Quantum Annealing for Minimum Bisection Problem: A Machine Learning-based Approach for Penalty Parameter Tuning

Renáta Rusnáková ⓘ, Martin Chovanec ⓘ Juraj Gazda ⓘ

Abstract—**The Minimum Bisection Problem is a well-known NP-hard problem in combinatorial optimization, with practical applications in areas such as parallel computing, network design, and machine learning. In this paper, we examine the potential of using D-Wave Systems' quantum annealing solvers to solve the Minimum Bisection Problem, which we formulate as a Quadratic Unconstrained Binary Optimization model. A key challenge in this formulation lies in choosing an appropriate penalty parameter, as it plays a crucial role in ensuring both the quality of the solution and the satisfaction of the problem's constraints. To address this, we introduce a novel machine learning-based approach for adaptive tuning of the penalty parameter. Specifically, we use a Gradient Boosting Regressor model trained to predict suitable penalty parameter values based on structural properties of the input graph, the number of nodes and the graph's density. This method enables the penalty parameter to be adjusted dynamically for each specific problem instance, improving the solver's ability to balance the competing goals of minimizing the cut size and maintaining equally sized partitions. We test our approach on a large dataset of randomly generated Erdős–Rényi graphs with up to 4000 nodes, and we compare the results with classical partitioning algorithms, Metis and Kernighan–Lin. Experimental findings demonstrate that our adaptive tuning strategy significantly improves the performance of quantum annealing hybrid solver and consistently outperforms the classical methods used, indicating its potential as an alternative for graph partitioning problem.**

Index Terms—**D-Wave Systems, Gradient Boosting Regressor, Graph Partitioning, Hybrid Computing, Machine Learning, Minimum Bisection Problem, Optimization, Penalty Parameter Tuning, QUBO, Quantum Annealing**

I. INTRODUCTION

GRAPH partitioning is a fundamental problem in combinatorial optimization with applications in task scheduling for multiprocessor computers with focus on parallel computation, partitioning the circuit with applications in microchips design, social network analysis e.g. Facebook topics segmentation, clustering of data in machine learning as well as traffic optimization and logistics. The problem involves dividing a given graph G into two or more subsets while optimizing certain objective, such as minimizing the number of inter-edges and/or assigned costs between them and producing balanced partitions. While general Graph Partitioning Problem (GPP) allow for flexible partition sizes, a significant special case is the Minimum Bisection Problem (MBP), where the graph is partitioned into two equal-sized subsets while minimizing the number of inter-edges [1].

The MBP is classified as NP-hard, meaning that no polynomial-time algorithm exists that guarantees an optimal solution for all instances [2], [3]. While exact classical methods, such as integer linear programming, can be applied, they become computationally infeasible for large graphs due to their exponential time complexity [4]. As a result, a variety of heuristic and approximation algorithms have been proposed to find near-optimal solutions efficiently [5]–[10]. Well-known heuristics include the Metis algorithm, developed by Karypis and Kumar, which applies multilevel recursive bi-sectioning, where the graph is coarsened into smaller subgraphs, partitioned, and then refined iteratively, making it highly scalable [11], [12]. Alternatively, the Kernighan-Lin algorithm provides a local refinement heuristic by iteratively swapping nodes between partitions to minimize the cut size, though its computational complexity grows significantly with graph size [13], [14]. Despite existence of these approaches, obtaining high-quality partitions remains challenging, especially for large and dense graphs.

Recent advancements in quantum computing have introduced new possibilities for solving complex optimization problems such as MBP [15]. One commonly used modeling approach in this area is Quadratic Unconstrained Binary Optimization (QUBO). It's a mathematical formulation used to represent combinatorial optimization problems in a way that can be solved by classical or quantum optimization algorithms—especially those used in Quantum Annealing (QA) systems [16]. Unlike classical algorithms, quantum annealers exploit quantum mechanics' effects (superposition and tunneling) to explore the solution space more efficiently.

D-Wave Systems, a pioneering company providing quantum annealers also for commercial use, has developed not only pure Quantum Processing Unit (QPU) solvers but also Quantum Annealing Hybrid Solvers (QA HS) that combine quantum annealing with classical optimization techniques [17]. These QA HS are capable of handling large-scale problem instances [18] and are used in this study.

Several studies have explored quantum annealing for different variants of graph partitioning problem, demonstrating that QA can achieve better results compared to classical solvers, particularly for small and moderately sized graphs in [19], [20]. For comparison we introduce the following studies, which are the most related to our work.

Ushijima-Mwesigwa et al. [21] investigated the application of quantum annealing on the D-Wave Systems 2X, an older QA implementation, for graph partitioning tasks,

comparing it with classical methods, with a focus on both computational time and solution quality[1]. The results show that while quantum annealing is competitive, it does not always outperform classical solvers like Metis and KaHIP in terms of solution quality. In contrast, our study emphasizes solution quality, demonstrating that proper penalty parameter tuning can significantly enhance the performance of quantum annealing, leading to superior results compared to classical solvers (Metis, Kerninghan-Lin).

In [22], Hartman et al., used the D-Wave Advantage Systems, the same QA implementation as in our study, to test the QUBO formulation for the specific graph partitioning problem—power grid partitioning. They demonstrated the application of quantum annealing for optimizing power grid simulations while highlighting the quantum hardware limitations and the need for further penalty parameter tuning. The study focuses more on the practical implementation of the approach, whereas our study is more focusing on the improvement of results.

Djidjev et al. [23] presented an in-depth study on combinatorial optimization using quantum annealers, specifically investigating their applicability to the MBP. Their work provided a comparative evaluation between D-Wave Systems 2X and classical algorithms (Gurobi, Metis). The study focused on direct QPU testing and emphasized the impact of hardware-embedding strategies and their limitations on solution quality. Due to embedding constraints, the maximum number of nodes used for MBP testing was 48. In our study, we extend the scope to QA HS, exploring their potential for improving solution quality and scalability for larger graph instances.

In summary, in this article, we provide a comprehensive evaluation of QA HS for solving the MBP, comparing its performance with classical methods, Metis and Kerninghan-Lin, using a dataset of randomly generated graphs based on the Erdős–Rényi model. Our QUBO formulation explicitly encodes both the cut-size minimization objective and a penalty term to enforce balanced partitions. A key challenge is determining the penalty parameter, which dictates the trade-off between balance enforcement and cut-size minimization. To address this, we first propose a mathematical approach for setting the penalty parameter using lower and upper bound estimations. We then refine this further using a machine learning-based model, Gradient Boosting Regressors (GBR) [24], to predict optimal penalty parameter values based on graph properties. Our empirical findings show that this adaptive approach significantly enhances the accuracy of QA HS, making it superior to state-of-the-art classical solvers (Metis, Kerninghan-Lin) and providing more optimal solutions in 100% of cases for graphs with up to 4000 nodes.[2]

The key contributions of this study can be summarized as:

- We evaluate the feasibility and solution quality of QA HS and QPU for solving the MBP.

- We develop an adaptive penalty parameter tuning strategy- an evaluation process involving QA HS testing and analyses of collected results.
- We introduce a machine learning-based approach for predicting penalty parameter values, optimizing the solver's effectiveness across different graph structures.
- We compare QA HS and QPU solvers with classical algorithms on a dataset of Erdős–Rényi random graphs.
- And we provide insights into the practical applications of quantum annealing for MBP.

The rest of the paper is structured as follows: Section II presents the problem definition and the QUBO formulation of the MBP. Section III provides an overview of the proposed approach and key takeaways from previous research. Section IV describes the penalty parameter estimation process, which is later tuned using a machine learning-based approach in Section V. The simulation results are summarized in Section VI, and Section VII provides an overview of practical applications as well as our future directions. Finally, Section VIII concludes the study and outlines its contributions.

II. PROBLEM DEFINITION

Among the different formulations of GPP, the MBP is an important variant, where the goal is to divide the graph into two disjoint subsets of equal size while minimizing the number of edges between them.

The MBP is formally defined as follows. Given an unweighted, undirected graph $G(V, E)$, where V is the set of nodes with $|V| = n$ and E is the set of edges, the objective is to partition V into two equal-sized disjoint subsets[3]: $S_0, S_1 \subset V$, $S_0 \cap S_1 = \emptyset$, $S_0 \cup S_1 = V$, such that $|S_0| = |S_1|$, while minimizing the number of inter-edges (i.e., edges connecting nodes from S_0 to nodes from S_1) [25].

