Network Vulnerability Analysis via Quantum Computing

Andrew P. Kennedy*, Thang N. Dinh†, My T. Thai*
*Department of Computer Information Science & Engineering
University of Florida, Gainesville, FL, 32611 USA
andrewphkennedy@gmail.com
mythai@cise.ufl.edu
†Department of Computer Science
Virginia Commonwealth University, VA 23284 USA
tndinh@vcu.edu

***Abstract*—In many complex networking systems, identifying critical nodes whose removal maximally disrupts network connectivity remains an important yet computationally challenging problem for network vulnerability analysis. Finding near-optimal solutions is known to be NP-hard. In this paper, we explore the potential of near-term quantum computing devices to efficiently solve the k-Critical Node Detection (k-CND) problem. We formulate the problem as a quadratic unconstrained binary optimization (QUBO), a mathematical optimization over binary variables amenable to solution on quantum annealers. We present a novel integer linear programming (ILP) formulation and its conversion into QUBOs and provide benchmarking results on D-Wave's quantum annealers. Our theoretical analysis proves that our proposed formulation, ILP2, generates substantially smaller QUBO than the state-of-the-art ILP1 formulation. Experimentally, our efficient QUBO yields a 59.7% decrease in QUBO variables on a graph with 10 vertices and 40 edges, and an 11.7% reduction in qubits on a 15-vertex, 22-edge graph compared to that of QUBO1. We analyze the solution quality and running time across quantum, classical, and hybrid solvers to assess the potential for quantum advantage. Our work showcases the promise and challenges of tackling this important graph problem on near-term quantum hardware.**

I. Introduction

Networks are pervasive in various domains, from social interactions and biological systems to technological infrastructure and financial markets. The connectivity patterns of these networks often determine their functionality, efficiency, and resilience. Robust networks maintain their operational integrity in the face of random failures or targeted attacks, while fragile networks are vulnerable to disintegration when critical nodes are compromised [1], [2]. Quantifying network robustness and identifying critical nodes is thus crucial for understanding and optimizing real-world systems.

The k-Critical Node Detection (k-CND) problem aims to identify a set of k nodes whose removal maximally disrupts pairwise network connectivity [3]–[6]. This problem has diverse applications, such as assessing the resilience of networks (e.g., power grids, transportation systems) [7], identifying key players in social networks [8], and characterizing the robustness of biological systems (e.g., protein interaction networks) [9]. However, the k-CND problem is computationally challenging, known to be NP-hard and inapproximable within any constant factor on general graphs [6].

A. Related Works

Classical approaches to the k-CND problem encompass a wide range of techniques, including exact algorithms, approximation algorithms, and heuristics. Exact methods, such as branch-and-cut algorithms based on integer linear programming (ILP) formulations [10]–[12], guarantee optimality but are limited to small instances due to their exponential worst-case time complexity. Approximation algorithms provide sub-optimal solutions with provable quality guarantees, but their practical performance often falls short of expectations [3]–[5], [13]. Heuristic methods, such as greedy strategies and local search algorithms [14], and volutionary algorithms [15] have been used to solve the k-CND problem. While these heuristic methods can handle larger instances, they lack optimality guarantees.

Given the limitations of classical approaches, there has been growing interest in leveraging quantum computing to tackle computationally hard problems like k-CND. Quantum algorithms have the potential to provide exponential speedups over their classical counterparts for certain problems, such as integer factorization [16] and unstructured search [17]. Quantum annealing, in particular, is a promising paradigm for solving combinatorial optimization problems [18], [19]. By exploiting quantum fluctuations, quantum annealers can explore the solution space more efficiently than classical optimizers, potentially leading to faster convergence and better solutions [20].

Recent advancements in quantum computing have spurred a race on applying quantum computers for real-world problems. In 2019, Google achieved a major milestone in quantum computing by demonstrating quantum supremacy using a 53-qubit superconducting quantum processor [21]. They performed a task involving the generation of random numbers that would take the world's most powerful supercomputer approximately 10,000 years to complete, while their quantum processor completed it in just 200 seconds. This achievement marked a significant step forward in the field of quantum

computing and showcased the potential of quantum devices to outperform classical computers on certain tasks. Furthermore, Bravyi et al. [22] presented an end-to-end quantum error correction protocol that implements fault-tolerant memory using a family of low-density parity-check codes. Their approach achieves an error threshold of 0.7% for the standard circuit-based noise model, on par with the surface code that was the leading code in terms of error threshold for 20 years. This advancement brings demonstrations of a low-overhead fault-tolerant quantum memory within the reach of near-term quantum processors.

In addition to universal quantum computers, quantum annealing hardware, such as D-Wave's quantum annealers [23], have sparked a flurry of research on solving real-world optimization problems. Quantum annealing has been applied to various domains, including machine learning, in which Neven et al. used quantum annealing to help boost weak classifiers for identifying cars in images [24], traffic flow optimization, in which Neukart et al. showed empirically that quantum approaches can be suitable even for real-world, time-critical tasks [25], and portfolio optimization, in which Rosenberg et al. investigated different methods for encoding integer variables to a format amenable to quantum annealers [26].

