Utilizing Graph Sparsification for Pre-processing in Max Cut QUBO Solver*

Vorapong Suppakitpaisarn[1][0000−0002−7020−395X]
and Jin-Kao Hao[2][0000−0001−8813−4377]

[1] The University of Tokyo, Tokyo, Japan vorapong@is.s.u-tokyo.ac.jp
[2] University of Angers, Angers, France jin-kao.hao@univ-angers.fr

Abstract. We suggest employing graph sparsification as a pre-processing step for max cut programs using the QUBO solver. Quantum(-inspired) algorithms are recognized for their potential efficiency in handling quadratic unconstrained binary optimization (QUBO). Various meta-heuristic approaches, including those based on the Quantum Approximate Optimization Algorithm, have been suggested for addressing QUBO challenges in this context. Given that max cut is an NP-hard problem and can be readily expressed using QUBO, it stands out as an exemplary case to demonstrate the effectiveness of quantum(-inspired) QUBO approaches. Here, the non-zero count in the QUBO matrix corresponds to the graph's edge count. Given that many quantum(-inspired) solvers operate through cloud services, transmitting data for dense graphs can be costly. By introducing the graph sparsification method, we aim to mitigate these communication costs. Experimental results on classical and quantum-inspired solvers indicate that this approach substantially reduces communication overheads and yields an objective value close to the optimal solution.

Keywords: quantum-inspired optimization, max cut problem · pre-processing · graph sparsification · quadratic unconstrained binary optimization (QUBO)

1 Introduction

Quantum and quantum-inspired computing are considered to have the potential to enhance the efficiency of solving various computational problems [49, 22, 17]. Consequently, many meta-heuristics have been proposed for solving QUBO on both quantum and quantum-inspired computers [10]. These methods include those based on quantum annealing [47] and the Quantum Approximate Optimization Algorithm [39]. Given that several combinatorial and network optimization problems can be reformulated as QUBO, numerous researchers are

* We are grateful to the reviewers, Prof. Philippe Codognet, and Prof. Hiroshi Imai for their comments and suggestions, which helped us to improve the paper. This work was partially supported by the Japanese-French cooperation project JST SICORP Grant Number JPMJSC2208, Japan.

actively exploring the most proficient methods for addressing these optimization problems with the aid of quantum(-inspired) QUBO solvers [8, 9].

Researchers are particularly drawn to the maximum cut problem (max cut) [41, 30] because it is an NP-hard problem [26] that can be easily expressed within the QUBO framework [11, 46]. It has been observed that pre-processing the input before feeding it to QUBO solvers can yield good solutions more efficiently than using the original data directly. As a result, various studies have introduced pre-processing strategies specifically designed for the max cut problem to enhance the solution process [14, 13, 31].

Although minimizing computation time is important for solving the max cut problem, there is an additional challenge in addressing the problem with quantum or quantum-inspired QUBO solvers. Since quantum-inspired computers will not be commercially available for the next several decades, we are compelled to utilize these solvers through cloud services. This requires us to transmit our problems to the service providers, a step which often results in communication becoming a significant bottleneck [43, 28]. Therefore, our focus in this paper is on diminishing the costs associated with this communication.

1.1 Our Contributions

The communication cost of the max cut problem is strongly related to the number of edges in the input graph. We therefore propose to use the graph sparsification technique by the effective resistance edge sampling [25, 2, 50] to reduce the communication cost. The effective resistance technique has been demonstrated to significantly reduce the number of edges in a graph while preserving the cut size [50].

Let the symbol $|V|$ represent the total number of nodes in our input graph. Building upon the theoretical foundations presented in [50], we demonstrate that for any chosen $\epsilon > 0$, the outcome of our sampling method can yield a solution for the max cut problem that approximates within a factor of $1+\epsilon$. Simultaneously, this approach manages to decrease the edge count to $O\left(\frac{|V|\log|V|}{\epsilon^2}\right)$.

While a graph with $O\left(\frac{|V|\log|V|}{\epsilon^2}\right)$ edges is typically considered sparse in many applications, our experimental findings with both classical and quantum-inspired solvers demonstrate that setting the number of edges to fewer than $5|V|$ can still yield a viable approximate solution. Our study encompassed tests on 17 distinct networks, with node counts ranging from 100 to 12912 and edge numbers varying from 2124 to 807535. Moreover, these networks have a variety of topological structures. Remarkably, we have been able to reduce the number of edges – and consequently, the communication cost – by as much as 90%, while consistently achieving solutions where the cut size is at most 10% smaller than the maximum cut. Furthermore, we extracted subgraphs of varying sizes from two of the networks and verified that similar experimental outcomes are achievable in each of these subgraphs. This consistently suggests that our findings can be scaled up to larger networks, for which it is not feasible to upload all information to the cloud service.

We have also noticed a decrease in computation time when using classical QUBO solvers like Gurobi [20] on max cut instances where the edges have been sparsified using our sampling technique. For instance, while Gurobi could not complete the task on the original, dense max cut problem within two hours, it was able to finish in under two seconds after the sparsification has eliminated 90% of the edges. However, we do not consider this improvement as significant, because these solvers can still quickly find a reasonably good solution for both the original and the sparsified max cut instances. The reason Gurobi does not terminate with dense input graphs is primarily due to the extensive time required to prove the optimality of its solution.