The rapid development of quantum computing has driven the need to adapt classical algorithms and reformulate problems so they can be processed by quantum hardware. Two main approaches have emerged:

- *Universal Quantum Computing* - based on quantum gates and circuits operating on qubits to solve complex problems (e.g. factorization or sorting)
- *Quantum Annealing* - exploits the adiabatic theorem and quantum annealing to slowly evolve the system towards the minimum energy configuration which encodes the optimal solution for optimization, search and sampling problems, such as MBP.

While companies like Google and IBM lead in universal quantum systems, D-Wave Systems specializes in commercial quantum annealers.

To use the quantum annealers, such as those developed by D-Wave Systems, for solving an optimization problem, the problem must be expressed either as an Ising model instance or equivalently, within the mathematical framework of QUBO [16], [26].

The Ising model originates from statistical mechanics and describes ferromagnetism through spin interactions, where

---

[1]In the context of optimization problems, solution quality refers to finding the optimal solution—the best possible outcome one can achieve from all feasible solutions according to the criteria set by the problem.

[2]Due to limitations in D-Wave's resources, we were unable to test larger graph instances.

[3]In this study we focus only on graphs with even number of nodes.

each spin takes a discrete value of -1 or $+1$ and interacts with its neighbors. The Ising problem, a problem formulated using the Ising model, involves finding a spin configuration that minimizes the total energy of the system. Many combinatorial optimization problems can be mapped onto the Ising problem [27], which has led to the development of Ising machines—physical systems capable of exploring the energy landscape and through thermal fluctuations and gradual cooling (annealing) navigate toward a low-energy state. However, classical Ising machines often become trapped in local minima, preventing them from reaching the optimal solution [28].

To address this, quantum Ising machines—quantum annealers [29], were introduced, leveraging quantum superposition and tunneling allowing the system to escape local minima and improve the chances of finding globally optimal solutions [30], [31].

The energy function for the Ising model is:

$$H(\mathbf{s}) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j , \qquad (1)$$

where $\mathbf{s} \in \{-1, +1\}^N$ is a vector of N binary spin variables, J_{ij} represents the interaction strength between spins s_i and s_j, h_i denotes the external magnetic field applied to spin s_i, and $H(\mathbf{s})$ is the Hamiltonian, representing the energy of the system.

The QUBO formulation [32] is mathematically equivalent to the Ising model but uses binary variables $x_i \in \{0, 1\}$, making it more suitable for defining combinatorial optimization problems. It takes the general form:

$$E(\mathbf{x}) = \sum_i \mathbf{Q}[i,i] x_i + \sum_{i<j} \mathbf{Q}[i,j] x_i x_j, \qquad (2)$$

where $\mathbf{Q}$ is a matrix encoding the objective function and penalty terms and $E(\mathbf{x})$ represents the overall function to be minimized. The vector of N binary variables $\mathbf{x} \in \{0,1\}^N$ is the solution to the given optimization problem if $\mathbf{x} = \operatorname{argmin}\{E(\mathbf{x})\}$ [33].

Since quantum annealers natively operate in the Ising model framework [34] and there is polynomial transformation between spins and binary variables ($x_i = \frac{s_i+1}{2}$) [21], the problem can be expressed in QUBO form and directly mapped onto their hardware. This compatibility allows problems such as MBP to be efficiently embedded [35] and solved using quantum annealers [16].

The QUBO formulation of MBP consists of two main terms:

- $E_{\text{cut}}(\mathbf{x})$ - an objective function that minimizes the number of inter-edges,
- $E_{\text{balance}}(\mathbf{x})$ - a penalty term that enforces equal partition sizes.

Based on the formulation used in related studies [36], the objective function is given by:

$$E_{\text{cut}}(\mathbf{x}) = \sum_{(i,j)\in E} (x_i + x_j - 2x_i x_j), \qquad (3)$$

where $x_i \in \{0, 1\}$ represents the binary partition assignment of node i either to set S_1 (in that case $x_i = 1$) or to the set S_0 ($x_i = 0$). The penalty term ensures that the partitions remain balanced:

$$E_{\text{balance}}(\mathbf{x}) = \lambda \left(\sum_{i\in V} x_i - \frac{n}{2} \right)^2, \qquad (4)$$

where the penalty parameter λ, often referred to as the Lagrange multiplier, plays a crucial role in QUBO formulation by determining the trade-off between minimizing the objective function and satisfying the penalty term.

A well-chosen λ ensures a balance between these two aspects. If λ is too small, the penalty is weak, potentially resulting in infeasible solutions where penalty term is not enforced. Conversely, if λ is too large, the solver prioritizes penalty satisfaction at the expense of optimizing the main objective, leading to poor-quality solutions [4].

The goal is to minimize the function $E_{\text{MBP}}(\mathbf{x})$, which is given as the sum of Equation (3) and Equation (4):

$$E_{\text{MBP}}(\mathbf{x}) = \sum_{(i,j)\in E} (x_i + x_j - 2x_i x_j) + \lambda \left(\sum_{i\in V} x_i - \frac{n}{2} \right)^2, \quad (5)$$

after expansion of the quadratic term[4]:

$$E_{\text{MBP}}(\mathbf{x}) = \sum_{(i,j)\in E} (x_i + x_j - 2x_i x_j) +$$
$$+ \lambda \left((1-n) \sum_{i\in V} x_i + 2 \sum_{i<j} x_i x_j + \frac{n^2}{4} \right). \quad (6)$$

The input to the quantum annealer is the matrix $\mathbf{Q}$ of coefficients, as described in the general QUBO formulation in Equation (2). Following [36] and Equation (6), the matrix for the MBP problem can be constructed as shown in Algorithm 1. Since the matrix is dependent on the structure of graph $G(V, E)$ (e.g. $|V| = |\mathbf{x}| = n$ is the order of $\mathbf{Q}$ and matrix elements $\mathbf{Q}[i,j]$ are coefficients of $x_i x_j$ which is nonzero if there exists and edge between node x_i and x_j) and also on the penalty parameter λ, we will denote it as $\mathbf{Q}_{G,\lambda}$.

Selecting an appropriate value for λ is particularly challenging, as there is no universal method for determining the optimal value. The tuning process is highly problem-specific and often relies on trial-and-error or heuristic approaches. One of the most commonly used and straightforward methods for estimating the penalty parameter is to base it on an upper bound of the objective function [15].

In this study, we begin with this estimation method, evaluate its effectiveness, and—based on the results—we propose a recalculation of λ, followed by a machine learning-based approach for tuning it. This adaptive method enables dynamic adjustment of the penalty parameter, with the goal of improving solution quality.

---

[4]Because $x_i \in \{0, 1\}$ are binary variables $x_i^2 = x_i$.

Algorithm 1 Constructing the matrix $\mathbf{Q}_{G,\lambda}$ for QUBO formulation of MBP

1: Initialize $\mathbf{Q}_{G,\lambda}$ as an empty matrix

2: Update $\mathbf{Q}_{G,\lambda}$ with coefficients from $E_{\text{cut}}(\mathbf{x})$
for each edge (i, j) in E **do**
    $\mathbf{Q}_{G,\lambda}[i, i] \leftarrow \mathbf{Q}_{G,\lambda}[i, i] + 1$
    $\mathbf{Q}_{G,\lambda}[j, j] \leftarrow \mathbf{Q}_{G,\lambda}[j, j] + 1$
    $\mathbf{Q}_{G,\lambda}[i, j] \leftarrow \mathbf{Q}_{G,\lambda}[i, j] - 2$
end for

3: Update $\mathbf{Q}_{G,\lambda}$ with coefficients from $E_{\text{balance}}(\mathbf{x})$
for each node i in V **do**
    $\mathbf{Q}_{G,\lambda}[i, i] \leftarrow \mathbf{Q}_{G,\lambda}[i, i] + \lambda(1 - n)$
end for
for each pair of nodes $(i, j), i \neq j$ **do**
    $\mathbf{Q}_{G,\lambda}[i, j] \leftarrow \mathbf{Q}_{G,\lambda}[i, j] + 2\lambda$
end for

III. PROPOSED APPROACH

The MBP problem was tested on graphs created following the Erdős–Rényi $G(n, p)$ model, where n[5] represents the number of nodes, and p[6] is the probability of an edge existing between any two nodes, controlling the overall graph density. The Erdős–Rényi model was chosen for its simplicity, as it allows for easy control of density through edge-probability p, which is critical for our testing. Additionally, it provides a straightforward way to calculate other graph properties, such as the maximum graph degree.