A recent breakthrough by King et al. [27] demonstrated quantum-critical spin-glass dynamics on thousands of qubits with a superconducting quantum annealer. They showed that quantum annealing can bring spin glasses into low-energy states faster than conventional thermal annealing, providing both theoretical and experimental support for large-scale quantum simulation and a scaling advantage in energy optimization. Furthermore, King et al. [28] established computational supremacy in quantum simulation by demonstrating that superconducting quantum annealing processors can rapidly generate samples in close agreement with solutions of the Schrödinger equation, outperforming state-of-the-art classical simulations. Despite these promising developments, the applicability of quantum annealing to complex network problems like k-CND remains largely unexplored.

B. Contributions

In this work, we bridge this gap by investigating the potential of quantum annealing for the k-CND problem. We propose a novel ILP formulations, called ILP2, and the conversions of the studied ILPs into quadratic unconstrained binary optimizations (QUBOs) [29], which are amenable to solution on quantum annealers. We prove that ILP2 substantially yields fewer QUBO variables than those for ILP1, potentially leading to more efficient quantum annealing. We conduct extensive experiments on D-Wave's quantum annealers, comparing the performance of quantum, classical, and hybrid solvers in terms of solution quality and running time. Our results demonstrate the promise and challenges of solving the k-CND problem on near-term quantum hardware.

The main contributions of this work are threefold:

1) We introduce two novel ILP formulations for the k-CND problem and provide efficient QUBO mappings for quantum annealing. We theoretically prove the efficiency of ILP2 over ILP1 in terms of QUBO size.
2) We present extensive experimental results on D-Wave's quantum annealers, benchmarking the performance of quantum, classical, and hybrid solvers on the k-CND problem. We analyze the impact of problem size, network density, and annealing parameters on solution quality and running time.
3) We discuss the practical challenges and future prospects of solving network optimization problems on near-term quantum hardware. We highlight the need for algorithmic innovations, error mitigation techniques, and hardware improvements to fully harness the potential of quantum annealing.

The rest of the paper is organized as follows. Section II presents preliminaries on the k-CND problem and an overview of quantum annealing. Section III introduces our efficient ILP-to-QUBO mappings. Section IV describes our experimental methodology and results. We conclude in Section V with a discussion of our findings and future research directions.

II. Preliminaries

A. Integer Linear Programming for k-CND

We present an Integer Linear Programming (ILP) formulation, denoted as ILP1, to solve the k-CND problem, for an arbitrary graph, $G = (V, E)$. Let s_i and d_{ij} be binary variables defined as follows:

$$s_i = \begin{cases} 1 & \text{if vertex } i \text{ is removed from the graph} \\ 0 & \text{otherwise} \end{cases}$$

$$d_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in different connected components} \\ 0 & \text{otherwise} \end{cases}$$

The ILP1 formulation for the k-CND problem is given by:

$$\text{minimize} \sum_{i=1}^{n} \sum_{j \neq i} (1 - d_{ij}) \qquad \textbf{(ILP1)}$$

$$\text{subject to } d_{ij} \leq s_i + s_j, \quad (i,j) \in E, \tag{1a}$$

$$d_{ij} + d_{jk} \geq d_{ik}, \ \forall i \neq j \neq k, \tag{1b}$$

$$\sum_{i=1}^{n} s_i \leq \mathrm{K} \tag{1c}$$

$$s_i \leq d_{ij}, \quad i \neq j, \tag{1d}$$

$$s_i, d_{ij} \in \{0,1\}, \ i,j \in [1..n] \tag{1e}$$

The objective function (**ILP1**) minimizes the pairwise connectivity of the graph by maximizing the number of vertex pairs that are in different connected components. Intuitively, this captures the goal of fragmenting the graph into disconnected components by removing a set of vertices.

Constraint (1a) ensures that if an edge exists between vertices i and j, and neither vertex is removed, then they must remain in the same connected component. In other words, if $s_i = s_j = 0$, then d_{ij} must also be 0. Constraint (1b) enforces the triangle inequality, which is a key property of connected components in a graph. It states that if vertices i and j are

in the same component (i.e., $d_{ij} = 0$), and vertices j and k are in the same component (i.e., $d_{jk} = 0$), then vertices i and k must also be in the same component (i.e., $d_{ik} = 0$). This constraint ensures the transitivity of connectivity among vertices. Constraint (1c) limits the number of removed vertices to at most k, which is the budget for the k-CND problem. This constraint allows the ILP solver to explore different subsets of vertices to remove while respecting the budget. Constraint (1d) guarantees that if a vertex i is removed (i.e., $s_i = 1$), it cannot be in the same connected component as any other vertex j. This constraint captures the effect of vertex removal on the connectivity of the graph. Finally, constraint (1e) defines the binary nature of the decision variables s_i and d_{ij}.

a) Formulation Size: The ILP1 formulation provides a precise mathematical model for the k-CND problem, allowing it to be solved using standard ILP solvers. However, the number of constraints in ILP1 grows rapidly with the size of the graph. In particular, constraint (1b) introduces $O(n^3)$ triangle inequality constraints, where $n = |V|$ is the number of vertices in the graph. This makes the ILP1 formulation computationally challenging for large graphs.

