

Modeling and Optimizing for NP-hard Problems in Graph Theory

Summary of Ph.D. Thesis

Ahmad Anaqreh

Supervisors: Dr. Boglárka G.-Tóth and Dr. Tamás Vinkó

Doctoral School of Computer Science
Department of Computational Optimization
Faculty of Science and Informatics
University of Szeged



Szeged
2024

1 Introduction

Optimization involves the search for the best possible solution for a given problem. It stands as a fundamental concept among diverse disciplines such as mathematics, computer science, engineering, economics, and beyond. Selecting the appropriate optimization technique depends on the problem's category as well as the trade-off between the quality of the solution and the time sufficient for computation.

In combinatorial optimization, recommended approaches to achieve the best possible solution include standard methods like integer linear programming, binary integer linear programming, and linear programming with the relaxation for certain or all variables, yet still achieving an integer solution. Additionally, decomposition methods like Benders decomposition and Dantzig–Wolfe decomposition are efficient for some problem classes. Conversely, if reaching the optimal solution demands significant computational resources then the focus is on attaining a good enough solution within a reasonable time, thus metaheuristics and heuristics gain substantial importance.

Graph theory is the study of graphs, which serve as mathematical structures used to represent and analyze connections between objects. Its practical implementations include diverse domains, including computer science, network science, social science, biology, and more. The connection between optimization and graph theory has long been a significant focal point for researchers, involving the utilization of optimization techniques to deal with graph theory problems.

The main objective of this work is to tackle three NP-hard graph problems, these problems cannot be solved by deterministic algorithms in polynomial time, by utilizing various optimization techniques, including standard methods, heuristics, and metaheuristics. The evaluation of the proposed methods involves comparing their results and execution times with the methods presented in the literature.

The work is structured into five parts. Chapter 1 is an introductory chapter, that introduces the fundamentals of optimization and graph theory, and clarifies the interrelationship between these two domains. Chapter 2, discusses a graph problem known as graph geodetic number, and the two greedy algorithms proposed to obtain upper bounds for the geodetic number in an algorithmic way. Chapter 3, clarifies how Symbolic Regression with an evolutionary algorithm called Cartesian Genetic Programming has been used to derive formulas capable of approximating the graph geodetic number. In Chapter 4, the aim is to deal with the longest induced (chordless) cycle problem by proposing three integer linear programs to yield optimal solutions for this problem. In Chapter 5, the purpose of the work is to maximize the smallest eigenvalue of the grounded Laplacian matrix. Degree centrality is used as the base for the first method. In addition, the vertex cover problem was employed as an additional method of solving the problem.

Thesis 1: Algorithmic Upper Bounds for Graph Geodetic Number

The graph geodetic number is a global measure for simple connected graphs and it belongs to the path covering problems, which entails identifying the minimal-cardinality set of vertices, such that all shortest paths between its elements cover every vertex of the graph. It has been shown that computing the geodetic number is a computationally challenging task for general graphs, as it falls into the NP-hard problem category [12]. As is often the case with graph theoretical problems, an *ILP* formulation is applicable. Hansen and van Omme presented such a model in a recent paper [16], which also featured the first computational experiments conducted on a collection of moderately sized random graphs.

Drawing inspiration from these results, I conduct an empirical investigation into upper bound algorithms. The core concept behind the algorithms is to construct a geodetic set in an iterative greedy manner. At each iteration, the algorithm selects a vertex/vertices, to be added to the geodetic set that leads to the most expansion of the covered set. These algorithms, as demonstrated by the experiments, consistently produce acceptable results on diverse sets of graphs, and require relatively low computational time, even when applied to large-scale graphs.

The first algorithm known as greedy algorithm uses Floyd's algorithm to compute all-pair shortest paths. The algorithm uses two functions: *LargestIncrease* and *LargestIncrease-Pair*. These functions are responsible for determining the vertex (or vertex pair) if they are included in the geodetic set, resulting in the most expansion of the covered set. Namely, the vertices that would be covered if one vertex or pair of vertices were included in the geodetic set. Based on the number of covered vertices the function chooses to add one or a pair of vertices to the geodetic set.

The locally greedy algorithm serves the same purpose as the greedy algorithm, but instead of calculating all shortest paths using Floyd's algorithm, this approach calculates distances from a specific vertex to all vertices not yet included in the geodetic set using Dijkstra's algorithm. The algorithm takes a specific vertex as input, which can be either a degree-one vertex or a simplicial vertex. A simplicial vertex is defined as a vertex whose neighbors collectively form a clique, meaning that every pair of neighbors are adjacent. It has been proved in [11, 17] that degree-one and simplicial vertices are always part of the geodetic set. In each iteration, the algorithm returns a vertex that would maximize the growth of the covered set if it is included in the geodetic set.

To assess the performance and efficiency of the proposed algorithms, experiments were conducted on random graphs and real-world graphs to compare the results and execution time to the *BILP* from [16]. The results shown in Table 1 present the findings from real-world graphs. The exact refers to the geodetic number obtained using the *BILP*. greedy refers to the upper bound founded by the greedy algorithm by adding pairs of vertices. The AddOne represents the upper bound specified by the greedy algorithm, which involves adding one vertex per iteration. Meanwhile, local indicates the upper bound determined by the locally greedy algorithm.

