shows the sorted degree sequence of the Twitter dataset.

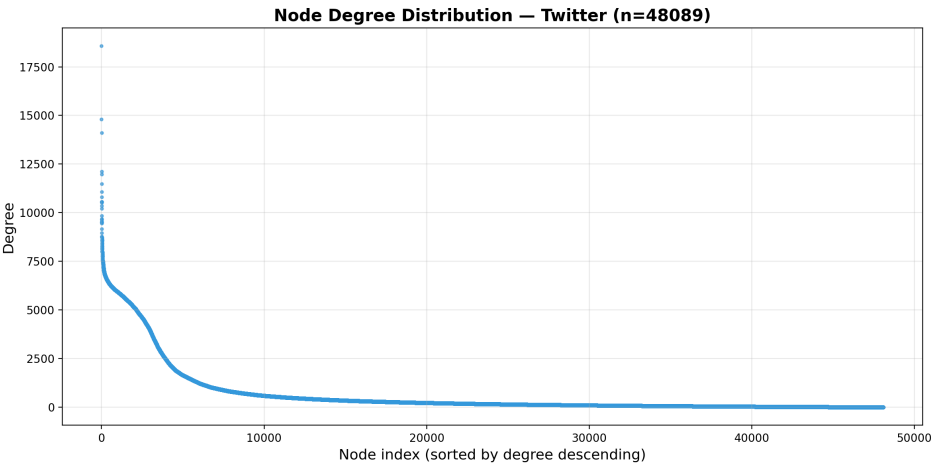


Figure 3. Sorted degree sequence of the Twitter dataset.

Tables 7–9 summarize the top-10 rankings of the Twitter dataset under the classical and entropy-based measures. The classical rankings in Table 7 show relatively little overlap, indicating that degree and eigenvector centrality emphasize different aspects of node prominence in this dense network. By contrast, Table 8 shows strong agreement between EWR and NED for $k = 2$, whereas Table 9 introduces partly different nodes at $k = 3$. In particular, nodes such as @Luke and @Caleb appear in EWR ($k = 3$), while NED ($k = 3$) retains nodes already highlighted at lower radius, including @Lucas and @Matthew. Taken together, these results indicate that entropy-based rankings may emphasize structural characteristics that are not highlighted by degree and eigenvector centrality.

Table 7. Top 10 nodes by classical measures in the Twitter dataset.

Rank	Degree			Eigenvector Centrality		
	Name	ID	Value	Name	ID	Value
1	@Oliver	4639	18577	@James	7812	1.0000
2	@Jack	4374	14804	@Henry	4367	0.9914
3	@Harry	4368	14105	@Leo	7383	0.9815
4	@George	4741	12110	@Alfie	7123	0.9800
5	@Noah	268	11968	@Joshua	7994	0.9798
6	@Charlie	4919	11474	@Freddie	10998	0.9786
7	@Jacob	5427	11079	@Archie	11468	0.9770
8	@Thomas	4303	10817	@Theo	12062	0.9756
9	@Oscar	2091	10567	@Isaac	11947	0.9756
10	@William	9155	10565	@Alexander	7873	0.9744

Table 8. Top 10 nodes by entropy-based measures ($k = 2$) in the Twitter dataset.

Rank	EWR ($k = 2$)			NED ($k = 2$)		
	Name	ID	Value	Name	ID	Value
1	@Lucas	41893	1.0000	@Lucas	41893	0.5000
2	@Mason	46367	1.0000	@Mason	46367	0.5000
3	@Ethan	33250	1.0000	@Ethan	33250	0.5000
4	@Logan	38746	0.8962	@Logan	38746	0.2241
5	@Daniel	45437	0.8962	@Daniel	45437	0.2241
6	@Matthew	29604	0.8962	@Matthew	29604	0.2241
7	@Aiden	43465	0.8155	@Samuel	26985	0.1600
8	@Joseph	43466	0.8155	@David	39059	0.1600
9	@Samuel	26985	0.8000	@Aiden	43465	0.1359
10	@David	39059	0.8000	@Joseph	43466	0.1359

Table 9. Top 10 nodes by entropy-based measures ($k = 3$) in the Twitter dataset.