b) Improved ILP Formulations: To mitigate the complexity of the ILP1 formulation, Santos et al. [30] consider an alternative formulation that provides a stronger linear programming (LP) relaxation bound at the cost of a larger number of constraints. This formulation, denoted as ILP1-S, replaces constraint (1b) with the following set of constraints:

$$d_{jk} \geq d_{ij} - s_i, (i,j) \notin E, |N(i)| \leq |N(j)|, k \in N(i), \quad \text{(2a)}$$
$$d_{ik} \geq d_{ij} - s_j, (i,j) \notin E, |N(i)| > |N(j)|, k \in N(j) \quad \text{(2b)}$$

Here, $N(i)$ denotes the set of neighbors of vertex i. Constraint (2a) applies when the degree of vertex i is less than or equal to the degree of vertex j, and constraint (3) applies otherwise. These constraints enforce the triangle inequality in a more granular way by considering the connectivity between pairs of vertices that are not directly connected by an edge.

The ILP1-S formulation introduces $O(n^2 d)$ constraints, where d is the maximum degree of any vertex in the graph. In addition to using fewer constraints than ILP1, this formulation provides a tighter LP relaxation bound, which can lead to faster convergence of the ILP solver. The choice between ILP1 and ILP1-S depends on the specific characteristics of the graph and the available computational resources.

The ILP1 formulation provides a straightforward mathematical model for the k-CND problem, but it suffers from a large number of triangle inequality constraints. The ILP1-S formulation offers a stronger LP relaxation bound, but may require more constraints than other similar formulations [30].

In a similar vein, we present a more suitable representation for ILP1 than (1) in the sections to follow. This alternative formulation, which we will denote ILP1-S2, is also constructed to reduce the number of constraints, and replaces constraint (1b), with the following constraint:

$$d_{ij} + d_{jk} \geq d_{ik}, \quad \forall i \neq j, k \in N(i) \quad (3)$$

Theorem 1: The ILP1-S2 formulation (3) is equivalent to the ILP1 formulation (1) for the k-CND problem.

Proof: We prove the equivalence by showing that any feasible solution to ILP1 is also feasible for ILP1-S2, and vice versa.

($\Rightarrow$) Let (s, d) be a feasible solution to ILP1. Since (3) enforces only a subset of the constraints from (1b), any valid solution for ILP1 which satisfies (1b) must also satisfy (3). Hence, any such feasible solution to ILP1 must also be valid for ILP1-S2.

($\Leftarrow$) Let (s, d) be a feasible solution to ILP1-S2. Notice that (1b) can only be violated if there exists some $i, j, k \in V$, such that $d_{ij} = 0, d_{jk} = 0, d_{ik} = 1$. Since constraint (1b) is already enforced by (3) in the case where $k \in N(i)$, we need only prove that this constraint is enforced by ILP1-S2 even when $k \notin N(i)$. In other words, we will show that if there exists some $d_{ij} = 0, d_{jk} = 0$, such that $k \notin N(i)$, then $d_{ik} = 0$.

Since $d_{jk} = 0$, there must exist some path of vertices, $\{k, v_1, ..., v_n, j\} \subseteq V$. From (3), $d_{ik} <= d_{iv_1} + d_{kv_1}$. And, since we know $s_k = s_{v_1} = 0$, since there is an edge between k and v_1, it follows from (1a) that $d_{kv_1} = 0$. Hence, $d_{ik} <= div_1 + 0$. Using the same argument, $d_{ik} <= d_{iv_1} <= d_{iv_2} + d_{v_1 v_2} = d_{iv_2} <= ... <= d_{v_n j} + d_{ij} = 0$. Hence, the triangle inequality is enforced using (3), since this constraint ensures there exist no triplets of vertices, $(i, j, k) \in V$ such that $d_{ij} + d_{jk} < d_{ik}$. ■

These formulations (ILP1, ILP1-S, and ILP1-S2) serve as the foundation for developing efficient solution methods for the k-CND problem, including quantum-inspired approaches that will be discussed in the following sections.

B. Solving Combinatorial Problems with Quantum Annealing

To solve a problem using quantum annealing, it must first be formulated as a Quadratic Unconstrained Binary Optimization (QUBO) problem [29]. A QUBO is an optimization problem of the form:

$$\text{minimize } H(x) = \sum_{i<j} Q_{ij} x_i x_j + \sum_i h_i x_i, \quad (4)$$

where $x_i \in \{0, 1\}$ are binary variables, and Q_{ij} and h_i are real-valued coefficients.

To convert an ILP into a QUBO, a penalty approach is commonly used [31]. Each constraint in the ILP is transformed into a quadratic penalty term that is added to the objective function. The penalty terms are designed such that they are minimized when the corresponding constraints are satisfied. The resulting unconstrained optimization problem is a QUBO.

Once the problem is formulated as a QUBO, it can be solved using a quantum annealer. However, the QUBO must first be mapped onto the quantum hardware graph through a process called minor-embedding [32]. The quantum hardware graph has a limited number of qubits and a specific connectivity pattern, which may not directly match the structure of the QUBO. Minor-embedding involves representing each logical variable in the QUBO with a chain of physical qubits on the hardware graph, ensuring that the necessary interactions between variables can be realized.