The values reported in Table 1 demonstrate that *BILP* can require hours to determine the exact geodetic number for graphs containing thousands of vertices and edges. In contrast, the proposed algorithms were capable of providing acceptable upper bounds within a reasonable time. Even for the largest graph instance (ia-email-univ), both versions

Table 1: Numerical results for real-world graphs, time is given in seconds unless indicated otherwise

graph name(n,m)	exact		greedy		AddOne		local	
	value	time	value	time	value	time	value	time
karate(34,78)	16	0.06	16	0.024	16	0.019	16	0.015
mexican(35,117)	7	0.21	7	0.025	8	0.023	9	0.011
sawmill(36,62)	14	0.09	14	0.022	14	0.019	15	0.015
chesapeake(39,170)	5	0.12	5	0.026	5	0.023	5	0.008
ca-netscience(379,914)	253	37 m	256	21.6	260	20.8	264	14.1
bio-celegans(453,2025)	172	1 h	183	40.8	188	34.3	225	14.8
rt-twitter-copen(761,1029)	459	6 h	459	101.7	459	103.4	490	112.6
soc-wiki-vote(889,2914)	275	14 h	276	236.4	277	232.0	409	120.5
ia-email-univ(1133,5451)	244	16 h	248	698.6	250	677.9	464	269.0
average	160.6	4 h	162.7	122.1	164.1	118.7	210.8	59.0

of the greedy algorithm were able to generate slightly less accurate upper bounds than the exact value in less than 700 seconds. Although the locally greedy algorithm proved to be the fastest, it missed the upper bound by a considerably larger margin for the larger graphs compared to the greedy method. This pattern is also reflected in the average performance data provided in the last row of Table 1.

Thesis 2: Symbolic Regression for Approximating Graph Geodetic Number

Symbolic Regression is a mathematical modeling technique aimed at finding a formula that accurately fits a given output based on a set of inputs. The input parameters and constants in Symbolic Regression are predetermined. Symbolic Regression combines these input elements using a set of predefined arithmetic operators, such as (+, −, ×, ÷, etc.) to formulate a mathematical expression.

Cartesian Genetic Programming is an iteration-based evolutionary algorithm, developed by Miller [20], that works as follows: Cartesian Genetic Programming begins by creating a set of initial solutions, from which the best one is chosen by evaluating the solutions based on a fitness function. Then these solutions will be used to create the next generation in the algorithm. The next generation’s solutions will be a mixture of chosen solutions from the previous generations, where the new solutions should not be identical to the previous ones, which can be done by mutation. Eventually, the algorithm must terminate. There are two cases in which this occurs: if the algorithm has reached the maximum number of generations, or the algorithm has reached the target fitness. At this point, a final solution is selected and returned. Märtens *et al.* [18] work was a notable starting point as they utilized Symbolic Regression and Cartesian Genetic Programming with inputs as eigenvalues of the Laplacian and adjacency matrices to optimize graph diameter and isoperimetric number on real-world graphs. Thus, I have employed Symbolic Regression with Cartesian Genetic Programming to derive formulas capable of approximating the graph geodetic number.

To utilize Symbolic Regression with Cartesian Genetic Programming, a training dataset is required. Each dataset contains instances, with each instance comprising two main components: (i) parameters representing graph properties and selected constants as inputs and (ii) the exact value of the corresponding graph property as the output. To construct the training dataset, 120 connected subgraphs were generated from 10 real-world graphs from the Network Repository¹, varying in size (with $14 \leq N \leq 140$). This was accomplished using a straightforward procedure. Given a real-world graph $G = (V, E)$, the process began by randomly selecting a set $W \subset V$ of vertices. Subsequently, the subgraph of V with vertex set W was extracted. This subgraph, denoted as \hat{G} , might not necessarily be connected. As a final step, the largest connected component of \hat{G} was chosen. From these graphs parameters that are closely related to the graph geodetic number have been obtained: eigenvalues of the Adjacency and Laplacian matrices, the count of degree-one vertices, the count of simplicial vertices, the number of vertices, and the number of edges. More specifically, the list of parameters used as training data:

- 1) N, M, γ, σ , and constants 1, 2, 3, 4, 5
- 2) $N, M, \gamma, \sigma, \lambda_i, \lambda_{N-i-1}$ ($i = 1, \dots, 5$)
- 3) $N, M, \gamma, \sigma, \mu_i, \mu_{N-i-1}$ ($i = 1, \dots, 5$)
- 4) $N, M, \gamma, \sigma, \lambda_i, \lambda_{N-i-1}$ ($i = 1, \dots, 5$) and constants 1, 2, 3, 4, 5
- 5) $N, M, \gamma, \sigma, \mu_i, \mu_{N-i-1}$ ($i = 1, \dots, 5$) and constants 1, 2, 3, 4, 5

where N is the number of vertices, M is number of edges, λ_i is the i -th eigenvalue of Adjacency matrix, μ_i is the i -th eigenvalue of Laplacian matrix, γ is the number of vertices with degree one in the graph, and σ is the number of simplicial vertices in the graph.