Rank	EWR ($k = 3$)			NED ($k = 3$)		
	Name	ID	Value	Name	ID	Value
1	@Carter	10347	0.2324	@Lucas	41893	0.000124
2	@Owen	6358	0.2204	@Matthew	29604	0.000062
3	@Dylan	6635	0.2150	@Ryan	45838	0.000011
4	@Luke	42707	0.1979	@Caleb	7171	0.000011
5	@Gabriel	7155	0.1966	@Nathan	45412	0.000010
6	@Anthony	7596	0.1964	@Carter	10347	0.000009
7	@IsaacJr	12018	0.1863	@Adrian	31762	0.000008
8	@Grayson	21728	0.1794	@Christian	10149	0.000008
9	@Wyatt	11173	0.1778	@Aaron	11911	0.000008
10	@Caleb	7171	0.1764	@Luke	42707	0.000007

5. Conclusion and Future Work

This paper investigated whether entropy-based k -hop metrics emphasize structural characteristics that are not highlighted by classical centrality measures. Using three Twitter-derived networks with different sizes and connectivity patterns, we compared degree centrality and eigenvector centrality with two entropy-based measures, Entropy-Weighted Redundancy (EWR) and Normalized Entropy Density (NED). Across all analyzed datasets, the results showed that entropy-based rankings often differ substantially from classical rankings and may emphasize structural characteristics that are not highlighted by degree or eigenvector centrality alone.

The empirical analysis further showed that the behavior of EWR and NED depends on both the underlying network topology and the neighborhood radius. In the Zandberg network, the entropy-based rankings were relatively stable across $k =$

2 and $k = 3$, whereas in the larger datasets the differences between configurations were more pronounced.

Overall, the results indicate that entropy-based rankings provide a complementary perspective to connectivity-based measures, particularly when different structural characteristics of node neighborhoods are of interest. Future studies should extend the empirical analysis to other network classes and additional baseline measures, investigate the influence of the neighborhood radius for a wider range of k values, and perform quantitative comparisons using Spearman correlation, Kendall's τ , top- k overlap, and Jaccard similarity. These investigations may contribute to a more comprehensive characterization of structural node importance in complex networks.

References

- [1] C. BIRÓ: *Hybrid Entropy-Based Metrics for k -Hop Environment Analysis in Complex Networks*, Mathematics 13.17 (2025), p. 2902, DOI: [10.3390/math13172902](https://doi.org/10.3390/math13172902).
- [2] C. BIRÓ: *Structural Sensitivity in Graphs: An Entropy-Based k -Hop Metric and its Applications*, in: 2025 IEEE 19th International Symposium on Applied Computational Intelligence and Informatics (SACI), IEEE, 2025, pp. 273–278, DOI: [10.1109/SACI66288.2025.11030155](https://doi.org/10.1109/SACI66288.2025.11030155).
- [3] W. BŁOCKI, M. SZEWCZYK, A. ADAMSKI: *Quantifying Information Distribution in Social Networks: The Structural Entropy Index of Community (SEIC) for Twitter Communication Analysis*, Entropy 27.11 (2025), p. 1140, DOI: [10.3390/e27111140](https://doi.org/10.3390/e27111140).
- [4] P. BONACICH: *Factoring and Weighting Approaches to Status Scores and Clique Identification*, Journal of Mathematical Sociology 2.1 (1972), pp. 113–120, DOI: [10.1080/0022250X.1972.9989806](https://doi.org/10.1080/0022250X.1972.9989806).
- [5] P. BONACICH: *Power and Centrality: A Family of Measures*, American Journal of Sociology 92.5 (1987), pp. 1170–1182, DOI: [10.1086/228631](https://doi.org/10.1086/228631).
- [6] M. DEHMER: *Information Processing in Complex Networks: Graph Entropy and Information Functionals*, Applied Mathematics and Computation 201.1–2 (2008), pp. 82–94, DOI: [10.1016/j.amc.2007.12.010](https://doi.org/10.1016/j.amc.2007.12.010).
- [7] M. DEHMER, F. EMMERT-STREIB, Y. SHI: *A History of Graph Entropy Measures*, Information Sciences 181.1 (2011), pp. 57–78, DOI: [10.1016/j.ins.2010.08.041](https://doi.org/10.1016/j.ins.2010.08.041).
- [8] M. DUNLAP: *What all higher education students should know about online social networking*, The Internet and Higher Education 11.1 (2008), pp. 60–62, ISSN: 1096-7516, DOI: [10.1016/j.iheduc.2007.12.001](https://doi.org/10.1016/j.iheduc.2007.12.001).
- [9] L. C. FREEMAN: *A Set of Measures of Centrality Based on Betweenness*, Sociometry 40.1 (1977), pp. 35–41, DOI: [10.2307/3033543](https://doi.org/10.2307/3033543).
- [10] L. C. FREEMAN: *Centrality in Social Networks: Conceptual Clarification*, Social Networks 1.3 (1978), pp. 215–239, DOI: [10.1016/0378-8733\(78\)90021-7](https://doi.org/10.1016/0378-8733(78)90021-7).
- [11] D. GAŠEVIĆ, S. DAWSON, G. SIEMENS: *Let's not forget: Learning analytics are about learning*, TechTrends 59.1 (2015), pp. 64–71, DOI: [10.1007/s11528-014-0822-x](https://doi.org/10.1007/s11528-014-0822-x).
- [12] J. G. S. GOLDIE: *Connectivism: A knowledge learning theory for the digital age?*, Medical Teacher 38.10 (2016), pp. 1064–1069, DOI: [10.3109/0142159X.2016.1173661](https://doi.org/10.3109/0142159X.2016.1173661).
- [13] J. M. KLEINBERG: *Authoritative Sources in a Hyperlinked Environment*, Journal of the ACM 46.5 (1999), pp. 604–632, DOI: [10.1145/324133.324140](https://doi.org/10.1145/324133.324140).