The problem was formulated as QUBO in form of a matrix and solved using D-Wave Systems' hybrid solver through the Leap cloud platform, accessed via the LeapHybridSampler class in the Ocean SDK.

To assess the effectiveness of QA HS, we used two well-established classical partitioning algorithms—Metis and Kernighan-Lin—as benchmarks. Metis, based on multilevel recursive bisection, is known for its efficiency in handling large-scale graphs, and we used its Python implementation, PyMetis [37], for our testing. For the Kernighan-Lin algorithm, we implemented the method ourselves in Python, as this allowed us to enforce a balanced solution by ensuring that the algorithm started with two equally sized sets before the first node assignment change. This adjustment was made to maintain fairness in comparison with QA HS, which inherently optimizes for balanced partitions.[7]

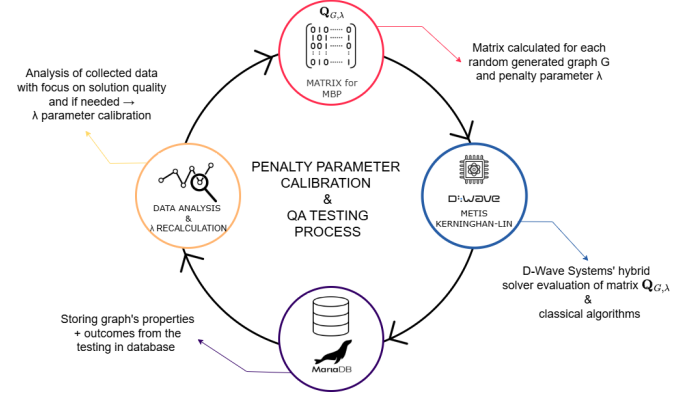Our proposed, novel approach consists of the following steps (graphically shown in Fig. 1):

1 Calculate the matrix $\mathbf{Q}_{G,\lambda}$ for random generated graph $G(n, p)$ and given penalty parameter λ.
2 Use the $\mathbf{Q}_{G,\lambda}$ as an input the the D-Wave Systems' QA HS & evaluate the problem using classical methods (Metis, Kerninghan-Lin algorithms).

---

[5] $n \in \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 2500, 3000, 4000\}$
[6] $p \in \{0.1, 0.25, 0.5, 0.75\}$
[7] The software is available in the corresponding GitHub repository.

Fig. 1. Diagram of penalty parameter calculation and testing process.

3 Store the graph properties, QA HS and classical algorithms results in database.
4 Analyze the results in terms of solution quality and constraint satisfaction → recalculate/tune penalty parameter λ if necessary.

The process starts with constructing QUBO matrix $\mathbf{Q}_{G,\lambda}$ following the Algorithm 1. For that we will need to establish the first estimate of penalty parameter λ.

At first we set the penalty parameter λ as the upper bound estimation for the objective function: In a fully connected graph, each node is connected to all others, meaning the maximum possible cut size for a balanced partition is

$$\text{max}_{\text{cut}} = \frac{n^2}{4}. \tag{7}$$

When considering the edge-probability p, the maximum number of edges in an Erdős–Rényi $G(n, p)$ random graph is reduced, and the expected maximum cut size scales accordingly:

$$\lambda = \text{max}_{\text{cut}}(p) = \frac{n^2}{4}p. \tag{8}$$

This is how the penalty parameter λ was estimated also in article [36]. It is worth mentioning that λ-values are dependent on graph structure, specifically on the number of nodes n and edge-probability p, and we should denote it as $\lambda_{n,p}$. However, for simplicity, we kept the notation as λ throughout the article.

The penalty parameter from Equation (8) assumes that all nodes are interconnected up to the edge-probability factor p, making the balance constraint excessively strong compared to the objective function. Real-world graphs, are rarely fully connected, and most nodes have significantly fewer connections than the theoretical maximum. Consequently, the actual expected max-cut size is much smaller than what Equation (8) suggests. This results in an overly restrictive λ, which places a disproportionately high emphasis on balance and causes the solver to neglect optimizing the actual graph partitioning objective. A more refined penalty estimation is required to guide the QA HS in finding the optimal solution, ensuring that both the objective function is satisfied and the penalty term is preserved [38].

Our preliminary tests on approximately 50 random generated graphs with $100 \leq n \leq 4000$ nodes have confirmed that using the proposed λ from Equation (8) used as penalty parameter often prevented the solver from finding valid solutions satisfying the objective function (solution found in 77.93% cases, QA HS outperformed used classical methods in 72.71%).

By considering the expected number of inter-edges and the benefit of moving nodes between sets rather than the theoretical maximum, a refined selection of λ can significantly improve solver's accuracy.

In the next sections we will provide an overview of λ parameter calculation to achieve better results and also collect enough data to be able use machine learning-based model to predict λ parameter values dynamically with even better solution quality.

IV. PENALTY PARAMETER CALCULATION

To refine the estimation of penalty parameter λ, we analyzed worst-case scenarios in terms of how the objective function and penalty term change when the node assignment changes-the node is moved from one set to another.

It is important to note that the edge probability factor p was not included in our estimation. This decision was made because p is specific to the Erdős–Rényi model and is generally unknown or undefined for arbitrary graph instances.

To establish a lower bound for λ, we considered the case where all nodes initially belong to S_1 ($x_i = 1, \forall i \in V$). The objective function in this case is:

$$E_{\text{cut}}(\mathbf{x}) = 0 \tag{9}$$

and the penalty term is:

$$E_{\text{balance}}(\mathbf{x}) = \lambda \left(\frac{n}{2}\right)^2. \tag{10}$$

By moving one node from S_1 to S_0 to improve balance, assuming the node is connected to all other nodes in S_1, the objective function becomes:

$$E_{\text{cut}}(\mathbf{x}) = 1 - n \tag{11}$$

and the penalty term changes to:

$$E_{\text{balance}}(\mathbf{x}) = \lambda \left(\frac{n}{2} - 1\right)^2. \tag{12}$$

Extending this to moving x nodes from S_1 to S_0, we have:

$$E_{\text{cut}}(\mathbf{x}) = x(x - n) \tag{13}$$

$$E_{\text{balance}}(\mathbf{x}) = \lambda \left(\frac{n}{2} - x\right)^2. \tag{14}$$

To ensure the penalty term dominates and is not neglected in favor of the objective function we use the derivations:

$$\frac{d}{dx}\left[x(x - n)\right] = 2x - n \tag{15}$$

$$\frac{d}{dx}\left[\lambda \left(\frac{n}{2} - x\right)^2\right] = \lambda(2x - n). \tag{16}$$

Hence the function (14) grows equally or faster than function (13) if:

$$\lambda(2x - n) \geq 2x - n, \tag{17}$$

which simplifies to:

$$\lambda \geq 1, \tag{18}$$

and it is setting the lower bound of the penalty parameter λ.

To establish the upper bound for λ, we considered the scenario where the partition is perfectly balanced, meaning $|S_0| = |S_1|$. In this case, the penalty term $E_{\text{balance}}(\mathbf{x})$ is minimized and equals to zero, and the objective function is entirely dependent on the number of inter-edges.

In the worst-case scenario, every node in subset S_0 is connected to every node in subset S_1 which in fully connected graph will be exactly the same as in Equation (7).

Next we tried to improve the objective function by moving a single node from S_1 to S_0. This change of node assignment has two effects:

- The penalty term $E_{\text{balance}}(\mathbf{x})$ increases by λ because the partition sizes become slightly unbalanced.
- The objective function $E_{\text{cut}}(\mathbf{x})$ increases by $\frac{n}{2} - 1$, assuming the moved node was well connected to nodes in S_1.

If we refine this assumption further, and instead of considering a fully connected graph, we take into account the actual graph structure where the maximum graph degree $\max(\deg(G))$ is known (similar assumption in [15]), the increase in $E_{\text{cut}}(\mathbf{x})$ is constrained by:

$$E_{\text{cut}}(\mathbf{x}) \leq \min\left(\max(\deg(G)), \frac{n}{2} - 1\right). \tag{19}$$

To prevent the penalty term from dominating the objective function and distorting the optimization process, we imposed the condition:

$$E_{\text{balance}}(\mathbf{x}) \leq E_{\text{cut}}(\mathbf{x}). \tag{20}$$

Substituting our refined estimates, we obtained:

$$\lambda \leq \min\left(\max(\deg(G)), \frac{n}{2} - 1\right). \tag{21}$$

After we calculated the lower (18) and upper (21) bound for λ, we were able to establish an interval:

$$1 \leq \lambda \leq \min\left(\max(\deg(G)), \frac{n}{2} - 1\right). \tag{22}$$

To determine an estimated value for λ, we took the midpoint of this interval:

$$\lambda_{\text{est}} = \frac{1 + \min\left(\max(\deg(G)), \frac{n}{2} - 1\right)}{2}. \tag{23}$$

This ensures that penalty parameter is neither too small, which would weaken the balance constraint, nor too large, which would overemphasize balance at the expense of minimizing the cut size.