As a pre-processing, our technique can benefit all solvers for max cut. The solvers which would have the biggest benefit from our pre-processing is the algorithm designed for addressing max cut on sparse graphs such as McSparse [7]. We believe that our pre-processing technique could improve the computation time of the McSparse algorithm, particularly when applied to dense graphs.

1.2 Related Works

The max cut problem has garnered widespread interest among researchers, leading to the development of numerous approximation and exact algorithms. Prominent among these are the well-known SDP relaxation algorithm [18, 37, 6] and algorithms for specific graph types [40, 19, 36, 48]. In this paper, however, our focus is not on the algorithms for solving the max cut problem itself, but rather on its pre-processing. Consequently, our algorithm is designed to be compatible with all these various algorithms.

As outlined in [45], several pre-processing techniques for QUBO solvers have been developed. Among the most significant are those based on autarkies and persistencies, which enable the determination of some binary variable values in the optimal solution [42, 21]. Additionally, there are methods that utilize the upper bound of the relaxed program to enhance solver efficiency [5, 12], as well as approaches centered around variable fixing [3]. These methods have been shown to yield smaller QUBO instances that can exactly solve the original problems. In contrast, our paper introduces a pre-processing technique aimed at generating approximate QUBO instances. Importantly, our approach is designed to be compatible with these existing pre-processing methods.

The graph sparsification by edge sampling technique has been introduced to give an efficient algorithm for the maximum flow problem and the sparsest cut problem [27]. Also, it has been used as a pre-processing of the maximum cut problem in [1]. The goal of using the sparsification in that paper is not to reduce the communication cost as in this paper but to increase the precision of publishing the maximum cut results under differential privacy. Consequently, the sampling technique in [1] is different from the effective resistance sampling, which we have used in this paper.

2 Preliminaries

2.1 Max Cut Problem

Consider a weighted graph (V, E, w), where V represents the set of nodes in the graph. The set of edges is denoted as $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$, and $w : E \to \mathbb{R}_{\geq 0}$ is the weight function assigning a non-negative weight $w(e)$ to each edge $e \in E$. A cut in graph G is defined as any subset $S \subseteq V$, with the weight of the cut S being $w_G(S) = \sum_{\{u,v\} \in E : u \in S, v \notin S} w(\{u, v\})$. The max cut problem aims to find the cut in G that has the highest weight.

2.2 QUBO Formulation for the Max Cut Problem

The quadratic unconstrained binary optimization (QUBO) is the following mathematical programming problem

$$\max \sum_u \sum_v Q_{u,v} x_u x_v$$

subject to $x_i \in \{0, 1\}$ for all i.

To express the max cut problem stated in the previous section using QUBO, we let $x_u = 1$ if $u \in S$ and $x_u = 0$ otherwise. Also, let $w'(\{u, v\}) = w(\{u, v\})$ when $\{u, v\} \in E$ and $w'(\{u, v\}) = 0$ otherwise. Since $x_u^2 = x_u$ when $x_u \in \{0, 1\}$, the weight of a cut S is then

$$\begin{aligned} w(S) &= \sum_{\{u,v\} \in E : x_u = 1, x_v = 0} w(\{u, v\}) = \sum_{\{u,v\} \in E} w(\{u, v\}) x_u (1 - x_v) \\ &= \sum_{u,v} w'(\{u, v\}) x_u (1 - x_v) \\ &= \sum_u \left[\sum_v w'(\{u, v\}) \right] x_u - \sum_{u \neq v} w'(\{u, v\}) x_u x_v \\ &= \sum_u \left[\sum_v w'(\{u, v\}) \right] x_u^2 - \sum_{u \neq v} w'(\{u, v\}) x_u x_v. \end{aligned}$$

By defining $Q_{u,u} = \sum_v w'(\{u, v\})$ and $Q_{u,v} = -w'(\{u, v\})$ for $u \neq v$, we establish that the objective value of the QUBO corresponds to the cut size, which is also the objective value of the max cut problem. Consequently, maximizing this objective value leads to an optimal solution for the max cut problem.

In the context of solving the max cut problem with QUBO solvers available through cloud services, it becomes necessary to transmit the values of $Q_{u,v}$ for every pair of u, v. Consequently, the quantity of real numbers required to be sent is on the order of $O(|V|^2)$. This count becomes substantially large for large graphs, turning the communication cost into a critical bottleneck for the max cut solver.

By opting to submit only the non-zero entries of $Q_{u,v}$, we can significantly reduce the communication cost. This means sending the QUBO problem in the format $(u, v, Q_{u,v})$ where $Q_{u,v} \neq 0$. From our definition of $Q_{u,v}$, it is evident that for $u \neq v$, $Q_{u,v}$ is non-zero if and only if $\{u, v\} \in E$. Therefore, the communication cost with this method of submission is $O(|E|)$, which is substantially more efficient for scenarios where $|E| \ll |V|^2$, or in other words, when the input graph is sparse.

2.3 Graph Sparsification by Effective Resistances [50]

In this section, we explore the concept of graph sparsification through effective resistances. Consider the input graph denoted as $G = (V, E, w)$. Our objective is to construct a graph $\mathcal{G} = (V, \mathcal{E}, \mathsf{w})$ in such a way that for any cut $S \subseteq V$, the relationship $w_G(S) \approx w_{\mathcal{G}}(S)$ holds true. This approach aims to ensure that the weight of any given cut S in the original graph G closely approximates the weight of the same cut in the sparsified graph $\mathcal{G}$.