The quantum annealing process itself involves evolving the quantum system from an initial state, where the qubits

are in a superposition of all possible states, to a final state that represents the solution to the optimization problem. This evolution is guided by an annealing schedule, which controls the strength of the interactions between qubits and the level of quantum fluctuations in the system [20].

Current quantum annealers have limitations in terms of the number of qubits and their connectivity [23]. These limitations can make it challenging to solve large-scale problems or problems with complex interaction patterns. Therefore, there is a strong motivation to find more compact QUBO formulations that can be efficiently embedded and solved on existing quantum hardware.

III. Efficient QUBO Formulation for the k-CND Problem

In this section, we introduce a new ILP formulation for the k-CND problem, denoted as ILP2, which leads to a more compact QUBO formulation compared to ILP1. The motivation behind ILP2 is to reduce the number of constraints in the formulation, which in turn results in a smaller QUBO with fewer variables and non-zero entries. We prove the correctness of ILP2 and provide a detailed analysis of the QUBO size reduction achieved by this formulation.

A. ILP2 Formulation

The ILP2 formulation for the k-CND problem is given by:

$$\text{minimize} \sum_{i=1}^{n}\sum_{j\neq i}(1-d_{ij}) \qquad \textbf{(ILP2)}$$

$$\text{subject to } d_{ij} \leq s_i + s_j, \qquad (i,j)\in E, \qquad (5a)$$

$$\frac{1}{|N(i)|}\sum_{k\in N(i)} d_{kj} \geq d_{ij} - s_i, \forall 1 \leq i < j \leq n \qquad (5b)$$

$$\sum_{i=1}^{n} s_i \leq \mathrm{K} \qquad (5c)$$

$$s_i \leq d_{ij}, \qquad i \neq j, \qquad (5d)$$

$$s_i, d_{ij} \in \{0,1\}, \qquad i,j \in [1..n] \qquad (5e)$$

The key difference between ILP2 and ILP1 is the replacement of the triangle inequality constraint (1b) in ILP1, which is given by:

$$d_{ij} + d_{jk} \geq d_{ik}, \quad \forall i \neq j \neq k, \qquad (6)$$

with the following constraint (5b) in ILP2:

$$\frac{1}{|N(i)|}\sum_{k\in N(i)} d_{kj} \geq d_{ij} - s_i, \quad \forall 1 \leq i < j \leq n. \qquad (7)$$

Here, $N(i)$ denotes the set of neighbors of vertex i, and $|N(i)|$ is the degree of vertex i. Constraint (5b) enforces the triangle inequality in a more compact manner by considering only the neighbors of each vertex, rather than all possible triples of vertices as in constraint (1b). This reduction in the number of constraints is crucial for obtaining a smaller QUBO formulation.

Theorem 2: The ILP2 formulation (5) is equivalent to the ILP1 formulation (1) for the k-CND problem.

Proof: We prove the equivalence by showing that any feasible solution to ILP1 is also feasible for ILP2, and vice versa.

($\Rightarrow$) Let (s,d) be a feasible solution to ILP1. Suppose, for the sake of contradiction, that (s,d) violates constraint (5b) for some $i,j \in V$. This implies that $s_i = 0$, $d_{ij} = 1$, and $\frac{1}{|N(i)|}\sum_{k\in N(i)} d_{kj} < 1$. The last condition means that there exists a vertex $k' \in N(i)$ such that $d_{k'j} = 0$. From constraint (1a), we have $s_{k'} = 0$. Since (s,d) is feasible for ILP1, constraint (1b) implies that $d_{ik'} = 0$. However, this contradicts the assumption that $d_{ij} = 1$, as the triangle inequality is violated. Therefore, (s,d) must also be feasible for ILP2.

($\Leftarrow$) Let (s,d) be a feasible solution to ILP2. Suppose, for the sake of contradiction, that (s,d) violates constraint (1b) for some $i,j,k \in V$. This implies that $d_{ij} = 0$, $d_{jk} = 0$, and $d_{ik} = 1$. Since $d_{ij} = 0$, there must be a path from i to j in the graph induced by the vertices not removed by the solution. Let $i, v_1, \ldots, v_m, j$ be the shortest such path. By constraint (5a), we have $s_{v_1} = 0$, and by constraint (5b), we have $d_{v_1 j} = 0$. This contradicts the assumption that $d_{ik} = 1$, as constraint (5b) would imply $d_{ik} = 0$. Therefore, (s,d) must also be feasible for ILP1. Thus, ILP1 and ILP2 are equivalent formulations for the k-CND problem. ∎

B. Converting ILP to QUBO: General Approach

Before diving into the specific QUBO formulations of ILP1 and ILP2, let us discuss the general approach for converting an ILP to a QUBO. An ILP can be written in the following standard form:

$$\text{minimize } \boldsymbol{c}^T\boldsymbol{x} \qquad (8a)$$

$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \qquad (8b)$$

$$\boldsymbol{x} \in \{0,1\}^n, \qquad (8c)$$

where $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, and $\boldsymbol{b} \in \mathbb{R}^m$.