Cartesian Genetic Programming then aims to combine these parameters and constants using arithmetic operators to achieve the desired output. The set of arithmetic operators used consistently across all cases includes $(+, -, \times, \div, \sqrt{x}, x^2, x^3)$. The implementation employed in this work was the CGP-Library, a cross-platform Cartesian Genetic Programming tool developed by Andrew Turner². To derive formulas, Cartesian Genetic Programming was executed a dozen times for each category. Among the generated formulas, the best ones were selected based on their absolute error and relative error compared to the exact values. Therefore, the best formulas exhibited the smallest errors. The best approximation for the geodetic number was obtained by formula (1):

$$\gamma + \sigma + \sqrt{M} - 2 \quad (1)$$

Formula (1) contains the number of degree-one vertices and the number of simplicial vertices as these vertices are proven to be essential components of the geodetic set, as demonstrated in previous research [11, 17]. Linear regression was employed to refine formula (1). Therefore, the formula can be expressed as:

$$0.99 \cdot \gamma + 0.79 \cdot \sigma + 0.97 \cdot \sqrt{M} - 0.99 \quad (2)$$

¹<http://networkrepository.com/>

²<http://www.cgplibrary.co.uk/>

To evaluate the accuracy of the formula, a validation process was conducted using 120 graphs (with $31 \leq N \leq 485$). The geodetic number was computed twice for each graph: first, the exact value was determined using the *BILP* from [16], and second, an approximation was obtained using formula (2). Figure 1 presents a comparison between the two sets of values for these graphs, illustrating that the approximations are close to the exact geodetic number values. Nevertheless, two noticeable gaps are evident in figure 1, signifying that for certain graphs, the approximation is significantly lower than the exact value. This occurs when the number of simplicial vertices and/or the number of degree-one vertices is zero for these specific graphs.

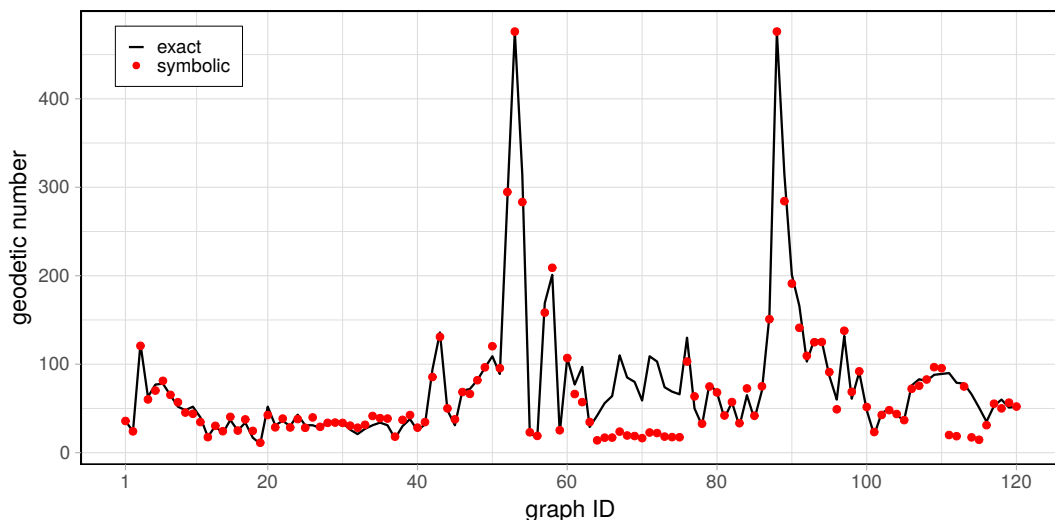


Figure 1: Exact $g(G)$ and values given by the optimized formula (2)

Thesis 3: Exact Methods for the Longest Induced Cycle Problem

The longest induced (or chordless) cycle problem is a graph problem defined as follows: For a graph $G = (V, E)$ and a subset $W \subseteq V$, the W -induced graph $G[W]$ comprises all the vertices from set W and the edges from G that connect vertices exclusively within W . The objective of the longest induced cycle problem is to determine the largest possible subset W for which the graph $G[W]$ forms a cycle.

I proposed three *ILP* designed to handle the longest induced cycle problem within general graphs. The first model is an order-based formulation, while the other two models build upon the models applied by prior work focused on solving the longest induced path problem [13, 19]. Cycle-elimination model, called *cec*, is a modified version of the model introduced in [19], and gave the best results compared to other models. Let $\delta(i) \subset E$ denote the edges incident to vertex i . The formalism of the model is as follows:

$$\max \sum_{i \in V} y_i \tag{3}$$

subject to

$$\sum_{e \in \delta(i)} x_e = 2y_i \quad \forall i \in V \quad (4)$$

$$x_e \leq y_i \quad \forall i \in V, e \in \delta(i) \quad (5)$$

$$x_e \geq y_i + y_j - 1 \quad \forall e = (i, j) \in E \quad (6)$$

$$\sum_{i \in C} y_i \leq |C| - 1 \quad \forall C \in \mathcal{C} \quad (7)$$

$$y_i \in \{0, 1\} \quad \forall i \in V \quad (8)$$

$$x_e \in \{0, 1\} \quad \forall e \in E \quad (9)$$

The binary decision variable y_i is equal to one if vertex i is part of the solution. Additionally, variable x_e is set to one if edge e is included in the solution. The objective function (3) aims to maximize the number of vertices within the induced cycle. Constraint (4) guarantees that each vertex within the solution is connected to precisely two vertices in the cycle. Constraints (5) and (6) are in place to ensure that the cycle is induced. To eliminate small cycles from solutions, constraint (7) is introduced. \mathcal{C} represents the set of the cycles for the given graph. Constraint (7) is combined into the model to enforce the solution to consist of a single cycle. This means that multiple small cycles are not deemed valid solutions.

cec model relies on the set of small cycles, which are usually created as part of the solution process, either through an iterative constraint generation approach or, more effectively, via branch-and-cut algorithm by employing separation. Hence, the branch-and-cut algorithm is employed as follows: the method is initiated with a model having no subtour elimination constraints, and if subtours arise in the solution, valid inequalities are added, and this process is repeated until the optimal solution is reached.