Given a parameter q, this method begins with an initially empty set $\mathcal{E}$. The process involves selecting edges from the graph G a total of q times to be added to $\mathcal{E}$. During each selection, every edge $e \in E$ has a chance of being chosen, with this probability denoted as p_e and to be detailed in the following paragraph. If an edge e that is not already in $\mathcal{E}$ is selected, we assign its weight in $\mathcal{G}$ as $\mathsf{w}(e) = w(e)/(q \cdot p_e)$. In cases where e is already in $\mathcal{E}$, we increase $\mathsf{w}(e)$ by $w(e)/(q \cdot p_e)$. This approach ensures that each edge's contribution to the total weight is adjusted based on its probability of selection and the number of selections, thereby maintaining the graph's structure in $\mathcal{G}$.

To establish the probability distribution $(p_e)_{e \in E}$, we start by defining the concept of effective resistance for each edge e in E, denoted as R_e. We treat the graph G as if it was an electrical circuit, where each edge e is equivalent to a resistor, the resistance of which is inversely proportional to the weight of the edge, given as $1/w_e$. In this analogy, the effective resistance R_e of an edge $e = \{u, v\}$ is understood as the electrical resistance experienced between nodes u and v.

Subsequently, the probability p_e for each edge e is defined as

$$p_e = w_e R_e / \sum_{e' \in E} (w_e R_{e'}).$$

This formulation assigns higher probabilities to edges with greater effective resistance, reflecting their relative importance in the electrical flow analogy of the graph.

The following theorem is shown in [50].

Theorem 1. *If $q = 9|V| \cdot \log |V|/\epsilon^2$, then, for all $S \subseteq V$, $w_G(S) \leq w_{\mathcal{G}}(S) \leq (1 + \epsilon) w_G(S)$.*

We have from the theorem that we would obtain a sparse graph with $|\mathcal{E}| = O(|V| \log |V|)$ that preserves the cut size by the sparsification technique.

3 Proposed Method

Our approach is depicted in Figure 1. Rather than directly sending the original graph G to the QUBO solver provided by cloud services, we initially apply effective resistance sampling to sparsify the graph. The resultant sparsified graph, denoted as $\mathcal{G}$, is then submitted to the solver.

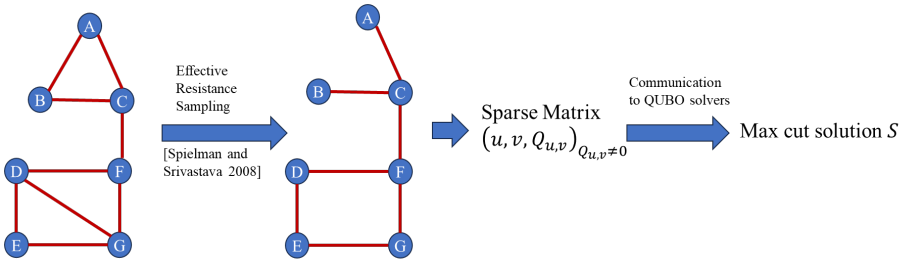

Fig. 1: Outline of our proposal

The following theorem is directly followed from Theroem 1.

Theorem 2. *Given that S' is a cut derived from the QUBO solver using our method, and S^* represents the optimal maximum cut, we can establish that:*

$$w_G(S') \leq w_G(S^*) \leq (1+\epsilon) w_G(S').$$

Consequently, our algorithm is a $(1+\epsilon)$-approximation algorithm for the max cut problem.

Proof. Because S' is the optimal max cut solution for the graph $\mathcal{G}$, we have that $w_{\mathcal{G}}(S') \geq w_{\mathcal{G}}(S^*)$. Applying Theorem 1, we obtain

$$w_G(S') \geq \frac{1}{1+\epsilon} w_{\mathcal{G}}(S') \geq \frac{1}{1+\epsilon} w_{\mathcal{G}}(S^*) \geq \frac{1}{1+\epsilon} w_G(S^*).$$

Hence, $w_G(S^*) \leq (1+\epsilon) w_G(S')$.

Theorem 1 reveals that $|\mathcal{E}| = O(|V| \log |V|)$, indicating that the communication cost associated with sending the sparsified graph $\mathcal{G}$ to cloud servers is also $O(|V| \log |V|)$. Therefore, our method can achieve an asymptotic improvement in communication costs for dense input graphs where the number of edges is on the order of $O(|V|^2)$. However, when dealing with sparse input graphs, our approach does not yield a significant reduction in communication costs.

In Theorem 2, we assume that our QUBO solver is exact, meaning it always delivers the optimal solution. However, this result can be extended to scenarios where the solver is approximate. If the QUBO solver functions as an α-approximation algorithm, then the outcome produced by our method can be demonstrated to be an $\alpha(1+\epsilon)$-approximation.

4 Experimental Results

We conduct experiments on the proposed method and give the experimental results in this section.