- [14] Z. LI, Z. YAN, J. YANG, X. TANG: *The Structure Entropy of Social Networks*, Journal of Systems Science and Complexity 37 (2024), pp. 1147–1162, DOI: [10.1007/s11424-024-2484-x](https://doi.org/10.1007/s11424-024-2484-x).
- [15] C. E. SHANNON: *A Mathematical Theory of Communication*, Bell System Technical Journal 27.3 (1948), pp. 379–423, DOI: [10.1002/j.1538-7305.1948.tb01338.x](https://doi.org/10.1002/j.1538-7305.1948.tb01338.x).
- [16] N. VAF AEI, F. SHEIKHI, A. GHASEMI: *A community-based entropic method to identify influential nodes across multiple social networks*, Social Network Analysis and Mining 15.1 (2025), p. 23, DOI: [10.1007/s13278-025-01441-y](https://doi.org/10.1007/s13278-025-01441-y).
- [17] Y.-B. ZHUANG, Z.-H. LI, Y.-J. ZHUANG: *Identification of influencers in online social networks: measuring influence considering multidimensional factors exploration*, Heliyon 7.4 (2021), e06472, DOI: [10.1016/j.heliyon.2021.e06472](https://doi.org/10.1016/j.heliyon.2021.e06472).



Local Scientific Collaboration Networks: A Case Study

Imre Varga*

Department of IT Systems and Networks,
Faculty of Informatics,
University of Debrecen
varga.imre@inf.unideb.hu

Abstract. This study analyzes local scientific collaboration through weighted co-authorship networks of two Hungarian university faculties using the domestic scientific bibliography (MTMT) data. Researchers are modeled as nodes and joint publications define weighted edges. Network metrics and a permutation-based statistical framework are applied to compare structures. While some global properties are similar, significant differences appear in degree and strength distributions, indicating distinct collaboration patterns independent of faculty size. The results highlight structural features that cannot be captured without a network-based approach. These insights are valuable for institutional strategy and decision-making in fostering collaboration.

Keywords: co-authorship, weighted network, MTMT, scientific collaboration

2020 Mathematics Subject Classification: Primary 05C82, 91D30, 05C90; Secondary 90C35

1. Introduction

Scientific collaboration networks have become a central object of study for understanding how knowledge is produced, disseminated, and validated within the scientific system. By modeling researchers and their joint publications as evolving graphs, publication networks make it possible to quantify patterns of cooperation, specialization, and interdisciplinary exchange at multiple scales. These networks

*The author thanks Bálint Kósa for the feasibility study.

capture not only who collaborates with whom, but also how scientific communities emerge, fragment, and reorganize over time in response to funding structures, institutional incentives, and emerging research topics [9, 10].

State-of-the-art research in this area combines large-scale bibliographic data with methods from network science, scientometrics, and computational social science. Recent advances include the identification of core-periphery structures, the detection of research communities and topical clusters, and the modeling of collaboration dynamics using temporal and multilayer networks. In particular, publication networks are increasingly used to study productivity, impact, and inequality in science, as well as to evaluate how collaborative structures influence innovation and the diffusion of ideas across disciplinary boundaries [5, 7].