Consequently, the model was addressed using two distinct methodologies. The first approach is outlined in Algorithm 1. In each iteration, the model is solved. In line 4 the algorithm checks for the presence of an integer solution. Based on this, the depth-first search algorithm is employed to detect any subtours within the solution, as shown in line 5. If a subtour exists, and its length is less than or equal to the value of the variable `longest_induced_cycle`, a cut is appended for that cycle. If not, the value of the variable is updated to reflect the length of the cycle, and there is no need to introduce a cut because the cycle could potentially be the optimal solution. These details are clarified in lines 6 through 9. The procedure terminates when there are no further subtours present in the solution, indicating the completion of the procedure.

The second cut generation approach is detailed in Algorithm 2. In each iteration, the model is solved, and if an integer solution is obtained, the algorithm verifies the presence of any subtours using the depth-first search algorithm, as described in lines 15 through 16. If any cycles are detected, a cut is integrated into the model (line 17), and the length of the cycle is updated if it exceeds the value of the variable `longest_induced_cycle` (line 19). Towards the end of the procedure, the length of the longest induced cycle is recorded in the variable. It is important to note that a constraint is added to the model in line 20 to enhance the procedure. This constraint ensures that the objective value must be greater than or equal to the length of the largest induced cycle discovered thus far.

Algorithm 1: Cut_Generation1

```
1 model Initialization() // initializing the cec model
2 longest_induced_cycle=0
3 Function Cut_Generation1()
4   if model.status==feasible_integer then // model has integer solution
5     C=DFS(feasible_integer) // find subtour in the solution
6     if length(C) ≤ longest_induced_cycle then
7       model.addConstr(7) // add cut (7)
8     else
9       longest_induced_cycle=length(C) // update variable value
10 model.optimize(Cut_Generation1()) // solve the model
11 print(longest_induced_cycle)
```

Algorithm 2: Cut_Generation2

```
12 model Initialization() // initializing the cec model
13 longest_induced_cycle=0
14 Function Cut_Generation2()
15   if model.status==feasible_integer then // model has integer solution
16     C=DFS(feasible_integer) // find subtour in the solution
17     model.addConstr(7) // add cut (7)
18     if length(C) > longest_induced_cycle then
19       longest_induced_cycle=length(C)
20     model.addConstr(model.ObjVal ≥ longest_induced_cycle+1)
21 model.optimize(Cut_Generation2()) // solve the model
22 print(longest_induced_cycle)
```

To verify the efficacy of the proposed method, I conducted a comparison with the results presented in the literature. Pereira et al. [21] proposed an *ILP* formulation along with additional valid inequalities to strengthen and refine the formulation, all of which were incorporated into the branch-and-cut algorithm. They applied a multi-start heuristic method for initial solution generation and then conducted performance evaluations of the algorithms on a range of randomly generated graphs. Note that *cec2* refers to the method outlined in Algorithm 2.

The results for the random graphs are presented in Table 2, where *cec2* compared against the top three algorithms introduced in [21]. The runtime represents the average duration of ten graphs in each case. Notably, *cec2* outperforms these algorithms in all cases. Moreover, *cec2* successfully solved the instance with 100 vertices and 30% density, a scenario where none of the other algorithms succeeded. Instances that resulted in timeouts are denoted by the symbol 🕒, where every run was restricted to a maximum duration of one hour.

Table 2: Running times on random instances for *cec2*, and *BC1*, *BC2*, *BC3* from [21], time is given in seconds.

Randomly generated graphs: 10% density				
n	<i>cec2</i>	<i>BC1</i>	<i>BC2</i>	<i>BC3</i>
50	0.32	0.33	0.3	0.39
60	0.79	1.19	1.2	1.34
70	4.17	5.57	4.93	5.58
80	20.5	37.15	26.9	27.34
90	93.93	160.1	155.82	168.02
100	518.75	1321.41	1129.47	1094.8
average	106.41	254.29	219.77	216.25
Randomly generated graphs: 30% density				
n	<i>cec2</i>	<i>BC1</i>	<i>BC2</i>	<i>BC3</i>
50	4.15	8.21	9.6	8.78
60	26.14	39.82	46.18	51.9
70	90.11	234.45	206.36	283.28
80	203.81	935.78	676.52	1139.55
90	544.88	2072.16	1874.91	3011.17
100	1810.49	🕒	🕒	🕒
average	446.6	658.08	562.71	898.94

Thesis 4: Maximizing the Smallest Eigenvalue of the Grounded Laplacian Matrix

Maximizing the smallest eigenvalue of the grounded Laplacian matrix $L(S)$ is a problem that involves identifying $(n - k) \times (n - k)$ principal submatrix obtained after removing k rows and corresponding columns from the Laplacian matrix L , where $S \subset V$, $|S| = k$, $0 < k \ll n$. The smallest eigenvalue of $L(S)$ is denoted by $\mu(S)$. $L(S)$ is a symmetric positive definite matrix, thus all its eigenvalues are strictly positive real numbers. Hence, $\mu(S) > 0$ holds.