We followed the process shown in Fig. 1 and used the newly established penalty parameter λ_{est} in the matrix $\mathbf{Q}_{G,\lambda}$. However, after further testing on approximately 70 randomly generated graphs with $100 \leq n \leq 4000$ nodes, we consistently encountered some instances of MBP where no valid solution was found (solution found in 92.03% cases, QA HS outperformed used classical methods in 90.20%). The penalty term was still overly dominant, causing the optimization process to prioritize balance enforcement at the expense of the objective function. As a result, the algorithm failed to sufficiently minimize the number of inter-edges, and in many cases, the balance condition was never satisfied, preventing the solver from reaching a feasible solution to MBP.

In the analysis phase of the process, we discovered that the penalty parameter λ must be even more scaled down for larger graphs. The adjusted lambda parameters' multipliers λ_{mult} were selected as shown in Table I and the penalty parameter was set as:

$$\lambda = \lambda_{est} \cdot \lambda_{mult}. \qquad (24)$$

The reason behind different λ_{mult} selections is that, large graphs tend to have higher variability in edge density, meaning a strict balance constraint with a large penalty parameter could dominate the objective function leading to suboptimal partitions. Mathematically, the penalty term grows quadratically as $\mathcal{O}(n^2)$, while the objective function scales with the number of edges, which typically follows $\mathcal{O}(n)$ in sparse graphs. If λ is not reduced accordingly, the solver prioritizes partition balance, making it difficult to minimize the cut size effectively.

After performing the next round of testing on 607 randomly generated graphs with recalculated λ-values, set as in Equation (24), we achieved results better than classical methods in 98.53% and the QA HS found the solution in $\mathbf{100.00}\%$ cases.

The results shown in Fig. 2 highlight the significant advantage of QA HS over the PyMetis and Kernighan-Lin algorithms, demonstrating its superior performance in terms of minimizing inter-edges. While Kernighan-Lin did not provide significant additional insights as a benchmark, it was still useful in comparing the performance against quantum annealing. Based on this, we excluded it from further testing.

Additionally, the QA HS inter-edges values were selected based on the best results for the given penalty parameter multipliers from the λ_{mult} values.

In cases where multiple λ_{mult} values led to the same minimal inter-edges, we chose the minimum λ_{min} and maximum λ_{max} penalty parameter multipliers' values. These values together formed a range $[\lambda_{min}, \lambda_{max}]$ representing an interval of values within which the penalty parameter multipliers should lie to achieve optimal partitioning and were later used as regressors in the machine learning-based prediction model.

To illustrate the relationship between the number of nodes and the λ_{mult}-parameter ranges, we present the candle plot in Fig. 3.

Although the achieved results were satisfying, it was necessary to run the QA HS for each λ_{mult} value from the set defined in Table I and evaluate the solution quality to decide the $[\lambda_{min}, \lambda_{max}]$ ranges.

TABLE I
SELECTED λ_{mult} VALUES FOR DIFFERENT GRAPH SIZES BASED ON ANALYSES OF COLLECTED DATA FROM PREVIOUS TESTING.

Number of Nodes (n)	Selected λ_{mult} values
100, 200	$\{0.05, 0.1, 0.2, 0.4\}$
300, 400, 500	$\{0.005, 0.01, 0.03, 0.05, 0.1, 0.2\}$
600, 700, 800, 900	$\{0.005, 0.01, 0.03, 0.05, 0.1\}$
1000, 1200, 1400	$\{0.002, 0.005, 0.01, 0.03, 0.05, 0.1\}$
1600, 1800, 2000	$\{0.002, 0.005, 0.01, 0.03, 0.05, 0.1\}$
2500, 3000, 3500, 4000	$\{0.0005, 0.001, 0.002, 0.005, \}$ $\{0.01, 0.03, 0.05, 0.1\}$

Fig. 2. Comparison of Kernighan-Lin, PyMetis, and QA HS inter-edges results across different numbers of nodes. For graphs with $100 \leq n \leq 1000$ nodes. We lacked sufficient data for $n > 1000$ nodes for the Kernighan-Lin algorithm due to its extended computational time.
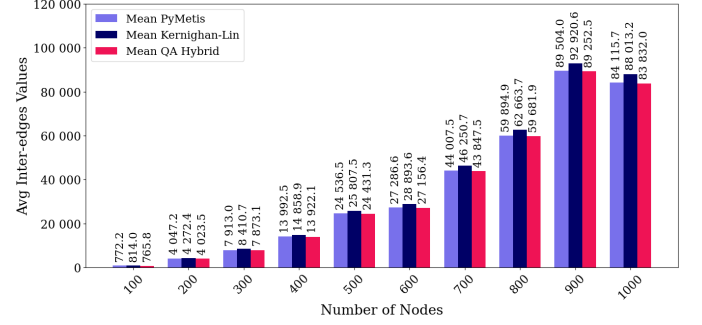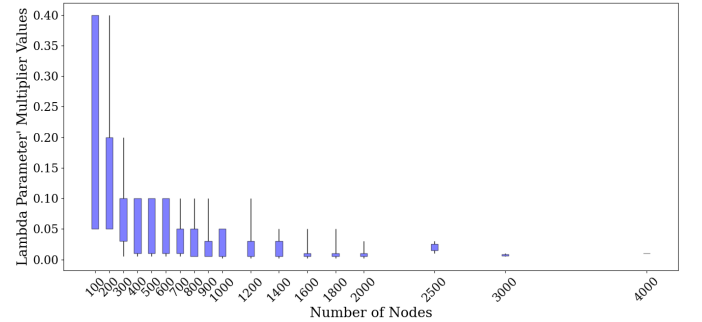

Fig. 3. Lambda parameter's multipliers ranges based on number of nodes, derived from tested data on 607 random regerated Erdős–Rényi graphs.

To further refine our approach and optimize penalty parameter selection, we leveraged the collected data from our testing. Using the dataset generated from our evaluations, we developed a machine learning-based prediction model to predict optimal penalty parameter values dynamically.

V. MACHINE LEARNING-BASED PARAMETER TUNING

Initially, we considered several graph properties as independent variables, including: number of nodes (n), number of edges ($|E|$), density ($\rho = \frac{2|E|}{n(n-1)}$), graph modularity and community structure.

Through simple regression analysis, we found out that number of edges and density are highly correlated. Thus, including both did not provide additional predictive advantage. Additionally, modularity and community detection [39] were considered, but these are computationally as hard as the MBP itself, making them impractical for our use case. This led to the selection of number of nodes and density as regressors.

Before implementing Gradient Boosting Regressor (GBR) machine learning-based model for penalty parameter λ prediction, we conducted an analysis to determine how often specific λ_{mult} parameter multipliers from Table I led to the minimum inter-edges for MBP. The primary objective was to evaluate the success rate of the QA HS under different graph configurations and chosen λ_{mult} values, where the used penalty parameter is than given as in Equation (24).

Fig. 4 presents a heatmap visualization of the QA HS success rates across various numbers of nodes and graph densities. The color intensity represents the probability that, under a given choice of the lambda parameter multiplier λ_{mult}, the solver achieved an optimal partitioning. The success rate was computed as the proportion of test cases where the QA HS found optimal solution (partition with minimal inter-edges) given the manually selected λ_{mult} values. From the visualization, we observe the following:

- The QA HS achieved higher success rates at moderate graph densities $(0.25-0.75)$, whereas very sparse or very dense graphs resulted in lower success rates.
- Larger graphs $(n \geq 1000)$ displayed relatively stable success rates, indicating that the solver scales well when lambda is properly chosen.
- The heatmap highlights the dependency of solver performance on density, suggesting that selecting lambda adaptively based on graph structure is critical.

This analysis underscores the importance of tuning solver parameters dynamically rather than relying on fixed calculated values. It also provided motivation for implementing machine learning-based penalty parameter selection, which further optimized the solver's success rate beyond these preliminary results.