Related works differ in many ways. They use different data sources e.g., Scopus [3], Web of Science [1], MedPub [10], arXiv [6], Google Scholar [4]. In these works, several problems are solved in different ways. For example, in some data sources, the authors cannot be identified unambiguously due to the different spellings or abbreviations of their names. Sometimes, each differently spelled name appearing in publications is considered a unique author, or authors with the same initials are considered identical authors [10]. We may face a similar problem with affiliations, if they are even specified in the databases. Some databases are not validated (e.g., Google Scholar), others cover only a narrow set of indexed journals (e.g., Scopus or WoS), but not all the publications of a given author.

In this work, we investigate the collaboration of faculty members of Hungarian universities using co-authorship networks of local scientific communities. Our goal is to quantify, understand, and visualize inter-personal and inter-department scientific collaborations through publications affiliated with the chosen universities. Earlier work has shown that node characterization based on the local graph environment may reveal structural properties that are not captured by purely global network measures [2].

2. Methodology

2.1. Data source

The present study builds on the publication data from the Hungarian Scientific Bibliography called MTMT [8], which provides a uniquely suitable data source for the analysis of scientific collaboration networks within the Hungarian research ecosystem. It is a national bibliography database for authentic recording and presentation of the results of domestic scientific research. MTMT records all scientific works, not just indexed articles. It has multiple levels of data validation: author upload and institutional review. Its use is mandatory, in a sense, so the majority of the Hungarian scientists have up-to-date MTMT accounts.

The MTMT MyCite2 API provides programmatic access to the database, enabling large-scale, automated retrieval of structured bibliographic and scientometric data. Through its RESTful interface, users can query publication records,

author profiles, and citation links. The API facilitates reproducible data collection workflows by allowing parameterized queries, filtering by document types or publication years. Importantly, the standardized and curated nature of MTMT data, combined with the API's machine-readable output formats (e.g., JSON), significantly reduces the need for post hoc data cleaning and disambiguation.

This work focuses on the two organizing faculties of the *13th International Conference on Applied Informatics (ICAI2026)*, namely:

- Faculty of Informatics, University of Debrecen (UD),
- Faculty of Informatics, Eszterházy Károly Catholic University (EKCU).

They are faculties of prestigious eastern Hungarian universities. At the time of the study (December 2025), the MTMT profile identifiers of full-time lecturers/researchers at the above universities were collected and the publication records of these authors were downloaded and filtered by an in-house developed *Python* program.

The study only considered those publication types that represent significant scientific work and extensive publications:

- Book (e.g., Monography, Conference volume),
- Book portion (e.g., Book chapter, Essay),
- Journal article (e.g., Short, Summary, Multi-authored),
- Conference paper (e.g., proceedings paper).

Shorter and simpler publications (conference abstracts, reviews, critiques, theses, repertoires, etc.) were ignored.

The data gathering, the cleaning, the network formation, and its analysis are implemented by self-developed *Python* programs using the *NumPy* package. The visualization of different co-authorship networks is made using the *Cytoscape 3.10.1* software.

2.2. Network creation

From the downloaded publication records, the author IDs are extracted for each publication. These authors' lists of publications served as the basis of our local network definition. A $G = (V, E, W)$ weighted graph is defined for each organization, where a $v_i \in V$ node represents an active full-time researcher of the given faculty at the end of 2025. Nodes v_i and v_j are connected by an edge $e_{ij} \in E$, when there is at least one publication in the filtered data set co-authored by both v_i and v_j researchers. The scientific collaboration of two authors during their full career is characterized by the number of joint publications, which is represented by the $w_{ij} \in W$ weight of the e_{ij} edge. It is important to highlight that the graphs contain only the current employees (no retired, deceased or terminated employees),

but all of their publications are considered, even if the author used to have a different affiliation earlier. The k_i degree of node i is the number of linked nodes. The s_i strength of node i is determined as the sum of w_{ij} weights of the connected edges, so $s_i = \sum_j w_{ij}$. This definition means that both the large average number of co-authors and the total number of co-authored publications increase this s_i collaboration strength of the author. The cardinality of the nodes is denoted by $N = |V|$, the cardinality of edges is denoted by $L = |E|$, and all w_{ij} weights are positive integers. The G graph is undirected (thus $w_{ij} = w_{ji}$) and loopless (so $e_{ii} \notin E$).