All experiments were carried out on a personal computer running Windows 11, equipped with an 11th Gen Intel(R) Core(TM) i7-1165G7 @2.80GHz CPU and 16 GB of RAM. The code for these experiments was written in Python. Furthermore, we utilized publicly available datasets as provided in [38]. However, as it is assumed by the effective resistance samplings that all weights are non-negative, the weights used in our experiments are absolute values of those provided in the publicly available datasets. The values presented in this paper represent the mean of ten separate replications.

4.1 Gap in Solutions Due to Graph Sparsification

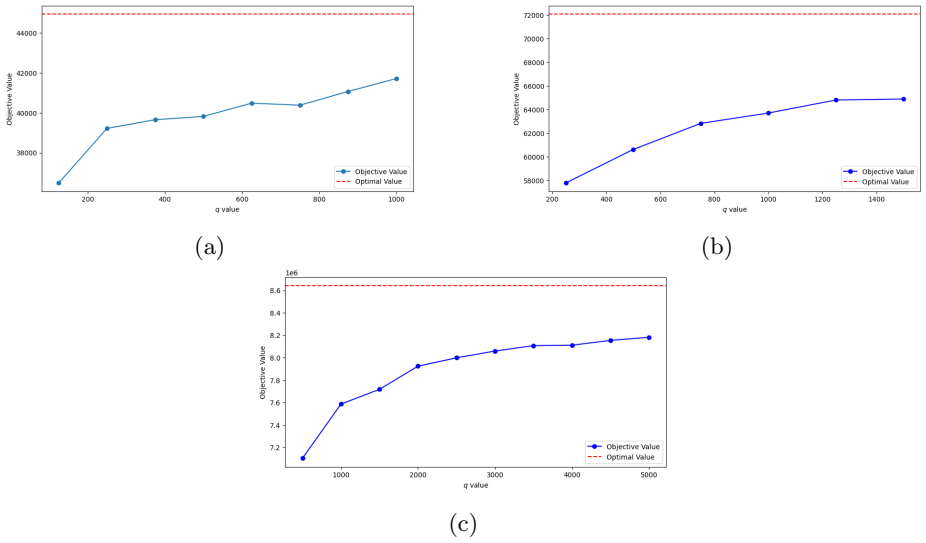

Fig. 2: Comparisons of the optimal values derived from the original max cut instances against the objective values from the sparsified graphs for (a) be120.3.1, (b) be250.1, and (c) mannino_k487.c

In this subsection, we examine the extent to which the optimal solutions are changed by effective resistance sampling. While our primary focus is on developing an algorithm suitable for quantum-inspired optimization, for these experiments, we have opted to use a classical solver, specifically Gurobi [20]. The rationale behind this choice is Gurobi's ability to guarantee the optimality of the solutions it generates. We employ the QUBO optimization feature available in Gurobi Optimods of Gurobi version 10.0.3.

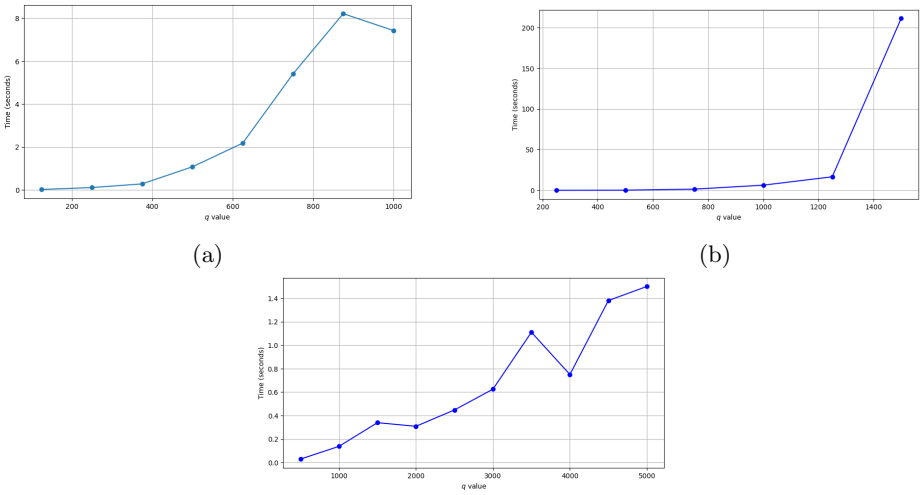

Fig. 3: Comparisons of the computation time of Gurobi when the inputs are sparsified graphs for (a) be120.3.1, (b) be250.1, and (c) mannino_k487.c

We initially evaluated our proposed methods using three distinct instances, each varying in node count and type. These instances are "be120.3.1", "be250.1", and "mannino_k487c". The datasets "be120.3.1" and "be250.1" are synthetic and were utilized in [3]. They were created using generators described in [44]. Specifically, "be120.3.1" comprises 121 nodes and 2242 edges, whereas "be250.1" contains 251 nodes and 3269 edges. The "mannino_k487c" dataset, on the other hand, is rooted in real-world data concerning radio frequency interferences among major Italian cities, as detailed in [4]. This dataset features 487 nodes and 8511 edges.

In these experiments, we focus on the variable q, which represents the number of times edges are sampled from the input graph. We experiment with varying the value of q. It is crucial to understand that q does not directly correspond to the number of edges in the sparsified graph, denoted as $|\mathcal{E}|$, since an edge can be selected multiple times during sampling. However, it is evident that $|\mathcal{E}| \leq q$, and generally, a larger q tends to result in a higher number of edges in the sparsified graph.