Two algorithms have been proposed to deal with the problem at hand. The first approach, referred to as DEGREE-G, draws inspiration from the well-known Gerschgorin circle theorem [15]. The theorem provides bounds on the eigenvalues of a square matrix. It claims that each eigenvalue of a square matrix resides within at least one Gerschgorin circle, with each circle corresponding to a row in the matrix. In this representation, the center of the circle corresponds to the diagonal element of the matrix, while the radius is determined by the sum of the absolute values of the off-diagonal elements. The key idea for maximizing the smallest eigenvalue is to move the Gerschgorin circles further away from the origin. To achieve this, the proposed method ranks the vertices based on a specific centrality measure and subsequently removes the corresponding row and column from the Laplacian matrix. Algorithm 3 outlines this approach which utilizes degree centrality for this purpose.

Algorithm 3: Degree-G Algorithm

```

1 vertex_cen = sort(centrality(V))
2 for  $i = 1 \rightarrow k$  do
3    $\lfloor$  remove( $L(\textit{vertex\_cen}[i])$ )
4 compute( $\mu(S)$ )

```

The second method known as COVER, takes advantage of the Gerschgorin circles, but it utilizes the concept of the maximum k vertex cover problem. This problem is rooted in the classic vertex cover problem [14], with the key distinction being that in the k vertex cover problem, the objective is to identify a set of k vertices that are incident to the maximum number of edges in the graph. This differs from the vertex cover problem, which aims to find the minimum number of vertices such that each edge in the graph is incident to at least one of these vertices. The *BILP* of k vertex cover is defined as follows:

$$\max \sum_{j \in V} y_j \tag{10}$$

subject to

$$\sum_{i \in V} x_i = k \quad (11)$$

$$y_j \leq \sum_{\forall i \in V: (j,i) \in E} x_i \quad \forall j \in V \quad (12)$$

$$k \in \mathbb{N} \quad (13)$$

$$x_i, y_i \in \{0, 1\} \quad \forall i \in V \quad (14)$$

The variables x_i represent vertices that cover the maximum number of vertices within the graph, while y_i denotes vertices that have been covered. The constraints are designed to guarantee that only k vertices can be chosen and that the value of y_i is equal to 1 if and only if at least one of its adjacent vertices, represented by x_i , is selected. After successfully solving this *BILP*, the algorithm proceeds to remove the rows and columns in L corresponding to the solution. Subsequently, it determines the smallest eigenvalue of the resulting modified matrix.

Additionally, combining vertex cover and degree centrality concepts has the potential to enhance the outcomes. Consequently, the objective function in the *BILP* has the following modifications:

$$\max \sum_{j \in V} y_j - \epsilon \sum_{j \in V} deg_j x_j, \quad (15)$$

where ϵ is a small number to not change the main objective. For instance $\epsilon = \frac{1}{\sum_{j \in V} deg_j}$ can be chosen. This modification aims to select k vertices with the lowest degree to maximize the objective. This approach is defined as COVER1.

The maximum k vertex cover problem often yields multiple solutions, starting to employ a method to explore the possibility of obtaining an improved value for $\mu(S)$ through alternate solutions. The concept involves an iterative process of solving the *BILP* while including a constraint that prevents the solution from resembling previous ones. By examining various solutions, different values for $\mu(S)$ can be obtained, and select the one that yields the best result. This additional constraint is formulated as follows for a given previously obtained solution $S \subset V$:

$$\sum_{i \in S} x_i \leq k - 1. \quad (16)$$

During each iteration, the algorithm ensures that the solution covers the maximum number of vertices, denoted as nc , which means that $\sum_{i \in V} y_i = nc$ must hold; otherwise, the algorithm terminates the iteration. This approach is referred to as COVER2. The maximum number of iterations, and consequently the maximum number of alternative solutions, is fixed at 100.

To evaluate and compare the effectiveness of the proposed methods, a comparative analysis against the algorithms introduced in the literature was conducted. Wang *et al.* [22] proposed two algorithms, the first one, referred to as the NAÏVE algorithm, involves k iterations. In each iteration, a candidate is chosen if adding it to set S maximizes

$\mu(S)$. The second algorithm, referred to as the FAST algorithm, evaluates a candidate node based on the sum of the eigenvalues of its adjacent nodes. The optimal candidate is chosen based on the maximum sum value.

Figure 2 displays the outcomes obtained by applying the proposed approaches to a range of real-world graphs. The figure presents the smallest eigenvalue, denoted as $\mu(S)$, for six distinct values of k .