In some analyses, the departments of the authors are also considered, defining disjoint subsets of nodes, denoted as $V^d \subset V$. In this case, the set of edges can be divided into two disjoint subsets $E = E^{inter} \cup E^{intra}$, where E^{inter} means inter-department edges linking authors of different departments and E^{intra} means intra-department edges connecting authors from identical department.

2.3. Statistical network comparison

In $G_1 = (V_1, E_1, W_1)$ and $G_2 = (V_2, E_2, W_2)$ weighted networks of different sizes and topologies, we determine whether the distribution of a network quantity is truly different, or whether the difference can be explained purely by the network structure and sampling. Let $\gamma_i(v)$ denote a node metric for $v \in V_i$ node (or an edge feature for $e \in E_i$ edge). The observed differences between networks can be defined by the $\langle \gamma_i(v) \rangle$ averages of the given characteristics within the graphs as

$$\Delta_{obs} = \langle \gamma_1(v) \rangle - \langle \gamma_2(v) \rangle = \frac{1}{|V_1|} \sum_{v \in V_1} \gamma_1(v) - \frac{1}{|V_2|} \sum_{v \in V_2} \gamma_2(v).$$

Null hypothesis: the values of the given characteristics can be exchanged within the network so that the topology is preserved and the two networks come from the same generating mechanism. Our perturbation null model has the following steps:

1. We fix the structure of the G_1 and G_2 graphs.
2. We collect all the node (edge) characteristics:

$$\Gamma = \{\gamma_1(v) : v \in V_1\} \cup \{\gamma_2(v) : v \in V_2\}.$$

3. We randomly redistribute these values of characteristics among the nodes (edges), so that G_1 and G_2 each receive the same number of values as they originally had.

This is a conditional random sample, and we calculate this for each permutation: $\Delta^{(k)} = \langle \gamma_1^{(k)} \rangle - \langle \gamma_2^{(k)} \rangle$. This gives the null distribution: $\Delta^{null} = \{\Delta^{(1)}, \dots, \Delta^{(M)}\}$. The statistical p -value can be defined as $p = P(|\Delta^{null}| \geq |\Delta_{obs}|)$ or empirically

$$p \approx \frac{1}{M} \sum_{k=1}^M 1(|\Delta^{(k)}| \geq |\Delta_{obs}|).$$

If $p < 0.05$, then the difference cannot be explained solely by the structure of the network and the random characteristics distribution.

3. Results

First, a broad scope network analysis was performed, where not just the faculty members are considered, but all the co-authors of the faculty members, so the node set is extended. The network of the Faculty of Informatics, University of Debrecen (hereafter UD) contains 855 nodes (including the 84 faculty members) and 4605 links connecting them based on 5184 considered publications. The network of the Faculty of Informatics, Eszterházy Károly Catholic University (hereafter EKCUC) contains 372 nodes (including the 40 faculty members) and 1065 links connecting them based on 1807 considered publications. However, the network of UD is much larger, network analysis shows that the $\langle k \rangle$ average degree of nodes representing the average number of co-authors of a researcher is not so different, for UD $\langle k \rangle = 22$, for EKCUC $\langle k \rangle = 18$. It shows that the personal connections of the researchers are not purely determined by the size of an organization. Naturally, the degree distribution is not uniform; there are hubs in the networks. An interesting finding emerges from this study: the deans of both faculties have the most co-authors, despite the fact that their academic careers are far from the longest.

The remaining part of the investigation focuses on only the local co-authorship networks, covering only the researchers of the given faculty. This restriction causes smaller networks and easier visualization. The network of UD consists of $N = 84$ nodes and $L = 270$ links, as it is illustrated in Figure 1a, while the network of EKCUC has $N = 40$ nodes and $L = 88$ links, see Figure 1b.