Theorem 1 suggests setting q to $\frac{9|V|\cdot\log|V|}{\epsilon^2}$. While this value is theoretically smaller than the edge count for dense graphs in an asymptotic sense, the sizeable constant factor $\frac{9}{\epsilon^2}$ can lead to a q that exceeds the actual number of edges, especially when the input graph G is relatively small. Take, for instance, when ϵ is 0.1, this results in approximately 522261 for "be120.3.1", 1248200 for "be250.1", and 2712316 for "mannino_k487c". Because of this, we have opted to use a reduced q value for our experimental evaluations.

Figure 2 presents the outcomes for the three specified instances. Examination of the figure reveals that the objective function improves as more edges are sampled and the value of q increases. Our method achieves a cut size that exceeds 90% of the optimal cut size for $q \geq 500$ in "be120.3.1", for $q \geq 1500$ in "be250.1", and for $q \geq 2000$ in "mannino_k487c". Correspondingly, these thresholds yield edge counts of 424 for "be120.3.1", 1092 for "be250.1", and 1231 for "mannino_k487c". These results indicate that our approach not only secures a 0.9-approximation to the solution but also facilitates a substantial reduction in communication costs—81% for "be120.3.1", 67% for "be250.1", and 86% for "mannino_k487c".

Table 1: The reduction in communication costs and the approximation ratios achieved by our algorithms across various graph types are detailed in [38]

Dataset Name	$\|V\|$	$\|E\|$	$\|\mathcal{E}\|$	**Reduction in Comm. Cost**	Optimal Value	Our Objective Value	**Approx. Ratio**
bqp250-1	251	3339	1163.6	**0.65151**	143669	129863	**0.90390**
gka1e	201	2124	810.7	**0.61831**	48263	42829	**0.88741**
ising2.5-150_6666	150	10722	387.3	**0.96388**	9067341	8502808	**0.93774**
g05_100.0	100	2475	452.3	**0.81725**	1430	1309	**0.91538**
w05_100.0	100	2343	432.3	**0.81549**	7737	7033.9	**0.90912**
G_1	800	19176	3598.3	**0.8124**	11598	10412.3	**0.8978**

Our experiment results with these three datasets yield a 0.9-approximation solution when setting q to roughly $5|V|$. Consequently, we extrapolate this finding to additional instance types in [38]. As demonstrated in Table 1, a similar approximation ratio is achieved for all tested instance types with q set at $5|V|$. Notably, there is a substantial decrease in communication cost particularly when the original graph G is dense.

4.2 Computation Time in Classical Solver

Because graph sparsification techniques are often employed to reduce computation time, it is worth investigating whether our sparsification method also reduces the computational times for classical solvers.

Figure 3 demonstrates that sparsification does indeed have a significant effect on reducing the computational time for Gurobi. For the original "be120.3.1" and "be250.1" inputs, the solver requires more than three hours to find a solution, whereas with the sparsified graphs at $q \approx 5|V|$, the computation times drop dramatically to 2.18 seconds and 16.6 seconds, respectively. There is also a clear pattern where larger values of q and increased edge counts correlate with longer computation times.

Despite this, it is noteworthy that Gurobi is able to quickly find reasonably good solutions for denser graphs. In every test conducted, solutions surpassing those of the sparsified graphs were obtained in under five seconds using the original graphs. A large part of the time that the solver spends on the dense graphs

is devoted to proving that its solutions are optimal. This leads us to conclude that reduced computation time may not be a decisive advantage of sparsification techniques. Gurobi is capable of providing viable solutions for larger graphs. Nevertheless, given our focus on quantum-inspired solvers in this paper, we opt to limit our experiments with Gurobi to cases where it can assure the optimality of its results.

4.3 Experiments on Quantum-Inspired Solvers

We employed the Fixstars Amplify Annealing Engine [16] to corroborate our findings with QUBO quantum-inspired solvers. For all instances, whether original or sparsified, we imposed a solver time constraint of 10 seconds. Consistent with the methodology outlined in the preceding section, we set the value of q to be $5|V|$ in this experiment.

Table 2: Reduction in communication cost and changes in objective value by the graph sparsification technique when conducting experiments on QUBO quantum-inspired solver

Dataset Name	Reduction in Comm. Cost	Changes in Objective Value
be120.3.1	0.779	0.911
be250.1	0.704	0.891
mannino_k487.c	0.835	0.92
bqp250-1	0.71	0.895
gka1e	0.65	0.909
ising2.5-150_6666	0.964	0.947
g05_100.0	0.816	0.908
w05_100.0	0.812	0.902
G_1	0.812	0.897

Table 2 shows that the outcomes obtained using quantum-inspired solvers align closely with those presented in Table 1, confirming consistency across all datasets tested with the classical solver.