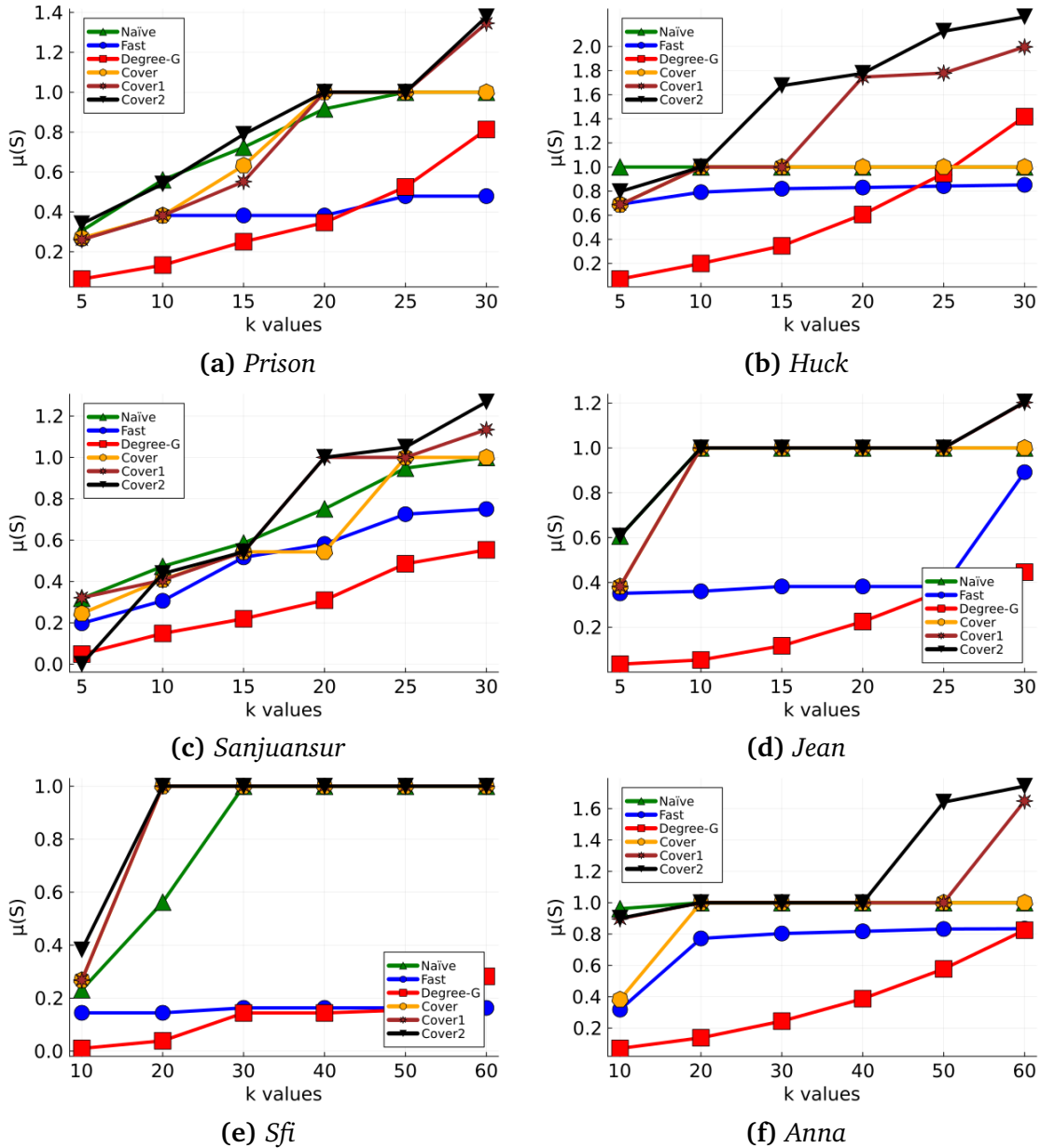


Figure 2: Values of $\mu(S)$ obtained by the algorithms for different k values.

The displayed results illustrate that the effectiveness of these methods varies across different graphs, yet certain patterns arise. It is evident that the DEGREE-G and FAST methods exhibit inferior performance compared to the NAÏVE and COVER methods. Notably, the NAÏVE method consistently outperforms the other techniques, although the COVER method performs equally well or even better than the NAÏVE method at specific values of k .

2 Contributions of the thesis

In the **first thesis group**, the contributions are related to obtaining upper bounds of the graph geodetic number in an algorithmic way which was capable of providing solutions of acceptable quality within a reasonable time.

- I/1. I introduced two greedy-type algorithms. The first, known as the greedy algorithm, relies on Floyd's algorithm, while the local greedy algorithm is based on Dijkstra's algorithm.
- I/2. I have empirically demonstrated that the proposed algorithms can efficiently obtain upper bounds that closely approximate the optimal solution obtained from the binary integer linear programming. Meanwhile, their computational time remains a small fraction of that needed to obtain the exact geodetic number.

In the **second thesis group**, the contributions are related to using Symbolic Regression and Cartesian Genetic Programming to derive optimized formulas for graph geodetic number.

- II/1. I have used Symbolic Regression together with Cartesian Genetic Programming to derive a general formula that approximates the graph geodetic number. The formula is simply the sum of the number of edges, the number of degree-one vertices, and the number of simplicial vertices. Thus, the approximation of the geodetic number can be obtained in a reasonable computational time, even for graphs with thousands of vertices and edges.
- II/2. I demonstrated how different training sets will lead to different formulas with different accuracy which validates that using parameters that are highly related to the graph property as training data will help in approximation in a better manner.

In the **third thesis group**, the contributions are related to finding an efficient approach that can deliver optimal solutions within a reasonable time for the longest induced cycle problem. Exact methods have been introduced and evaluated against existing methods from the literature in order to address this problem.

- III/1. I proposed three integer linear programs, some of which are extensions of models created for solving the longest induced path problem. The proposed programs had varying execution times and the number of instances they were able to solve optimally.
- III/2. I conducted a comparison between my methods and the methods proposed previously in the literature, and the results demonstrated that the newly introduced methods consistently outperformed those presented in the literature.