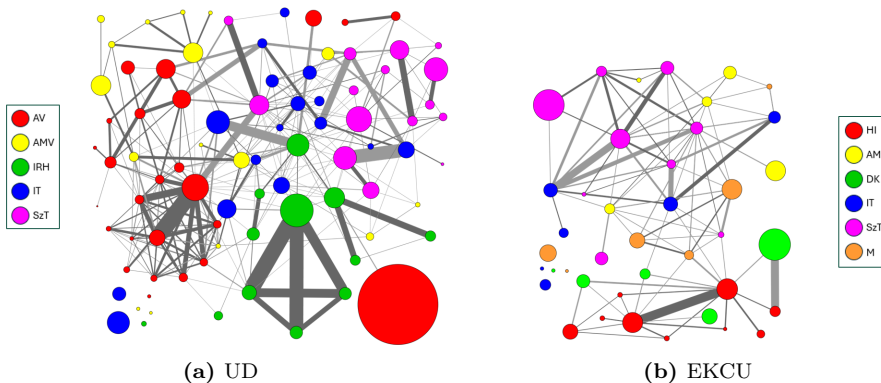


Figure 1. Local co-authorship network of faculties. The area of nodes is proportional to the number of publications of the given author, and their colors represent the departments. The width of links is proportional to the edge weight, and different shades represent the intra- and inter-department connections.

As Figure 1 shows, there are tiny nodes of mainly young researchers with only a few publications, but there are big nodes as well, illustrating a large number of publications of productive researchers. An exceptionally large node is also visible in the UD network, representing a researcher frequently participating in nuclear physics projects (however, the faculty is related to computer science), where a large number of papers with many authors are common. (This author has 1822 publications, nevertheless the average number of considered publications for other UD scientists is just 51.) Wide links between two nodes show a lot of joint publications, illustrating strong collaborations. The maximum belongs to an author pair having 52 jointly written papers.

These local networks are not connected, since the majority of the nodes form a giant component, but there are isolated nodes without connections to others. It means that some of the researchers have never published together with others from the given faculty. The size of the isolated node is not always small, so we cannot assume these represent young scientists. One of the isolated nodes at UD has more than 140 publications affiliated with the given organization over decades, but none of them is co-authored by other faculty members, because this researcher has completely different research topics than others. An experienced researcher of the EKC has 33 publications, but has not got links to other nodes, because he was recently hired on the faculty. The ratio of the giant component in both networks is close to 90%, however, the size of UD network is twice that of the EKC.

Further network characteristics are also calculated for both local collaboration networks in order to highlight differences or similarities between these graphs [12]. The features are summarized in the Table 1.

Table 1. The main network characteristics for the studied co-authorship networks.

Network metrics	UD	EKC
Number of Components	7	6
Giant Component	92.9%	87.5%
Characteristic Path Length	2.658	2.582
Clustering Coefficient	0.419	0.513
Density	0.078	0.113
Average Degree	6.923	5.029
Minimum Degree	0	0
Maximum Degree	30	12
Network Diameter	6	5
Network Radius	3	3
Heterogeneity	0.725	0.696
Centralization	0.308	0.217

The average degree, the density, the centralization, or the average clustering coefficient are very different for these organizational networks. At the same time, the characteristic path length, the network radius, or the heterogeneity is almost the

same. As one can see, from a given aspect, the networks are similar, but from other aspects, they are quite different. In order to determine whether these differences are caused by only the different size of the networks or these are structural differences, a statistical comparison of these weighted networks is needed.

The statistical comparison of weighted networks requires methods that account for the inherent dependence structure of relational data. To evaluate whether two networks differ in their weight- or structure-related properties, we adopt a permutation-based framework [11] that avoids the independence assumptions of classical parametric tests. Let G_1 and G_2 denote two weighted networks, and let $T(G)$ represent a network-level statistic of interest, such as the mean edge weight, mean node degree, or mean node strength. The observed difference between the networks is defined as $\Delta_{obs} = T(G_1) - T(G_2)$. To assess statistical significance, we construct an empirical null distribution by pooling the relevant values (e.g., edge weights or node-level measures) from both networks and repeatedly permuting them while preserving the original sample sizes. For each permutation, the statistical difference is recomputed, yielding a distribution of differences expected under the null hypothesis that both networks arise from the same underlying distribution. The two-sided p -value is then estimated as the proportion of permuted differences whose absolute value exceeds that of the observed difference. This procedure is applied separately to each network metric, resulting in independent significance assessments for w_{ij} edge weights, k_i node degrees, and s_i node strengths. The box plot of these 3 characteristics is shown in Figure 2.