To convert the ILP into a QUBO, we first transform each inequality constraint into an equality constraint by introducing slack variables. For each constraint $\boldsymbol{a}_i^T\boldsymbol{x} \leq b_i$, where $\boldsymbol{a}_i^T$ is the i-th row of matrix $\boldsymbol{A}$, we introduce a non-negative slack variable $s_i \geq 0$ such that:

$$\boldsymbol{a}_i^T\boldsymbol{x} + s_i = b_i. \qquad (9)$$

If the slack variable s_i is not restricted to be binary, we can represent it using a binary encoding. Let $s_i = \sum_{j=0}^{l_i} 2^j y_{ij}$, where $y_{ij} \in \{0,1\}$ and $l_i = \lfloor \log_2(b_i - \min \boldsymbol{a}_i^T\boldsymbol{x}) \rfloor$. The equality constraint becomes:

$$\boldsymbol{a}_i^T\boldsymbol{x} + \sum_{j=0}^{l_i} 2^j y_{ij} = b_i. \qquad (10)$$

Next, we apply the penalty method to convert the equality-constrained problem into an unconstrained QUBO. For each

equality constraint, we introduce a penalty term in the objective function:

$$P_i \left(\boldsymbol{a}_i^T \boldsymbol{x} + \sum_{j=0}^{l_i} 2^j y_{ij} - b_i \right)^2, \tag{11}$$

where $P_i > 0$ is a penalty coefficient. The penalty terms are designed such that they are minimized when the corresponding constraints are satisfied. The resulting unconstrained optimization problem is a QUBO.

C. QUBO Formulation of ILP1

To convert ILP1 into a QUBO, we introduce penalty terms for each constraint and add them to the objective function. Let us go through the constraints one by one.

a) Constraint (1a)*:* $d_{ij} \leq s_i + s_j, \quad (i,j) \in E$.

We can transform this inequality constraint into an equality constraint by introducing a binary slack variable y_{ij}:

$$d_{ij} - s_i - s_j + y_{ij} = 0, \quad (i,j) \in E. \tag{12}$$

The corresponding penalty term in the QUBO is:

$$P_1 \sum_{(i,j)\in E} (d_{ij} - s_i - s_j + y_{ij})^2. \tag{13}$$

However, we can eliminate the need for the slack variable y_{ij} by observing that when $s_i = s_j = 1$, the constraint is always satisfied regardless of the value of d_{ij}. Therefore, we can rewrite the penalty term as:

$$P_1 \sum_{(i,j)\in E} (d_{ij} - d_{ij}s_i - d_{ij}s_j + s_i s_j). \tag{14}$$

b) Constraint (3)*:* $d_{ij} + d_{jk} \geq d_{ik}, \quad \forall i \neq j, k \in N(i)$.

We can rewrite this constraint as:

$$d_{ij} + d_{jk} + (1 - d_{ik}) \geq 1, \quad \forall i \neq j, k \in N(i). \tag{15}$$

To convert this inequality into a QUBO penalty term, we can use the following transformation for $x + y + z \geq 1$:

$$xy + yz + zx + 2w - (1 + w)(x + y + z), \tag{16}$$

where w is an auxiliary binary variable. Applying this transformation to our constraint, we get:

$$P_2 \sum_{i\neq j} \sum_{k\in N(i)} \Big(d_{ij}d_{jk} + d_{jk}(1 - d_{ik}) + (1 - d_{ik})d_{ij} + 2y_{ijk} - (1 + y_{ijk})(d_{ij} + d_{jk} + (1 - d_{ik})) \Big), \tag{17}$$

where y_{ijk} are auxiliary binary variables. This transformation reduces the number of slack variables needed compared to the direct conversion of the inequality into an equality.

c) Constraint (1c)*:* $\sum_{i=1}^n s_i \leq \mathrm{K}$.

We can convert this constraint into an equality by introducing $\lceil \log_2(\mathrm{K}+1) \rceil$ binary slack variables. However, we can simplify this by observing that there must exist an optimal solution where exactly K vertices are removed. Therefore, we can replace the inequality with an equality:

$$\sum_{i=1}^n s_i = \mathrm{K}. \tag{18}$$

The corresponding penalty term in the QUBO is:

$$P_3 \left(\sum_{i=1}^n s_i - \mathrm{K} \right)^2. \tag{19}$$

d) Constraint (1d)*:* $s_i \leq d_{ij}, \quad i \neq j$.