To optimize λ selection further, we trained GBR, machine-learning-based model, to predict the lower and upper bound of the penalty parameter multipliers range $[\lambda_{\text{min}}, \lambda_{\text{max}}]$, see Algorithm 2.

GBR model was chosen because it effectively captures nonlinear relationships in the data and can handle a mixture of continuous and categorical input features without requiring extensive feature engineering. Additionally, it has demonstrated high predictive accuracy in similar optimization scenarios [40].
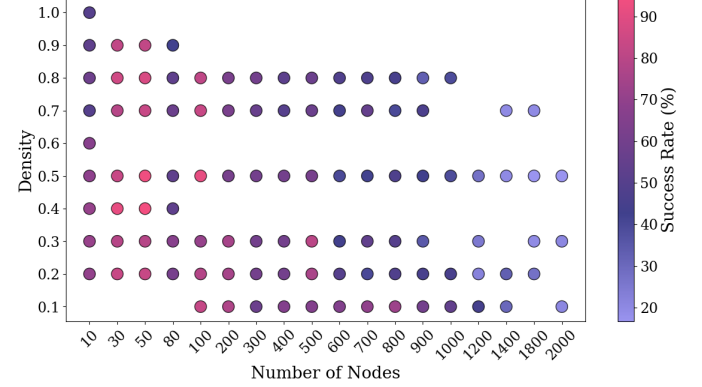
The model was trained using:

- Features: number of nodes n, graph density ρ, and λ_{est}.
- Targets: minimum and maximum lambda parameters multiplier $(gbr_{\text{min}}, gbr_{\text{max}})$ for which the QA HS produced optimal solution.

After training, the model was evaluated using a test dataset obtained by splitting the original data (80/20).

The GBR model demonstrated high effectiveness, Fig. 5, in approximating the bounds of λ values. For the upper bound λ_{max}, the model achieved an R^2 score of 0.9028, with a low RMSE of 0.0360 and MAE of 0.0188, indicating strong predictive power and reliability. For the lower bound λ_{min}, the model attained an R^2 score of 0.4489, with MAE of 0.0124 and RMSE of 0.0211, suggesting moderate explanatory power.

These results are consistent with the observed distributions of λ_{min} and λ_{max} across graph instances, see Fig. 6, where λ_{min}
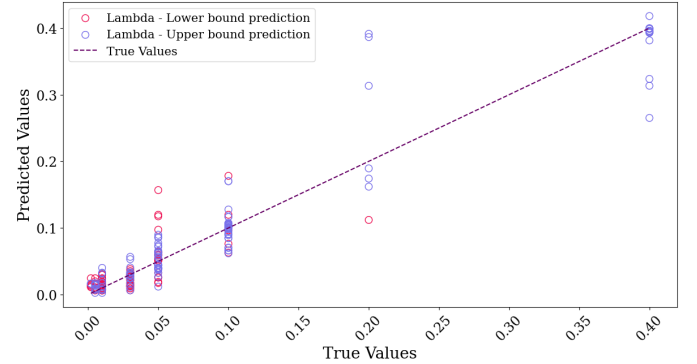

Fig. 4. Success Rate dependence on chosen lambda parameter's multipliers for graphs with different densities and number of nodes. The color intensity represents the probability of achieving an optimal partition, with higher success rates observed for low and moderate densities.

Algorithm 2 Training the GBR model

1: Extract collected graph properties and partitioning results (λ_{min} and λ_{max}) from database.
2: Compute features: n, ρ, and λ_{est}.
3: Train two Gradient Boosting Regressors ($gbr_{\text{min}}, gbr_{\text{max}}$) to predict λ_{min} and λ_{max}.
4: Evaluate models using RMSE, MAE, R^2 score.
5: Save trained models for future predictions.

Fig. 5. Comparison of predicted penalty parameter multipliers' bounds $gbr_{\text{min}}, gbr_{\text{max}}$ with true values using GBR. The dashed line represents the ideal perfect fit.

values are highly concentrated in a narrow range—making it inherently more difficult for the model to explain variance—while λ_{max} values show a broader and more structured distribution. It is worth noting that the GBR model was used with default parameters, without any hyperparameter tuning. As the results were satisfactory for our needs, no additional predictive models were tested.
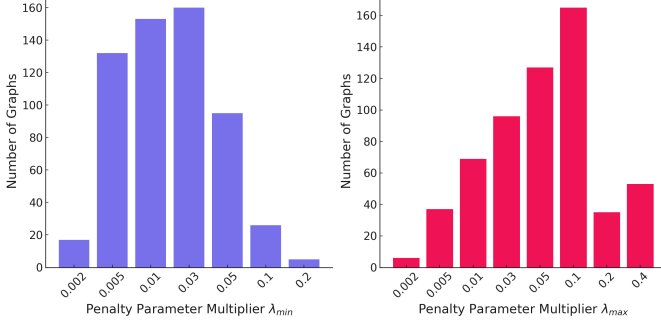
The final predicted values for penalty parameter λ were obtained by combining the lower and upper bound predictions

$$\lambda = \lambda_{\text{est}} \frac{\lambda_{\text{min}} + \lambda_{\text{max}}}{2}, \qquad (25)$$

as shown in Algorithm 3.

The predicted penalty parameter λ was then used to construct QUBO matrix $\mathbf{Q}_{G,\lambda}$, which was then provided as the input to the QA HS for further testing.

Fig. 6. Distribution of predicted $\lambda_{\min}$ and $\lambda_{\max}$ values across graph instances used for training the GBR model. The $\lambda_{\min}$ values are concentrated in a narrow range, whereas $\lambda_{\max}$ values are more broadly distributed, contributing to the difference in model performance as seen in the R^2 scores.

Algorithm 3 Predicting lambda for new graphs

1: Compute $\lambda_{\text{est}} = \frac{1+\min(n/2-1,\max(\deg(G)))}{2}$ for new graph $G(n,p)$.
2: Construct feature vector $f = (n, \rho, \lambda_{\text{est}})$.
3: Predict $\lambda_{\min}$ and $\lambda_{\max}$ using trained regressors $(gbr_{\min}, gbr_{\max})$.
4: Compute final lambda as $\lambda = \lambda_{\text{est}} \frac{\lambda_{\min} + \lambda_{\max}}{2}$.

VI. SIMULATION RESULTS

In this section, we present the results of our simulations after GBR penalty parameter tuning using QA HS for graphs with $100 \leq n \leq 4000$ nodes, as well as QPU for graphs with $n \leq 100$ nodes. At the end of the section, we summarize the advantages of GBR-based penalty parameter tuning, highlighting it as a novel approach for improving solution quality for optimization problems solved using D-Wave Systems' quantum anealers.

All computations, including quantum annealing solver execution and classical graph partitioning, were programmed using Python. The results and experiment metadata were stored in a MariaDB relational database for further analysis.

The classical computations were executed on a dedicated Linux VPS server equipped with an Intel Core Processor (Haswell, no TSX, IBRS), featuring 3 CPU cores running at 2.6 GHz and 6 GB of RAM, operating on Ubuntu Server 24 LTS (64-bit). This setup ensured a consistent and reliable computational environment for benchmarking classical algorithms against the quantum approach.

A. Hybrid Solver Testing

To evaluate the QA HS after tuning λ with the GBR model, Equation (25), we used the optimized λ values to construct the QUBO matrices $\mathbf{Q}_{G,\lambda}$ for new 126 random generated graphs and compared QA HS against PyMetis in terms of solution quality. Additionally, we calculated the percentage of cases in which PyMetis produced a balanced solution[8] and the percentage of cases where QA HS successfully found a valid solution - which is constantly 100%. A valid solution in QA

---

[8]In the Metis algorithm, the balance constraint is not enforced.

HS means that the subsets were balanced and that the solver was able to return a solution within the execution time limit.

Table II presents a comparative analysis of the performance of both solvers.

From the table, we observe that:

- PyMetis was outperformed by QA HS in 100% of cases, indicating that QA HS produced better partitions in every MBP instance.
- PyMetis provided balanced solution in approximately 50% of cases.
- The percentage difference fluctuates for smaller graphs but stabilizes at approximately $0,30 - 0,40\%$ for larger graphs.
- The absolute difference in inter-edges increases with the number of nodes, reaching a maximum of 1372 inter-edges at $n = 4000$.