In the **fourth thesis group**, the contributions are related to proposing new methods to maximize the smallest eigenvalue of the grounded Laplacian matrix, which can deliver

solutions of acceptable quality within a reasonable time. Two approaches were introduced and demonstrated through experiments and compared to the methods proposed in the literature.

- IV/1. I proposed two algorithms to address the problem: the first one, named DEGREE-G relies on vertex centrality, while the second one, called COVER is based on the vertex cover problem.
- IV/2. Both algorithms, as evidenced by experimental results, consistently produced promising solutions within a reasonable time, highlighting their competitiveness when compared to existing algorithms in the literature.

The author’s publications on the subjects of the thesis

Table 3 summarizes the relation between the thesis points and the corresponding publications.

Table 3: Correspondence between the thesis points and my publications.

Publication	Thesis point							
	I/1	I/2	II/1	II/2	III/1	III/2	IV/1	IV/2
[J1]	•	•						
[J2]			•	•				
[J3]					•	•		
[C1]							•	•

Journal publications

- [J1] **Anaqreh, Ahmad T** and G.-Tóth, Boglárka and Vinkó, Tamás. Algorithmic Upper Bounds for Graph Geodetic Number. *Central European Journal of Operations Research* **30**, 1221–1237, 2022.
- [J2] **Anaqreh, Ahmad T** and G.-Tóth, Boglárka, G and Vinkó, Tamás. Symbolic Regression for Approximating Graph Geodetic Number. *Acta Cybernetica* **25**, 151–169, 2021.
- [J3] **Anaqreh, Ahmad T** and G.-Tóth, Boglárka and Vinkó, Tamás. Exact Methods for the Longest Induced Cycle Problem. *Discrete Applied Mathematics (under review)*.

Full papers in conference proceedings

- [C1] **Anaqreh, Ahmad T** and G.-Tóth, Boglárka and Vinkó, Tamás. New Methods for Maximizing the Smallest Eigenvalue of the Grounded Laplacian Matrix. *In Proceedings of the 12th International Conference on Applied Informatics. ICAI 2023*, 1–9, 2023.

Further publications

- [J4] Baniata, Hamza and **Anaqreh, Ahmad** and Kertesz, Attila. PF-BTS: A Privacy-Aware Fog-enhanced Blockchain-assisted task scheduling. *Information Processing & Management*, **58**, 102393, 2021.

- [J5] Baniata, Hamza and **Anaqreh, Ahmad** and Kertesz, Attila. DONS: Dynamic Optimized Neighbor Selection for smart blockchain networks. *Future Generation Computer Systems*, **130**, 75–90, 2022.
- [J6] Baniata, Hamza and **Anaqreh, Ahmad** and Kertesz, Attila. Distributed Scalability Tuning for Evolutionary Sharding Optimization with Random-equivalent Security in Permissionless Blockchain. *Internet of Thing*, **24**, 100955, 2023.

Other References

- [11] H A Ahangar, F Fujie-Okamoto, and V Samodivkin. On the forcing connected geodetic number and the connected geodetic number of a graph. *Ars Combinatoria*, 126:323–335, 2016.
- [12] M Atici. Computational complexity of geodetic set. *International journal of computer mathematics*, 79(5):587–591, 2002.
- [13] Fritz Bökler, Markus Chimani, Mirko H Wagner, and Tilo Wiedera. An experimental study of ILP formulations for the longest induced path problem. In *International Symposium on Combinatorial Optimization*, pages 89–101. Springer, 2020.
- [14] Jianer Chen, Iyad A Kanj, and Weijia Jia. Vertex cover: further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001.
- [15] Gene H Golub and Charles F Van Loan. *Matrix computations*. Johns Hopkins University Press, 2013.
- [16] P Hansen and N van Omme. On pitfalls in computing the geodetic number of a graph. *Optimization Letters*, 1(3):299–307, 2007.
- [17] F Harary, E Loukakis, and C Tsouros. The geodetic number of a graph. *Mathematical and Computer Modeling*, 17(11):89–95, 1993.
- [18] M Märten, F Kuipers, and P Van Mieghem. Symbolic regression on network properties. *Genetic Programming*, pages 131–146, 2017.
- [19] Ruslán G Marzo, Rafael A Melo, Celso C Ribeiro, and Marcio C Santos. New formulations and branch-and-cut procedures for the longest induced path problem. *Computers & Operations Research*, 139:105627, 2022.
- [20] J Miller. *Cartesian genetic programming*. Springer, 2011.
- [21] Dilson Lucas Pereira, Abilio Lucena, Alexandre Salles da Cunha, and Luidi Simonetti. Exact solution algorithms for the chordless cycle problem. *INFORMS Journal on Computing*, 34(4):1970–1986, 2022.
- [22] Run Wang, Xiaotian Zhou, Wei Li, and Zhongzhi Zhang. Maximizing the smallest eigenvalue of grounded laplacian matrix. *arXiv preprint arXiv:2110.12576*, 2021.