We can rewrite this constraint as:

$$s_i - s_i d_{ij} \leq 0, \quad i \neq j. \tag{20}$$

Observe that $s_i - s_i d_{ij} \geq 0$ always holds, so we can convert the inequality into an equality:

$$s_i - s_i d_{ij} = 0, \quad i \neq j. \tag{21}$$

The corresponding penalty term in the QUBO is:

$$P_4 \sum_{i\neq j} (s_i - s_i d_{ij}). \tag{22}$$

Putting everything together, the QUBO formulation of ILP1, denoted as QUBO1, is given by:

$$\begin{aligned}
\min\ & P' \sum_{i=1}^n \sum_{j\neq i} (1 - d_{ij}) && \textbf{(QUBO1)} \\
& + P_1 \sum_{(i,j)\in E} (d_{ij} - d_{ij}s_i - d_{ij}s_j + s_i s_j) && (23) \\
& + P_2 \sum_{i\neq j} \sum_{k\in N(i)} \Big(d_{ij}d_{jk} + d_{jk}(1 - d_{ik}) + (1 - d_{ik})d_{ij} && (24) \\
& \qquad + 2y_{ijk} - (1 + y_{ijk})(d_{ij} + d_{jk} + (1 - d_{ik})) \Big) && (25) \\
& + P_3 \left(\sum_{i=1}^n s_i - \mathrm{K} \right)^2 && (26) \\
& + P_4 \sum_{i\neq j} (s_i - s_i d_{ij}) && (27)
\end{aligned}$$

The specific improvements in the QUBO formulation, such as eliminating slack variables for constraint (1a) and using the special transformation for constraint (1b), help reduce the size of the QUBO matrix in terms of the number of variables and non-zero entries. The number of original variables in ILP1 is $n^2 + n$, where n is the number of vertices in the graph. The number of auxiliary variables introduced in the QUBO formulation is $O(n^2 d)$, where d is the maximum node degree, dominated by the y_{ijk} variables introduced for constraint (1b). Therefore, the total number of variables in QUBO1 is $O(n^2 d)$.

The number of non-zero entries in the QUBO matrix is also $O(n^2d)$, as each auxiliary variable y_{ijk} introduces $O(1)$ non-zero entries in the matrix.

D. QUBO Formulation of ILP2

The QUBO formulation of ILP2 follows a similar approach to that of ILP1, with the main difference being the conversion of constraint (5b). The penalty terms for constraints (5a), (5c), and (5d) remain the same as in QUBO1.

$$\min z = P' \sum_{i=1}^{n} \sum_{j \neq i} (1 - d_{ij}) \qquad \textbf{(QUBO2)}$$

$$+ P_1 \sum_{(i,j) \in E} (d_{ij} - d_{ij}s_i - d_{ij}s_j + s_i s_j) \tag{28}$$

$$+ P_2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \Big(- \sum_{k \in N(i)} (d_{kj}) + |N(i)| d_{ij} - |N(i)| s_i + \sum_{b=0}^{\lceil \log_2 2|N(i)| \rceil} 2^b y_{ij}^{(b)} \Big)^2 \tag{29}$$

$$+ P_3 \left(\sum_{i=1}^{n} s_i - K \right)^2 \tag{30}$$

$$+ P_4 \sum_{i \neq j} (s_i - d_{ij} s_i) \tag{31}$$

For constraint (5b), we introduce $O(n^2 \log d)$ binary slack variables $y_{ij}^{(b)}$, where n is the number of vertices in the graph and d is the maximum node degree This is because we need $\lceil \log_2(|N(i)| + 1) \rceil$ slack variables for each pair (i, j), and $|N(i)| \leq n - 1$ for all i. The resulting penalty term in the QUBO is

$$P_2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} \Big(\sum_{k \in N(i)} d_{kj} - |N(i)| d_{ij} + |N(i)| s_i + \sum_{b=0}^{\lceil \log_2(|N(i)|+1) \rceil} 2^b y_{ij}^{(b)} \Big)^2.$$

The size of QUBO2 can be analyzed similarly to QUBO1. The number of original variables remains $n^2 + n$. The number of auxiliary variables introduced in QUBO2 is $O(n^2 \log d)$, dominated by the $y_{ij}^{(b)}$ variables introduced for constraint (5b). Therefore, the total number of variables in QUBO2 is $O(n^2 \log d)$, which is a significant reduction compared to the $O(n^2 d)$ variables in QUBO1. The number of non-zero entries in the QUBO2 matrix is also $O(n^2 \log d)$.

In summary, the main difference between the QUBO formulations of ILP1 and ILP2 lies in the conversion of the triangle inequality constraint. In QUBO1, this constraint leads to $O(n^2 d)$ auxiliary variables and non-zero entries in the QUBO matrix. In QUBO2, the modified constraint results in only $O(n^2 \log d)$ auxiliary variables and non-zero entries. This reduction in the size of the QUBO matrix can lead to more efficient quantum annealing for the k-CND problem.

Theorem 3: For any connected graph $G = (V, E)$ with $|V| > 2$, the QUBO2 formulation (**QUBO2**) requires fewer binary variables than the QUBO1 formulation (**QUBO1**).

Proof: Let $n = |V|$ be the number of vertices in the graph G. Since we proved (5b) was valid for any arbitrary ordering of vertices, let us number the vertices in V in ascending order of their degrees, such that $V = \{v_1, ..., v_n\}$, and $|N(i)| >= |N(j)|$ for all $i >= j$. We will prove the theorem by induction, by first showing that QUBO1 requires fewer variables for our base case. Then we will prove that any augmentation of a connected graph which yields a connected graph and which satisfies the invariant that the number of variables for QUBO2 is larger than the number of variables for QUBO1 will yield a graph which retains this invariant.