Fig. 7 graphically represents the absolute and percentage differences in inter-edges between QA HS and PyMetis. The absolute difference grows exponentially as the number of nodes increases and stabilizes at $n = 2000$, whereas the percentage difference fluctuates for smaller graphs but stabilizes for larger graphs.

This analysis demonstrates that data-driven penalty parameter tuning enhances the effectiveness of quantum hybrid solvers for medium-sized instances of MBP.

In addition to partition quality, we also analyzed computational time. For classical algorithms, this refers to the total execution time of the partitioning process, whereas for the QA HS, we accounted for additional factors. Since used QA HS is a cloud-based solver with hybrid processing, its runtime includes multiple components beyond annealing time, such as embedding, preprocessing, quantum-classical post-processing [41], network latency as well as the QUBO matrix calculation. This must be considered when interpreting computational efficiency, as the response time includes both quantum and classical components. Here we measured the overall computational time and QUBO matrix separately.

The QUBO matrix calculations were executed on a classical computer. This step is computationally demanding due to the iteration process for large number of nodes, see Algorithm 1.

On average, the execution time of PyMetis was three times faster than the overall computational time of QA HS for graphs with $n \geq 1000$ nodes. For smaller graphs, PyMetis showed an even greater performance advantage. The QUBO matrix calculation was not optimized and grew significantly with the number of nodes, taking 163 seconds for graphs with 4000 nodes, but the matrix calculation optimization is outside the scope of this article. It is also important to note that its performance as well as the PyMetis execution time is highly dependent on the properties of the classical computer used.

B. QPU testing

To evaluate the performance of a fully quantum approach, we conducted tests using the D-Wave's QPU of D-Wave Advantage Systems directly via Ocean SDK without any annealer's parameter settings. As a benchmark we used again PyMetis. Unlike the QA HS, which utilizes both classical and
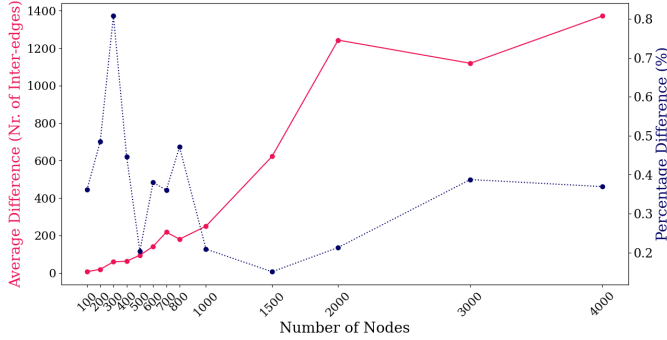
TABLE II
COMPARISON OF QA HS AND PyMetis ON SOLVING MBP AFTER λ PENALTY PARAMETER TUNING USING GBP

Nr. of Nodes (n)	Avg density	PyMetis balanced (%)	QA HS better (%)	Abs[a] diff	Perc[b] diff %
100	0.715	66.7	**100**	6.0	0.361
200	0.438	40.0	**100**	19.2	0.485
300	0.360	66.7	**100**	58.7	0.808
400	0.379	57.1	**100**	62.4	0.446
500	0.766	0.0	**100**	94.3	0.204
600	0.437	75.0	**100**	141.0	0.380
700	0.521	80.0	**100**	217.6	0.360
800	0.259	60.0	**100**	180.0	0.471
1000	0.495	36.4	**100**	249.4	0.209
1500	0.750	0.0	**100**	622.0	0.151
2000	0.600	0.0	**100**	1243.0	0.213
3000	0.138	25.0	**100**	1119.3	0.388
4000	0.100	100.0	**100**	1372.0	0.369

[a] |avg(qa_hs_inter_edges) - avg(pymetis_inter_edges)|

[b] $\frac{\text{avg(pymetis\_inter\_edges)} - \text{avg(qa\_hs\_inter\_edges)}}{\text{avg(qa\_hs\_inter\_edges)}} \cdot 100\%$

Fig. 7. Comparison of QA and PyMetis Absolute and Percentage differences of inter-edges. The absolute difference (red solid line) shows an increasing trend, while the percentage difference (blue dotted line) fluctuates but stabilizes for larger graphs.
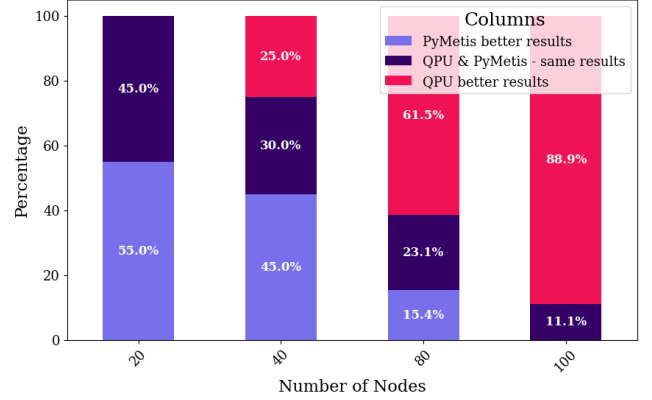

quantum resources, the QPU-only method is limited by the number of available qubits and requires embedding the graph problem into the physical quantum architecture [35].

We selected a set of 70 random generated Erdős–Rényi graphs with up to 100 nodes to ensure compatibility with the QPU's connectivity constraints. D-Wave Advantage Systems with Pegasus topology operates on 5,640 qubits, each coupled to up to 15 other qubits. This connectivity allows the embedding of complete graphs of up to 119 vertices with a chain length of 17 [42]. It's important to note that the embeddable size is not solely determined by the number of vertices but also by the graph's density and connectivity. [43].

From Fig. 8 we observe:

- For small graphs (20-40 nodes), PyMetis outperforms QPU.
- For medium-sized graphs (60-100 nodes), QPU achieves better partitions more frequently, with 88.9% success rate at 100 nodes.
- The transition point where QPU starts performing comparably to PyMetis occurs around 80 nodes. This can be caused by the penalty parameter λ values determined by the GBR algorithm, as the training data for GBR was

Fig. 8. Comparison of QPU and PyMetis partitioning results on graphs with up to 100 nodes. The penalty parameter λ was determined using the GBR algorithm.

derived from graphs with $n \geq 100$ nodes, potentially leading to suboptimal results for smaller graphs.

Despite the promising results at larger graph sizes, fully quantum approach face several challenges that limit its practical usage for large-scale problems. One of the primary limitations is qubit connectivity constraints [4]. The QPU's limited connectivity requires minor-embedding techniques to map logical problem variables onto physical qubits [43]. This embedding process introduces chain lengths, where logical variables are represented by multiple physical qubits, thereby reducing the number of independent variables available for computation. This leads to the need for chain strength parameter tuning, making optimization more challenging. Another significant factor affecting QPU performance is error sensitivity and annealing schedule optimization [44]. Unlike hybrid solvers, QPU-only execution is highly sensitive to the configuration of the annealing schedule, meaning incorrect parameter settings can significantly degrade performance. Tuning annealing parameters such as anneal offsets and flux bias correction plays a crucial role in obtaining optimal partitions [45], but is outside the scope of this article.

Given these constraints, hybrid approaches may still be preferable for graphs with $n \geq 100$ nodes, as they balance the advantages of quantum computing with the stability and efficiency of classical methods.

C. Summary

The evaluation of QA HS and QPU for solving the MBP yielded several key insights:

- **Adaptive penalty parameter selection:** The initial theoretical estimation of the penalty parameter λ, based on the max-cut approach, was overly restrictive and often prevented the solver from finding valid solutions. This limitation was addressed by refining the penalty parameter selection process, which involved calculating both lower and upper bounds for λ and adjusting it based on graph characteristics such as number of nodes and maximum graph degree. The improved approach significantly enhanced the accuracy of the solver, enabling it to find

optimal solutions more effectively, but there were still instances of MBP for which the QA HS was struggling to find any valid solution. The penalty parameter multiplier values (λ_{mult}) were introduced, to scale the penalty parameter down. After the tuning we achieved better results and created bases for GBR parameter prediction.

- **Machine learning-based parameter tuning:** To further optimize the penalty parameter, a GBR model was trained on key graph properties (number of nodes and density) and specific λ_{mult} values ($\lambda_{\min}, \lambda_{\max}$) to predict optimal λ values. This data-driven tuning improved the solver's performance by allowing it to dynamically adjust the penalty parameter based on the specific characteristics of each graph. The GBR model successfully minimized inter-edges while ensuring that the partitions remained balanced, leading to high-quality solutions.