We also conducted experiments to test the effectiveness of our pre-processing methods on large graphs and real-world social networks. Our findings, presented in Table 3, demonstrate that our approach yields consistent results even with graphs exceeding 100,000 edges. We used graphs generated by a tool given in [24]. Specifically, `rnd_graph1000_10_1` is a randomly generated graph with 1,000 nodes, a 10% connection density, and seed = 1. The `leap_xx_y_z` graphs represent leap graphs on y-dimensional chessboards of size xx, with the "z" parameter indicating the graph type: root graphs for `z = 1` and bishop graphs for `z = 2`. Additionally, "facebook", "congress", and "wiki_vote" were sourced from the Stanford large network dataset collection, representing real social networks [33]. These networks have been previously analyzed in several notable studies on social network behavior and characteristics [34, 32, 15]. The graph labeled "gplus_100000" represents a subgraph of the gplus network, induced by

the initial 100,000 edges listed in the file. Given that this selection includes some repeated edges, the resulting graph comprises 12,912 nodes and 807,535 edges. This specific subset was chosen because the basic package of the Fixstars Amplify Annealing Engine supports a maximum of 16,000 nodes.

Table 3: Reduction in communication costs and changes in objective values by the graph sparsification technique when conducting experiments on QUBO quantum-inspired solver

Dataset Name	$\|V\|$	$\|E\|$	$\|\mathcal{E}\|$	**Reduction in Comm. Cost**	Optimal Value	Our Objective Value	**Reduction in Obj. Value**
rnd_graph1000_10_1	1000	49950	4752	**0.90486**	28587	26019.8	**0.91020**
leap_30_2_1	900	26100	401	**0.98464**	202500	178281.6	**0.88040**
leap_30_2_2	900	17110	3739.8	**0.78143**	101250	89984.5	**0.88874**
leap_10_3_1	1000	13500	3982.2	**0.70502**	37500	32921.9	**0.87792**
leap_10_3_2	1000	17100	4148.1	**0.75742**	31284	27711.2	**0.88579**
facebook	4039	88234	15874.2	**0.82009**	50600	46368.3	**0.91637**
congress	475	13289	1940.5	**0.85398**	38.98	34.37	**0.88173**
wiki_vote	7115	103689	22078.7	**0.78707**	73407	67675.3	**0.92192**
gplus_1000000	12912	807535	56175.7	**0.93043**	563656	400932.1	**0.71131**

Moreover, experiments were carried out on subgraphs of varying sizes derived from wiki_vote. These subgraphs were obtained through random walks to maintain the integrity of the network's structure. The data presented in Table 4 indicates that across all subgraph sizes tested, the cost reduction and approximation ratios achieved were consistent. This consistency leads us to posit that comparable outcomes are attainable for substantially larger graphs. Consequently, our pre-processing technique has the potential to markedly diminish communication costs in such scenarios.

Table 4: Reduction in communication costs and changes in objective values of subgraphs of the wiki_vote graph obtained from random walks on QUBO quantum-inspired solver

Random Walk Length	$\|V\|$	$\|E\|$	$\|\mathcal{E}\|$	**Cost Reduction**	Optimal Solution	Our Objective Value	**Approx. Ratio**
500	411	6523	1640.8	**74.85%**	4371	3749.4	**85.78%**
1000	755	18703	3208.6	**82.84%**	12563	10658.3	**84.84%**
1500	985	25812	4169.4	**83.85%**	17295	14861.8	**85.93%**
2000	1206	34352	5154	**85.04%**	23352	20100.4	**86.08%**
2500	1411	43040	6087.5	**85.86%**	29164	25123.8	**86%**
3000	1524	47535	6544.3	**86.23%**	32296	27871.2	**86.30%**
3500	1653	51371	7099.4	**86.18%**	35361	30478.3	**86.19%**
4000	1807	56964	7742.6	**86.41%**	39199	34066.9	**86.91%**
4500	1928	61425	8207.1	**86.64%**	42419	37179.1	**87.65%**
5000	2038	63534	8679.2	**87.26%**	43846	38260.9	**87.26%**

Table 5 displays the results for the Facebook network's subgraphs. It is evident that there's a variation in cost reduction among these subgraphs. This variation is likely a consequence of the network's multi-cluster structure, with each cluster possessing a distinct edge density, leading to differing levels of cost

reduction. The random walk method does not uniformly sample nodes across clusters, resulting in varied cost reduction outcomes for the graphs generated by the algorithm. Although the cost reduction figures exhibit some fluctuation, they consistently fall within the range of 75% to 90%. On the other hand, despite the varied structures of each subgraph, the approximation ratio remains consistent across all experiments, lying within the 91% to 93% bracket.

Table 5: Reduction in communication costs and changes in objective values of subgraphs of the facebook graph obtained from random walks on QUBO quantum-inspired solver

Random Walk Length	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\mathcal{E}\rvert$	**Cost Reduction**	Optimal Solution	Our Objective Value	**Approx. Ratio**
500	312	13305	1390.1	**89.55%**	7298	6772.1	**92.79%**
1000	684	12186	2715.3	**77.72%**	7043	6452.4	**91.61%**
1500	882	25028	3658.2	**85.38%**	14029	12948.9	**92.30%**
2000	1022	16441	4045.3	**75.40%**	9600	8775.5	**91.41%**
2500	982	35349	4238.3	**88.01%**	19685	18181.6	**92%**
3000	1355	42222	5799	**86.26%**	23689	21843.5	**92.21%**
3500	1465	30107	5968.2	**80.20%**	17375	15889.8	**91.45%**
4000	1709	50934	7285.1	**85.70%**	28637	26409.1	**92.22%**
4500	1898	47302	7949.8	**83.19%**	26859	24618.1	**91.66%**
5000	1828	57824	7851.5	**86.42%**	32543	29918	**91.93%**

4.4 Discussions on Results on Classical and Quantum-Inspired Solvers

The argument could be made that similar or even superior approximation ratios to those achieved in our research might be attainable using an approximation algorithm based on semi-definite programming, as demonstrated in previous studies [18, 37]. This algorithm is indeed capable of providing polynomial-time approximation solutions for max cut problems. However, a notable limitation of semi-definite programming is its computational intensity, particularly for problems involving over 100,000 nodes [29], where local execution becomes impractical. In such scenarios, our method proves advantageous, offering a viable solution by enabling the processing of these large instances through cloud services.