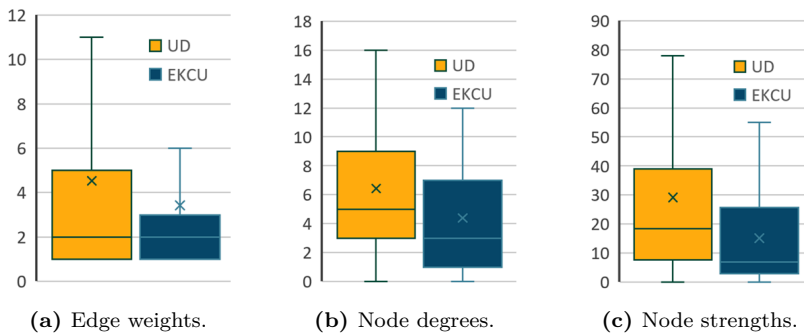


Figure 2. The box plot of the values of different network metrics for UD (orange) and EKC (darkblue).

The box plots represent different statistical samples, but the statistical significance of the differences of metrics for these two compared weighted co-authorship networks can be captured by the p -values. This investigation revealed that link weights in these two networks there are no relevant differences ($p = 0.1915$). However, the collaboration networks of UD and EKC are significantly different from the point of view of the node degrees (where $p = 0.0291$) and node strength (where $p = 0.0121$).

The researchers of a given department can be grouped together into one node

representing the department within the faculty. In this way, one can focus on only the $e \in E^{inter}$ inter-department links. Results are illustrated in Figure 3.

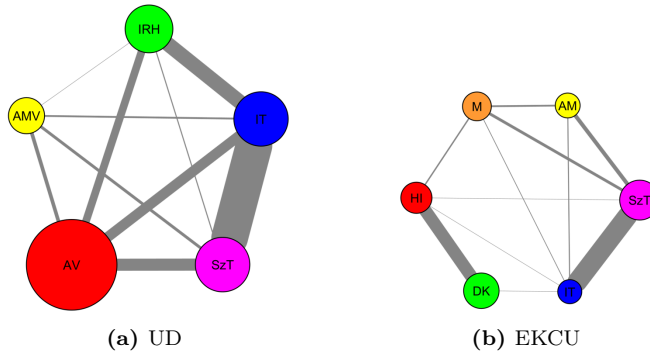


Figure 3. Department-level co-authorship network of faculties. The area of nodes is proportional to the number of publications that belong to the given department. The width of links is proportional to the edge weight.

One of the most conspicuous differences between the two faculties is that, although the UD network is much larger than the EKC network, the latter is still divided into more departments. This results in smaller departments with far fewer publications. One can see in Figure 3, that there are strongly and weakly connected departments in both faculties. This can be explained on the one hand by the different research areas of the departments, but we should also not ignore the fact that only a few research collaborations belonging to different departments can dominate the collaborations between departments that seem significant at first glance. In the latter, the restructure of the organization based on research interest could strongly influence inter-department collaborations.

4. Conclusions

This study set out to investigate the structure and characteristics of local scientific collaboration networks at the faculty level, with a particular focus on two Hungarian institutions (both faculties have the same research fields, namely computer science). By constructing weighted co-authorship networks from the Hungarian MTMT database, the research aimed to quantify interpersonal and inter-departmental collaboration patterns, and to reveal structural similarities and differences between the examined academic communities.

Methodologically, the work combined curated bibliographic data with tools from network science and statistical analysis. Co-authorship relations were modeled as weighted, undirected graphs, where edge weights captured the intensity of collaboration. In addition to standard network measures (e.g., degree, strength, clustering coefficient), a permutation-based statistical framework was applied to rigorously

assess whether observed differences between networks could be attributed to structural effects or random variation. The analysis was performed at multiple levels, including extended collaboration networks, restricted local faculty networks, and aggregated department-level structures.

The results demonstrate that, despite differences in institutional size, certain global properties – such as characteristic path length and heterogeneity – are remarkably similar across the two faculties. At the same time, significant differences were identified in key structural metrics, including node degree and node strength distributions, indicating distinct collaboration patterns. The presence of hubs, the dominance of a giant component alongside isolated researchers, and the variability in inter-departmental connectivity all highlight the complex and heterogeneous nature of scientific collaboration at the local level. Furthermore, the analysis revealed that apparent differences in simple descriptive metrics (e.g., density or average degree) cannot be fully understood without statistical validation that accounts for network structure.

Importantly, these insights are inherently tied to the network-based approach. Many of the observed phenomena—such as the identification of central actors, the role of strong and weak ties, the structure of collaboration components, or the significance of inter-departmental links—cannot be captured by traditional, non-relational statistical methods. The network perspective enables the simultaneous consideration of individual behavior and system-level structure, thereby uncovering patterns that would otherwise remain hidden.