- **Superior accuracy of QA HS over used classical methods:** After implementing penalty parameter tuning, the QA HS consistently found valid solutions across all test cases and produced better partitions than PyMetis in 100% of the MBP instances. Although PyMetis was computationally faster, it struggled to maintain balance in approximately 50% of the cases, particularly for larger graphs. The accuracy gap between QA HS and PyMetis widened as the graph size increased, demonstrating the scalability and effectiveness of the hybrid quantum-classical approach in handling larger, more complex problems.

- **Fully quantum execution on small-sized graphs:** Direct execution on D-Wave's QPU showed improved results for larger small-scale graphs ($n \geq 80$), potentially due to the penalty parameter λ being predicted using trained data from larger graph instances.

- **Hybrid quantum-classical scalability and flexibility:** The hybrid solver's ability to combine quantum annealing with classical optimization techniques provides flexibility in solving combinatorial optimization problems at different scales. As shown in the study, the hybrid approach scaled better than classical solvers for larger instances of MBP.

- **Improvement in solution quality:** Table III clearly demonstrates the effectiveness of adaptive and data-driven penalty parameter tuning strategies. As the selection of λ becomes more informed—from basic estimation to machine learning-based predictions—the success rate of finding solutions and the advantage of the QA HS significantly improve. This highlights the critical role of dynamic penalty parameter tuning in optimizing QA HS performance.

VII. Practical Applications and Future Directions

Beyond academic experimentation, these findings have potential in real-world applications, particularly in intelligent transportation systems, network optimization, and other large-scale combinatorial problems. One promising use case is in map segmentation, where dynamic road networks can be

TABLE III
Overall performance of QA HS partitioning across different λ selection strategies

Strategy for setting λ	Solution Found [%]	Hybrid Better [%]
$\lambda = \max_{\text{cut}}(p)$	77.93	72.71
$\lambda = \lambda_{\text{est}}$	92.03	90.20
$\lambda = \lambda_{\text{est}} \cdot \lambda_{\text{mult}}$	**100.00**	98.53
λ predicted via GBR	**100.00**	**100.00**

partitioned efficiently to optimize computational and communication overhead. In transportation and mobility planning, graph partitioning can enhance route optimization and traffic flow management. Additionally, in distributed computing and data clustering, dynamically adjusting partitioning based on graph properties such as density and node connectivity enables efficient workload distribution and parallel processing.

MBP plays crucial role also in machine learning, particularly in semi-supervised learning and neural network compression, where large-scale data structures need to be divided into smaller, manageable components to improve efficiency. In image processing, balanced graph partitioning aids in image segmentation, where regions with similar features are grouped together for object detection and recognition. Moreover, in social network analysis, such as X or Facebook, optimized partitioning techniques can help detect communities, improve recommendation systems, and enhance information propagation models. The ability to dynamically optimize partitions across these diverse fields demonstrates the broad applicability and impact of QA-based partitioning methods.

While this study primarily focuses on the MBP, its findings open the door for broader applications of QA-based solvers in complex optimization scenarios. One potential extension is the adaptation of QA HS for the k-partitioning problem. This study focuses on bi-partitioning, but the methodology can be extended recursively to partition a graph into k subsets. By iteratively applying the bi-partitioning approach and refining partitions in subsequent steps, larger and more complex partitioning tasks can be handled efficiently.

A key area of our future research is the application of QA HS in the domain of autonomous mobility, particularly in navigation. QA could be leveraged to predict optimal lane-changing maneuvers based on real-time sensor data in multi-lane road environments. Also by (dynamically) partitioning road segments and traffic data, the system could reduce computational complexity in V2V communication, enabling more efficient information exchange between vehicles. This approach could also enhance route optimization and planning by evaluating individual road segments, identifying congestion hotspots, and recommending alternative routes in real-time. These advancements would contribute to more adaptive and efficient traffic management systems, ultimately improving safety and reducing travel times.

In our upcoming study, we will focus on an extended version of the MBP, the k-partitioning problem, and construct the QUBO matrix in terms of balanced partitions. Each subset will encode all possible routes for n vehicles, with the objective of identifying the subset that minimizes route overlaps.

This approach aims to reduce traffic congestion and improve navigation efficiency. Furthermore, penalty parameter tuning, which has proven to be effective in this study, could be applied in our future study and also beyond MBP. By refining penalty parameters in real-time, QA-based models could provide more efficient solutions to adaptive traffic control and route planning.

Ultimately, by combining both classical and quantum methods, practical large-scale optimization solutions can be achieved, paving the way for broader adoption of quantum annealing in real-world applications.

VIII. Conclusion

This paper provides a comprehensive evaluation of D-Wave Systems' quantum annealing solvers for solving the Minimum Bisection Problem, focusing on penalty parameter tuning to improve solution quality. We present a novel, adaptive penalty tuning strategy that combines classical and quantum optimization methods, showing that Gradient Boosting Regressor can predict optimal penalty parameter values based on graph properties such as number of nodes and graph density. Our results demonstrate that the quantum annealing hybrid solver with machine learning-based tuning outperforms classical solvers, Metis and Kernighan-Lin, achieving superior results in 100% of test cases across various graph sizes up to 4 000 nodes. The study highlights that dynamic penalty parameter selection plays a crucial role in significantly improving the solution quality of quantum annealing, positioning it as a viable alternative to classical partitioning algorithms for optimization problems.

All authors are with the Department of Computers and Informatics, Technical University of Kosice, 04200 Kosice, Slovakia. (e-mail: renata.rusnakova@tuke.sk; martin.chovanec@tuke.sk; juraj.gazda@tuke.sk)

References

[1] D. H. Shin, D. Qian, and J. Zhang, "Cascading effects in interdependent networks," *Network, IEEE*, vol. 28, pp. 82–87, 08 2014.

[2] M. Bastian, "Expected complexity of graph partitioning problems," *Discrete Applied Mathematics*, vol. 157, no. 2, pp. 193–212, 2009.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[4] F. A. Quinton, P. A. S. Myhr, M. Barani, P. C. del Granado, and H. Zhang, "Quantum annealing versus classical solvers: Applications, challenges and limitations for optimisation problems," *arXiv preprint*, 2024, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/2409.05542

[5] A. Buluc, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, "Recent advances in graph partitioning," *arXiv preprint*, 2015, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/1311.3144

[6] P. Sanders and C. Schulz, "Engineering multilevel graph partitioning algorithms," *arXiv preprint*, 2011, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/1012.0006

[7] S. Barnard and H. Simon, "A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems," in *Proceedings of the 1993 ACM/IEEE Conference on Supercomputing (SC '93)*. ACM, 1993, pp. 711–718.

[8] B. Hendrickson and R. Leland, "A multi-level algorithm for partitioning graphs," in *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (SC '95)*. IEEE, 1995, pp. 28–28.

[9] P. Sanders and C. Schulz, "Kahip v3.00 – karlsruhe high quality partitioning – user guide," 2020, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/1311.1714

[10] Y. Akhremtsev, P. Sanders, and C. Schulz, "High-quality shared-memory graph partitioning," *arXiv preprint*, 2018, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/1710.08231

[11] G. Karypis and V. Kumar, "A fast and high-quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[12] ——, "Metis—a software package for partitioning unstructured graphs, partitioning meshes and computing fill-reducing ordering of sparse matrices," Department of Computer Science, University of Minnesota, Tech. Rep. TR-97-031, January 1997, accessed: 2025-03-07. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/fetch/papers/metis/manual.pdf

[13] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, 1970.

[14] S. V. Patil and D. B. Kulkarni, "Graph partitioning using heuristic kernighan-lin algorithm for parallel computing," in *Next Generation Information Processing System*, P. Deshpande, A. Abraham, B. Iyer, and K. Ma, Eds. Singapore: Springer Singapore, 2021, pp. 281–288, accessed: 2025-03-07.

[15] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, pp. 1–15, 2014. [Online]. Available: http://dx.doi.org/10.3389/fphy.2014.00005

[16] F. Glover, G. Kochenberger, and Y. Du, "Quantum bridge analytics i: A tutorial on formulating and using qubo models," *4OR*, vol. 17, pp. 1–20, December 2019.