In these experiments, our primary objective is to demonstrate that edge sampling can yield reasonable approximation ratios. Therefore, we confined our experimentation to smaller instances (with $|V| \leq 12912$) where obtaining optimal solutions is feasible. Nonetheless, given the consistent results across all tested instance sizes, we are confident that similar outcomes would be achievable with larger graphs.

5 Conclusion and Future Works

We introduce the application of graph sparsification as a pre-processing step for solving the maximum cut problem in cloud-based environments. Our ex-

perimental results demonstrate that this approach, when applied to classical and quantum-inspired solvers, consistently yields solutions with an approximation ratio of about 0.9, while simultaneously achieving a significant reduction in communication costs to cloud servers, ranging between 60% and 95%.

In our future research, we plan to expand our experiments to include quantum solvers. At present, quantum solvers are limited to addressing small-scale max cut instances. As a result, the communication overhead required to send the max cut problem to the solvers is not significantly high at this point. On the other hand, it is understood that a sparser graph results in shallower quantum circuits, thereby reducing the noise in quantum computations. The graph sparsification technique has already been used for solving max cut for the noisy data published under differential privacy [1]. Additionally, a recent work [23] highlights that an increased number of edges may result in higher complexity during circuit optimization processes. A dense input graph reduces the likelihood of achieving an efficient quantum circuit. In summary, we hypothesize that the graph sparsification could improve solution quality and simplify the process of optimizing quantum circuits.

Pre-processing techniques for combinatorial optimization problems utilizing machine learning algorithms have been proposed in previous studies [51, 35]. However, our initial experiments suggest that directly applying these methods to the max cut problem may not yield the best results. We observed that a machine learning model trained on small graphs does not effectively transfer to larger graphs within this problem domain. As a result, our future work aims to develop a machine learning-based sparsification technique specifically tailored for the max cut problem.

References

1. Arora, R., Upadhyay, J.: On differentially private graph sparsification and applications. Advances in Neural Information Processing Systems **32** (2019)
2. Benczúr, A.A., Karger, D.R.: Approximating s-t minimum cuts in $O(n^2)$ time. In: STOC'96. pp. 47–55 (1996)
3. Billionnet, A., Elloumi, S.: Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. Mathematical Programming **109**, 55–68 (2007)
4. Bonato, T., Jünger, M., Reinelt, G., Rinaldi, G.: Lifting and separation procedures for the cut polytope. Mathematical Programming **146**, 351–378 (2014)
5. Boros, E., Crama, Y., Hammer, P.L.: Upper-bounds for quadratic 0–1 maximization. Operations Research Letters **9**(2), 73–79 (1990)
6. Burer, S., Monteiro, R.D., Zhang, Y.: Rank-two relaxation heuristics for max-cut and other binary quadratic programs. SIAM Journal on Optimization **12**(2), 503–521 (2002)
7. Charfreitag, J., Jünger, M., Mallach, S., Mutzel, P.: McSparse: Exact solutions of sparse maximum cut and sparse unconstrained binary quadratic optimization problems. In: ALENEX'22. pp. 54–66 (2022)
8. Codognet, P.: Constraint solving by quantum annealing. In: ICPP Workshops'21. pp. 1–10 (2021)