For the base case, in which our graph is a tree and $|V| = 3$, the number of slack variables used for QUBO1 is 8 (we get 2 slack variables from each vertex of degree 1 and 4 slack variables from the vertex of degree 2). Using the careful ordering of vertices described, the number of slack variables for QUBO2 is 6 (4 and 2 slack variables from our first and second selected vertices of degree 1, respectively).

Now, suppose we are given a graph $G = (V, E)$, such that the number of variables for QUBO1 for G is greater than the number of variables used for QUBO2 for G. We will show that augmenting G, either by adding an edge from v_1 to v_2 for $v_1, v_2 \in V$ or by adding some vertex, u to V in a way which leaves G connected (adding some edge $(u, v_k)|v_k \in V$), will yield a graph for which the number of variables for QUBO1 for G is greater than the number of variables used for QUBO2 for G.

1) Adding an edge between $v_1, v_2 \in V$ adds $2|V|$ slack variables to QUBO1 ($|V|$ for each vertex incident to the new edge), and since $\lceil \log_2 2x \rceil + 1 >= \lceil \log_2 2(x+1) \rceil, \forall x \in \mathbb{N}$, the maximum number of slack variables added for each edge added in QUBO2 is $2|V|$ ($|V|$ slack variables for each vertex incident to the new edge).
2) Adding a vertex, u, and an edge $(u, v_k)|v_k \in V$, to connect u to some other vertex in the network adds at least as many variables to QUBO1 as it does to QUBO2. Note that we only need to worry about the new vertex and edge incident to it and not the other endpoint of our new edge, since we already proved that adding another edge incident to a vertex adds at least as many variables to QUBO1 as to QUBO2. Adding a new vertex and an edge to exactly 1 neighbor will increase the number of variables for QUBO2 by at most $|V| - 1$, since by our definition for adding a vertex. This is because u must be of degree 1, and because of our ordering of vertices by degree in ascending order, a vertex of degree one will be the first vertex (WLOG, u can be this first vertex), which then contributes $|V| - 1$ vertices. Adding a vertex for QUBO1 also clearly increases number of variables by at least $|V|$.

Hence, since we proved for a base case of a tree of 3 vertices that QUBO1 yields fewer variables than QUBO2 and showed that any augmentation to a connected graph which has the invariant that the number of variables for QUBO1 is greater than the number of variables for QUBO2, which yields a connected graph must retain this invariant, we have shown

that any connected graph, $G = (V, E)$ for $|V| >= 3$ must use fewer variables for QUBO2.

■

The major reduction in QUBO size is due to the modified constraint (5b) in ILP2 allows for a more compact representation of the triangle inequality, resulting in fewer auxiliary variables when converting to QUBO. The asymptotic analysis shows that the number of variables in QUBO2 grows much slower than in QUBO1 as the graph size increases. In the next section, we will also show empirically that QUBO2 is always more compact than QUBO1.

IV. Experiments

Computational experiments were run by testing QUBO1 and QUBO2 with different solvers on random graphs of varying sizes. The code for each approach was written in Github Codespaces, using an API key provided by D-Wave, in the python programming language. We ran two experiments; one using $|V|$ as the independent variable (while fixing $\frac{|E|}{|V|}$), and one using $|E|$ as the independent variable. Each set of parameters was tested by constructing 5 graphs using the networkx library and using our solvers with QUBO1 and QUBO2.

a) Parameter settings: We set the number of vertices with a default value of 10, the number of edges with a default based on an average vertex degree of 3, graph topology with a default setting of a random graph, the number of vertices to be removed (denoted as K) with a default of 3, penalty coefficient which defaulted to the coefficient provided by D-Wave, number of reads with a default of 1000, and chain strength which also defaulted to the strength provided by D-Wave. We also chose to use penalty coefficient of $|V|^3$ for the constraint which enforces at most K vertices chosen to be removed (5b) and $|V|$ for the other constraints. These penalties were chosen, because we would like to keep penalties small and integral, as choosing larger, fractional penalties causes solution quality to degrade. The penalty for the QUBO term for constraint (5b) was chosen to be sufficiently large to ensure no more than K vertices are removed, which allows us to perform error correction even on infeasible solutions. This error correction is conducted by finding the pairwise connectivity of the $|V| - K$ unremoved vertices via a BFS on the induced subgraph.

A. Overview of Approaches

We use three different solvers to help compare our QUBO formulations (Quantum Annealing, Simulated Annealing, Hybrid), as well as three more classical approaches to serve as benchmarks for the quantum approaches (CPLEX, random, greedy).

Quantum Annealing is an approach to solving optimization problems via the use of the adiabatic theorem [33]. Here, we feed our QUBO coefficient matrix through the QA solver from d-wave to obtain solutions. Note that a major (and time consuming) challenge with this step is properly embedding the problem onto the hardware; here we make use of D-Wave libraries to perform embedding.

The hybrid approach is an approach to finding the optimal solution to optimization problems by mixing classical and quantum computing as discussed. The software written for this project passes the QUBO to this hybrid solver (LeapHybrid-Sampler with default parameters) to solve.