[17] D-Wave Systems, Inc., *Hybrid Solvers Handbook*, 2025, accessed: 2025-03-18. [Online]. Available: https://docs.dwavesys.com/docs/latest/handbook_hybrid.html

[18] J. Raymond, R. Stevanovic, W. Bernoudy, K. Boothby, C. C. McGeoch, A. J. Berkley, P. Farré, J. Pasvolsky, and A. D. King, "Hybrid quantum annealing for larger-than-qpu lattice-structured problems," *ACM Transactions on Quantum Computing*, vol. 4, no. 3, pp. 1–30, April 2023. [Online]. Available: http://dx.doi.org/10.1145/3579368

[19] M. Fernández-Campoamor, C. O'Meara, G. Cortiana, V. Peric, and J. Bernabé-Moreno, "Community detection in electrical grids using quantum annealing," 2021, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/2112.08300

[20] E. Pelofske, G. Hahn, and H. N. Djidjev, "Reducing quantum annealing biases for solving the graph partitioning problem," in *Proceedings of the 18th ACM International Conference on Computing Frontiers*. ACM, May 2021, pp. 133–139. [Online]. Available: http://dx.doi.org/10.1145/3457388.3458672

[21] H. Ushijima-Mwesigwa, C. F. A. Negre, and S. M. Mniszewski, "Graph partitioning using quantum annealing on the d-wave system," *arXiv preprint*, 2017, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/1705.03082

[22] C. Hartmann, J. Zhang, C. D. Gonzalez Calaza, T. Pesch, K. Michielsen, and A. Benigni, "Quantum annealing based power grid partitioning for parallel simulation," *arXiv*, 2025. [Online]. Available: https://arxiv.org/abs/2408.04097

[23] H. Djidjev, G. Chapuis, G. G. Guerrini, M. I. Gluza, M. Moradi, and M. Udrescu, "Efficient combinatorial optimization using quantum annealing: A case study on graph partitioning," *Quantum Information Processing*, vol. 17, no. 9, pp. 1–24, 2018.

[24] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[25] M. Karpinski, "Approximability of the minimum bisection problem: An algorithmic challenge," in *Mathematical Foundations of Computer Science 2002*, K. Diks and W. Rytter, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 59–67.

[26] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M.

Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[27] F. Barahona, "On the computational complexity of ising spin glass models," *Journal of Physics A: Mathematical and General*, vol. 15, no. 10, pp. 3241–3253, October 1982. [Online]. Available: https://dx.doi.org/10.1088/0305-4470/15/10/028

[28] T. Leleu, Y. Yamamoto, S. Utsunomiya, and K. Aihara, "Destabilization of local minima in analog spin systems by correction of amplitude heterogeneity," *Physical Review Letters*, vol. 122, p. 040607, 2019.

[29] W. Xuan, Z. Zhao, L. Fan, and Z. Han, "Lagrangian relaxation based parallelized quantum annealing and its application in network function virtualization," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 4260–4274, 2024.

[30] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Physical Review E*, vol. 58, no. 5, pp. 5355–5363, 1998.

[31] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, p. 015002, 2018. [Online]. Available: https://doi.org/10.1103/RevModPhys.90.015002

[32] V. Choi, "Minor-embedding in adiabatic quantum computation: I. the parameter setting problem," *arXiv preprint*, 2008, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/0804.4884

[33] J. Raymond, N. Ndiaye, G. Rayaprolu, and A. D. King, "Improving performance of logical qubits by parameter tuning and topology compensation," in *Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, October 2020, pp. 295–305. [Online]. Available: https://doi.org/10.1109/QCE49297.2020.00044

[34] N. Mohseni, P. McMahon, and T. Byrnes, "Ising machines as hardware solvers of combinatorial optimization problems," *Nature Reviews Physics*, vol. 4, pp. 363–379, June 2022, accepted: 28 February 2022, Published: 04 May 2022. [Online]. Available: https://doi.org/10.1038/s42254-022-00440-8

[35] D-Wave Systems, Inc., *QPU Solvers: Minor-Embedding – D-Wave System Documentation*, 2024, accessed: 2024-02-07. [Online]. Available: https://docs.dwavesys.com/docs/latest/handbook_embedding.html

[36] ——, *Collection of Examples - Graph Partitioning*, 2024, accessed: 2024-02-07. [Online]. Available: https://cloud.dwavesys.com/leap/examples/222052595

[37] A. Kloeckner, M. Wala, N. Hartland, A. Danial, F. Obermeyer, C. Wu, and C. Gohlke, "Pymetis: A python interface to metis," July 2022, accessed: 2025-03-07. [Online]. Available: https://github.com/inducer/pymetis

[38] A. Verma and M. Lewis, "Penalty and partitioning techniques to improve performance of qubo solvers," *Discrete Optimization*, vol. 44, p. 100594, 2022, quadratic Combinatorial Optimization Problems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1572528620300281

[39] S. Fortunato and C. Castellano, "Community structure in graphs," in *Computational Complexity: Theory, Techniques, and Applications*, R. A. Meyers, Ed. New York, NY: Springer New York, 2012, pp. 490–512. [Online]. Available: https://doi.org/10.1007/978-1-4614-1800-9_33

[40] C. Zheng, J. Zhang, G. Liu, Z. Song, Z. Xu, H. Jin, and X. Li, "Customized routing optimization based on gradient boosting regressor model," *arXiv preprint arXiv:1710.11118*, 2017. [Online]. Available: https://arxiv.org/abs/1710.11118

[41] D-Wave Systems, Inc., *QPU Timing: Understanding Quantum Processing Unit Execution Time*, 2024, accessed: 2024-02-27. [Online]. Available: https://docs.dwavesys.com/docs/latest/c_qpu_timing.html

[42] ——, *Advantage Processor Overview*, 2021, accessed: 2025-03-07. [Online]. Available: https://www.dwavequantum.com/media/3xvdipcn/14-1058a-a_advantage_processor_overview.pdf

[43] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, "Next-generation topology of d-wave quantum processors," *arXiv preprint*, 2020, accessed: 2025-03-07. [Online]. Available: https://arxiv.org/abs/2003.00133

[44] O. Galindo and V. Kreinovich, "What is the optimal annealing schedule in quantum annealing," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 963–967. [Online]. Available: https://doi.org/10.1109/SSCI47909.2020.9283983

[45] A. Perdomo-Ortiz, B. O'Gorman, J. Fluegemann, R. Biswas, and V. N. Smelyanskiy, "Determination and correction of persistent biases in quantum annealers," *Scientific Reports*, vol. 6, p. 18628, 2016, accessed: 2025-03-25. [Online]. Available: https://www.nature.com/articles/srep18628

Renáta Runsáková received the M.S. degree in mathematical methods of information security from Charles University, Prague, Czech Republic, in 2012. Her master's thesis focused on computational complexity and reduction among NP-hard problems.

She is currently a Ph.D. candidate at the Technical University of Košice (TUKE), Slovakia, specializing in quantum computing and its applications in optimization problems. Her research interests include quantum annealing, combinatorial optimization, and quantum-inspired algorithms for mobility and cybersecurity applications.

Mgr. Rusnakova is actively involved in quantum computing research, contributing to various projects and publications in the field.

Martin Chovanec is a university lecturer and researcher in the field of information and cyber security at the Technical University of Košice, specifically at the Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics (FEI TUKE). His scientific research focuses on cyber security, with an emphasis on intrusion detection systems and network traffic analysis. His expertise includes the design of security architectures, risk management, and data protection in modern IT environments.

He is the author of several scientific publications contributing to the development of discourse in the field of information system security.

Assoc. Prof. Martin Chovanec, PhD. also works as the director of the Institute of Computer Technology at TUKE, where he contributes to the development of the university's IT infrastructure and the implementation of modern technologies in the management and security of information systems.

Juraj Gazda is currently a Vice-Rector for Innovation and Technology Transfer and Professor with the Faculty of Electrical Engineering at the Technical University of Košice (TUKE), Slovakia.

He has been a guest researcher at Ramon Llull University, Barcelona, and the Technical University of Hamburg-Harburg, and has been involved in development projects for Nokia Siemens Networks (NSN) and Ericsson. In 2017, he was recognized as the Best Young Scientist at TUKE.

Prof. Ing. Juraj Gazda PhD. currently serves on the executive board of IT Valley, an innovation ecosystem supporting collaboration between academia, industry, and public sector, and on the executive board of AI4Slovakia, a national initiative driving the adoption and strategic development of artificial intelligence in Slovakia. He is also an editor of the KSII Transactions on Internet and Information Systems and a guest editor for Wireless Communications and Mobile Computing (Wiley).