Beyond methodological contributions, the findings have practical relevance. Understanding the structure of local collaboration networks can support institutional strategy, for example, by identifying key researchers, potential collaboration gaps, or structurally isolated individuals and groups. At the department level, insights into interconnections may inform organizational design, interdisciplinary initiatives, and resource allocation. Consequently, such analyses can provide valuable input for local academic communities, faculty leadership, and higher-level decision-makers seeking to foster more effective and integrated research environments.

References

- [1] T. ARAÚJO, E. FONTAINHA: *The specific shapes of gender imbalance in scientific authorships: A network approach*, Journal of Informetrics 11.1 (2017), pp. 88–102, DOI: [10.1016/j.joi.2016.11.002](https://doi.org/10.1016/j.joi.2016.11.002).
- [2] C. BIRÓ, G. KUSPER: *Some k-hop based graph metrics and node ranking in wireless sensor networks*, Annales Mathematicae et Informaticae 50 (2019), pp. 19–37, DOI: [10.33039/ami.2018.09.002](https://doi.org/10.33039/ami.2018.09.002).
- [3] F. CATALÁ-LÓPEZ, A. ALONSO-ARROYO, M. J. PAGE, L. CASTELLÓ-COGOLLOS, B. HUTTON, M. RIDAO, R. TABARÉS-SEISDEDOS, R. ALEIXANDRE-BENAVENT, D. MOHER: *A cross-sectional analysis identified co-authorship networks and scientific collaboration on reporting guidelines for health research*, Journal of Clinical Epidemiology 157 (2023), pp. 22–34, DOI: [10.1016/j.jclinepi.2023.02.001](https://doi.org/10.1016/j.jclinepi.2023.02.001).

- [4] Y. CHEN, C. DING, J. HU, R. CHEN, P. HUI, X. FU: *Building and Analyzing a Global Co-Authorship Network Using Google Scholar Data*, in: Proceedings of the 26th International Conference on World Wide Web Companion, 2017, pp. 1219–1224, DOI: [10.1145/3041021.3053056](https://doi.org/10.1145/3041021.3053056).
- [5] S. DROŹDŹ, A. KULIG, J. KWAPIEŃ, A. NIEWIAROWSKI, M. STANUSZEK: *Hierarchical organization of H. Eugene Stanley scientific collaboration community in weighted network representation*, Journal of Informetrics 11.4 (2017), pp. 1114–1127, DOI: [10.1016/j.joi.2017.09.009](https://doi.org/10.1016/j.joi.2017.09.009).
- [6] D. V. FILHO, D. R. J. O'NEALE: *Structure dynamics of evolving scientific networks*, arXiv, 2005.13251 (2020), pp. 1–16, DOI: [10.48550/arXiv.2005.13251](https://doi.org/10.48550/arXiv.2005.13251).
- [7] M. J. KRAWCZYK, M. LIBIRT, K. MALARZ: *Analysis of scientific cooperation at the international and intercontinental level*, Scientometrics 129 (2024), pp. 4983–5002, DOI: [10.1007/s11192-024-05107-y](https://doi.org/10.1007/s11192-024-05107-y).
- [8] MTA: *MTMT*, 2026, URL: <https://www.mtmt.hu> (visited on 01/10/2026).
- [9] M. E. J. NEWMAN: *Scientific collaboration networks. I. Network construction and fundamental results*, Phys. Rev. E 64 (1 2001), p. 016131, DOI: [10.1103/PhysRevE.64.016131](https://doi.org/10.1103/PhysRevE.64.016131).
- [10] M. E. J. NEWMAN: *The structure of scientific collaboration networks*, Proceedings of the National Academy of Sciences 98.2 (2001), pp. 404–409, DOI: [10.1073/pnas.98.2.404](https://doi.org/10.1073/pnas.98.2.404).
- [11] S. L. SIMPSON, R. G. LYDAY, S. HAYASAKA, A. P. MARSH, P. J. LAURIENTI: *A permutation testing framework to compare groups of brain networks*, Frontiers in Computational Neuroscience 7 (2013), p. 171, DOI: [10.3389/fncom.2013.00171](https://doi.org/10.3389/fncom.2013.00171).
- [12] A. ZINILLI: *Elements of Network Science*, Springer Nature Switzerland, 2025, DOI: [10.1007/978-3-031-84712-7](https://doi.org/10.1007/978-3-031-84712-7).