3 Összefoglalás

Ez a PhD értekezés optimalizációs módszereket mutat be NP-nehez gráfproblémákhoz. Ezen feladatok megoldására, jelen tudásunk szerint, nem léteznek polinomiális időben végrehajtható algoritmusok. Konkrétan három gráfproblémát vizsgáltunk, és mindegyikhez különböző optimalizációs módszereket alkalmaztunk, úgy mint egészértékű lineáris programozás, metaheurisztikák és heurisztikák. Minden esetben a javasolt módszerek teljesítményét összehasonlítottuk a szakirodalomban találhatóakkal, figyelembe véve olyan tényezőket, mint a végrehajtási idő és az elért megoldások minősége. Az összehasonlító elemzéseink célja a javasolt optimalizációs módszerek hatékonyságának bemutatása.

A disszertáció öt részre tagolódik. Az 1. fejezet bevezető jellegű, ahol az optimalizálás és a gráfelmélet alapjait mutatjuk be, valamint tisztázzuk ezek két terület kölcsönhatását. A 2. fejezet az első NP-nehez gráfproblémáról, a gráf geodetikus számról szól, valamint két mohó algoritmust tárgyal, amelyek algoritmikus felső korlátokat szolgáltatnak a geodetikus számhoz. A 3. fejezet azt tárgyalja, hogyan alkalmaztuk a szimbolikus regressziót a genetikus programozás segítségével annak érdekében, hogy olyan formulákat állítsunk elő, amelyek képesek a gráf geodetikus számát elfogadhatóan jó közelítéssel megbecsülni. A 4. fejezet témája a leghosszabb indukált kör problémájának megoldása, amely egy NP-teljes gráfprobléma, három vegyes egészértékű lineáris program használatával, amelyek optimális megoldásokat szolgáltatnak erre a problémára. Az 5. fejezetben vizsgált feladat a Dirichlet-Laplace mátrix (az angol nyelvű szakirodalomban *grounded Laplacian* mátrix) legkisebb sajátértékének maximalizálása. A javasolt módszereink alapja egyrészt a foksám centralitás, valamint a csúcsfedés probléma alkalmas átfogalmazása.

Declaration

In the Ph.D. dissertation of Ahmad Anaqreh entitled “**Modeling and Optimizing for NP-hard Problems in Graph Theory**”, with list of publications:

[J1] Anaqreh, Ahmad T and G.-Toth, Boglarka and Vinko, Tamas. Algorithmic Upper Bounds for Graph Geodetic Number. *Central European Journal of Operations Research* 30, 1221–1237, 2022.

[J2] Anaqreh, Ahmad T and G.-Toth, Boglarka, G and Vinko, Tamas. Symbolic Regression for Approximating Graph Geodetic Number. *Acta Cybernetica* 25, 151–169, 2021.

[J3] Anaqreh, Ahmad T and G.-Toth, Boglarka and Vinko, Tamas. Exact Methods for the Longest Induced Cycle Problem. *Discrete Applied Mathematics* (under review).

[C1] Anaqreh, Ahmad T and G.-Toth, Boglarka and Vinko, Tamas. New Methods for Maximizing the Smallest Eigenvalue of the Grounded Laplacian Matrix. In *Proceedings of the 12th International Conference on Applied Informatics. ICAI 2023*, 1–9, 2023.

Ahmad's contribution was decisive in the following results:

- Thesis I dealt with a graph problem known as graph geodetic number, which is a global measure for simple connected graphs and it belongs to the path covering problems, to find the minimal-cardinality set of vertices, such that all shortest paths between its elements cover every vertex of the graph. Two greedy algorithms were proposed to obtain upper bounds for the geodetic number in an algorithmic way. The efficiency of these algorithms is demonstrated on real-world graphs and randomly generated graphs. [J1]
- In thesis II, Symbolic Regression with an evolutionary algorithm called Cartesian Genetic Programming has been used to derive formulas capable of approximating the graph geodetic number. The obtained formulas are evaluated on random and real-world graphs. It demonstrated how various graph properties as training data can lead to diverse formulas with different accuracy. It also investigated which training data are closely related to the graph property. [J2]

- In thesis III, the work aimed to deal with the longest induced (chordless) cycle problem, which is a graph problem involves the task of determining the largest possible subset of vertices within a graph in such a way that the induced subgraph forms a cycle. Within this work, three integer linear programs have been proposed to yield optimal solutions for this problem. To demonstrate the efficiency of these methods, experiments were conducted on a range of real-world graphs as well as random graphs. Additionally, a comparative analysis against approaches previously proposed in the literature was performed. [J3]
- In thesis IV, the purpose is to maximize the smallest eigenvalue of the grounded Laplacian matrix which is the Laplacian matrix's $(n-k) \times (n-k)$ submatrix after k rows and associated columns have been removed. Motivated by the Gershgorin circle theorem, the degree centrality is used to select k nodes that would maximize the smallest eigenvalue. In addition, the vertex cover problem was employed as an additional method of solving the problem. The efficiency of these approaches is demonstrated on real-world graphs and compared to the methods proposed in the literature.

These results cannot be used to obtain an academic research degree, other than the submitted Ph.D. thesis of Ahmad Anaqreh.

Szeged, 15-01-2024

Ahmad Anaqreh
Ph.D. candidate



Boglarka G.-Toth, Ph.D.
Supervisor




Tamas Vinko, Ph.D.
Supervisor



The head of the Doctoral School of Computer Science declares that the declaration above was sent to all of the coauthors and none of them raised any objections against it.

Szeged, 15-01-2024




Mark Jelasity, DSc
Head of the Doctoral School