9. Codognet, P.: Domain-wall/unary encoding in QUBO for permutation problems. In: QCE'22. pp. 167–173 (2022)
10. Dahi, Z.A., Alba, E.: Metaheuristics on quantum computers: Inspiration, simulation and real execution. Future Generation Computer Systems **130**, 164–180 (2022)
11. Dunning, I., Gupta, S., Silberholz, J.: What works best when? a systematic evaluation of heuristics for max-cut and QUBO. INFORMS Journal on Computing **30**(3), 608–624 (2018)
12. Elloumi, S., Faye, A., Soutif, E.: Decomposition and linearization for 0-1 quadratic programming. Annals of Operations Research **99**(1-4), 79–93 (2000)
13. Ferizovic, D.: A Practical Analysis of Kernelization Techniques for the Maximum Cut Problem. Ph.D. thesis, Karlsruher Institut für Technologie (KIT) (2019)
14. Ferizovic, D., Hespe, D., Lamm, S., Mnich, M., Schulz, C., Strash, D.: Engineering kernelization for maximum cut. In: ALENEX'20. pp. 27–41 (2020)
15. Fink, C.G., Fullin, K., Gutierrez, G., Omodt, N., Zinnecker, S., Sprint, G., McCulloch, S.: A centrality measure for quantifying spread on weighted, directed networks. Physica A (2023)
16. Fixstars: About Amplify AE (2023), https://amplify.fixstars.com/ja/docs/amplify-ae/about.html
17. Gharibian, S., Le Gall, F.: Dequantizing the quantum singular value transformation: hardness and applications to quantum chemistry and the quantum pcp conjecture. In: STOC'22. pp. 19–32 (2022)
18. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM **42**(6), 1115–1145 (1995)
19. Grötschel, M., Nemhauser, G.L.: A polynomial algorithm for the max-cut problem on graphs without long odd cycles. Mathematical Programming **29**(1), 28–40 (1984)
20. Gurobi Optimization, L.: Gurobi optimizer reference manual (2021)
21. Hammer, P.L., Hansen, P., Simeone, B.: Roof duality, complementation and persistency in quadratic 0–1 optimization. Mathematical Programming **28**, 121–155 (1984)
22. Herrero-Collantes, M., Garcia-Escartin, J.C.: Quantum random number generators. Reviews of Modern Physics **89**(1), 015004 (2017)
23. Ito, T., Kakimura, N., Kamiyama, N., Kobayashi, Y., Okamoto, Y.: Algorithmic theory of qubit routing. In: WADS'23. pp. 533–546 (2023)
24. JRT: Rudy: a rudimental graph generator by JRT (2023), https://web.stanford.edu/ yyye/yyye/Gset/
25. Karger, D.R.: Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In: SODA'93. pp. 21–30 (1993)
26. Karp, R.M.: Reducibility among combinatorial problems. Springer (2010)
27. Khandekar, R., Rao, S., Vazirani, U.: Graph partitioning using single commodity flows. Journal of the ACM **56**(4), 1–15 (2009)
28. Kikuchi, S., Togawa, N., Tanaka, S.: Dynamical process of a bit-width reduced Ising model with simulated annealing. IEEE Access **11**, 95493–95506 (2023)
29. Kim, S., Kojima, M.: Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations. Computational Optimization and Applications **26**, 143–154 (2003)
30. King, R.: An improved approximation algorithm for quantum max-cut. Quantum **7**, 1180 (2022)

31. Lamm, S.: Scalable Graph Algorithms using Practically Efficient Data Reductions. Ph.D. thesis, Karlsruher Institut für Technologie (KIT) (2022)
32. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Predicting positive and negative links in online social networks. In: WWW'10. pp. 641–650 (2010)
33. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data (Jun 2014)
34. Leskovec, J., Mcauley, J.: Learning to discover social circles in ego networks. Advances in Neural Information Processing Systems **25** (2012)
35. Li, M., Tu, S., Xu, L.: Generalizing graph network models for the traveling salesman problem with Lin-Kernighan-Helsgaun heuristics. In: NeurIPS'23. pp. 528–539 (2023)
36. Liers, F., Pardella, G.: Partitioning planar graphs: a fast combinatorial approach for max-cut. Computational Optimization and Applications **51**(1), 323–344 (2012)
37. Mahajan, S., Ramesh, H.: Derandomizing approximation algorithms based on semidefinite programming. SIAM Journal on Computing **28**(5), 1641–1663 (1999)
38. Mallach, S., Junger, M., Charfreitag, J., Jordan, C.: (Prototype of a) maxcut and BQP instance library (2021), http://bqp.cs.uni-bonn.de/library/html/index.html
39. Mazumder, A., Sen, A., Sen, U.: Benchmarking metaheuristic-integrated quantum approximate optimisation algorithm against quantum annealing for quadratic unconstrained binary optimization problems. arXiv preprint arXiv:2309.16796 (2023)
40. McCormick, S.T., Rao, M.R., Rinaldi, G.: Easy and difficult objective functions for max cut. Mathematical Programming **94**, 459–466 (2003)
41. Mirka, R., Williamson, D.P.: An experimental evaluation of semidefinite programming and spectral algorithms for max cut. ACM Journal of Experimental Algorithmics **28**, 1–18 (2023)
42. Nemhauser, G.L., Trotter Jr, L.E.: Vertex packings: structural properties and algorithms. Mathematical Programming **8**(1), 232–248 (1975)
43. Oku, D., Tawada, M., Tanaka, S., Togawa, N.: How to reduce the bit-width of an Ising model by adding auxiliary spins. IEEE Transactions on Computers **71**(1), 223–234 (2020)
44. Pardalos, P.M., Rodgers, G.P.: Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing **45**(2), 131–144 (1990)
45. Punnen, A.P.: The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications. Springer Nature (2022)
46. Rehfeldt, D., Koch, T., Shinano, Y.: Faster exact solution of sparse maxcut and QUBO problems. Mathematical Programming Computation pp. 1–26 (2023)
47. Rosenberg, G., Vazifeh, M., Woods, B., Haber, E.: Building an iterative heuristic solver for a quantum annealer. Computational Optimization and Applications **65**, 845–869 (2016)
48. Shih, W.K., Wu, S., Kuo, Y.S.: Unifying maximum cut and minimum cut of a planar graph. IEEE Transactions on Computers **39**(5), 694–697 (1990)
49. Shor, P.W.: Introduction to quantum algorithms. In: Proceedings of Symposia in Applied Mathematics. vol. 58, pp. 143–160 (2002)
50. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. In: STOC'08. pp. 563–568 (2008)
51. Tayebi, D., Ray, S., Ajwani, D.: Learning to prune instances of k-median and related problems. In: ALENEX'22. pp. 184–194 (2022)