Simulated annealing approximates solutions to optimization problems using classical computing behaving like heating and cooling crystalline solids in order to obtain a lattice with the lowest energy [34]. The goal of SA is to find a globally optimal solution by using classical computing and heuristics to obtain the lowest energy state, analogous to the energy state found by a quantum computer. Even without finding a global optimum, SA converges to a locally optimal solution [35]. SA is applied to the QUBO using D-Wave Simulated Annealing Sampler.

The CPLEX algorithm is used to determine the optimal solution for the ILP since it always obtains an optimal solution (so long as the gap is chosen to be small enough such that the entire solution space is checked by CPLEX). Using the docplex library, we add the constraints and objective function to an assignment model and solve the CPLEX for the IP. CPLEX is also used to help verify the correcness of the QUBO formulations by transforming the QUBOs back into MIP formulations and feeding them into CPLEX to verify the optimal solution is found.

The randomized algorithm, which chooses K random vertices to remove then runs a BFS to determine remaining pairwise connectivity, is used as a benchmark.

The greedy algorithm, which iteratively chooses K vertices, picking the vertex remaining with the highest degree, is a benchmark for the other algorithms. As with randomized, a BFS is used to determine the graph's pairwise connectivity.

B. Metrics Measured

The datasets were constructed by using the networkx library. For each experiment, we construct a new graph. We run 5 trials for each independent variable and present the results by displaying the median and range for the network disruption in Table I. We also display the median for the number of qubits, QUBO variables, and QUBO terms used via graphs Figs. 1a–f. For each trial, we calculate:

1) **QUBO Variables** - The number of qubits which would be used to represent the QUBO for QA if each qubit had a chain of length 1. This is the number of distinct variables (eg. d_{00}) used in the QUBO formulation.
2) **Number of qubits** - The number of qubits used by D-Wave after embedding the QUBO for QA.
3) **QUBO Terms** The number of total terms (and the number of quadratic terms) in the constructed QUBO. (We do not bother to include the number of linear terms, as from our specific QUBO constructions this is just equivalent to the number of QUBO variables).
4) **OPT** - The minimum value of $C(G)$ for the given graph after removing K vertices.
5) $\boldsymbol{OPT - C(G*)}$ - The difference between the minimum possible pairwise connectivity, $C(G)$, for a graph G, after removing K vertices (this optimal solution is found via CPLEX) and the connectivity of the resulting graph,

$G*$, after applying one of the methods of QA, SA, hybrid, greedy, or random. Note that in cases where solutions are invalid, by choosing penalties carefully to ensure that it is least favorable to violate constraint (1c), we ensure that we are able to reconstruct a feasible solution from the chosen vertices and use the subgraph induced by the removal of K vertices for our connectivity difference measurement.

C. Results

1) Solution Size: As can be seen from Figs. 1a–d, the number of variables and qubits both grow for QUBO1 and QUBO2 as the network size increases. As proven in Theorem 3, the number of variables for every graph is greater for QUBO1 than QUBO2, and from Fig. 1d, there is a 59.7% decrease in the number of variables used for a graph of 10 vertices and 40 edges. While QUBO2 also used fewer qubits in all experiments run (Fig. 1a, Fig. 1b), the difference in the number of qubits is smaller than the difference in variables for the two formulations. This discrepancy is caused by longer chain lengths for QUBO2. In other words, each variable is being represented by more physical qubits, on average for QUBO2 than QUBO1. The two principal causes for this are:

1) The coefficients for the QUBO2 terms (especially for the slack variables) are larger than for the QUBO1 terms.
2) The greater number of QUBO terms for QUBO2 than QUBO1. We see this in the results from Fig. 1e and Fig. 1f, where the number of quadratic terms and total QUBO terms for QUBO2 are greater than QUBO1. We can see why by counting the number of QUBO terms from constraint (3) for QUBO1 and (5b) for QUBO2 since these are the constraints which contribute the most QUBO terms for sufficiently large graph instances:

QUBO1 The number of QUBO terms for **QUBO1** from constraint (3) is $O(n^2 d)$, where n is the number of vertices, and where d is the maximum node degree.

QUBO2 We obtain the upper bound from counting the number of terms from (5b) from the highest order term. Counting the number of terms each pair of vertices, $i < j \in V$, such that $i \neq j$, we get $\binom{|N(i)|+\log_2|N(i)|+1}{2}$ terms; hence, there are $O(|N(i)|^2)$ terms for each i, j, and there are $\sum_{i=1}^{n}\sum_{j=i+1}^{n}|N(i)|^2$ total highest order terms for QUBO2. As analysis from de Caen [36] has shown, the sum $\sum_{i=1}^{n}|N(i)|^2$, known as the first Zagreb index is bounded by $\frac{|E|^2}{|V|}$. Hence, we can bound the number of variables for QUBO2 as $O(|V| * \frac{|E|^2}{|V|})$, or, $O(|E|^2)$.

Clearly, for sufficiently large, dense graphs, the number of variables for QUBO2 exceeds the number required for QUBO1, since there are $|N(i)|$ vertices for every $i, j \in V$ for QUBO1 and $|N(i)|^2$ vertices for every $i, j \in V$ for QUBO2.

Despite the chain strengths being longer, experimentally we witness that the benefit from QUBO2 having fewer variables

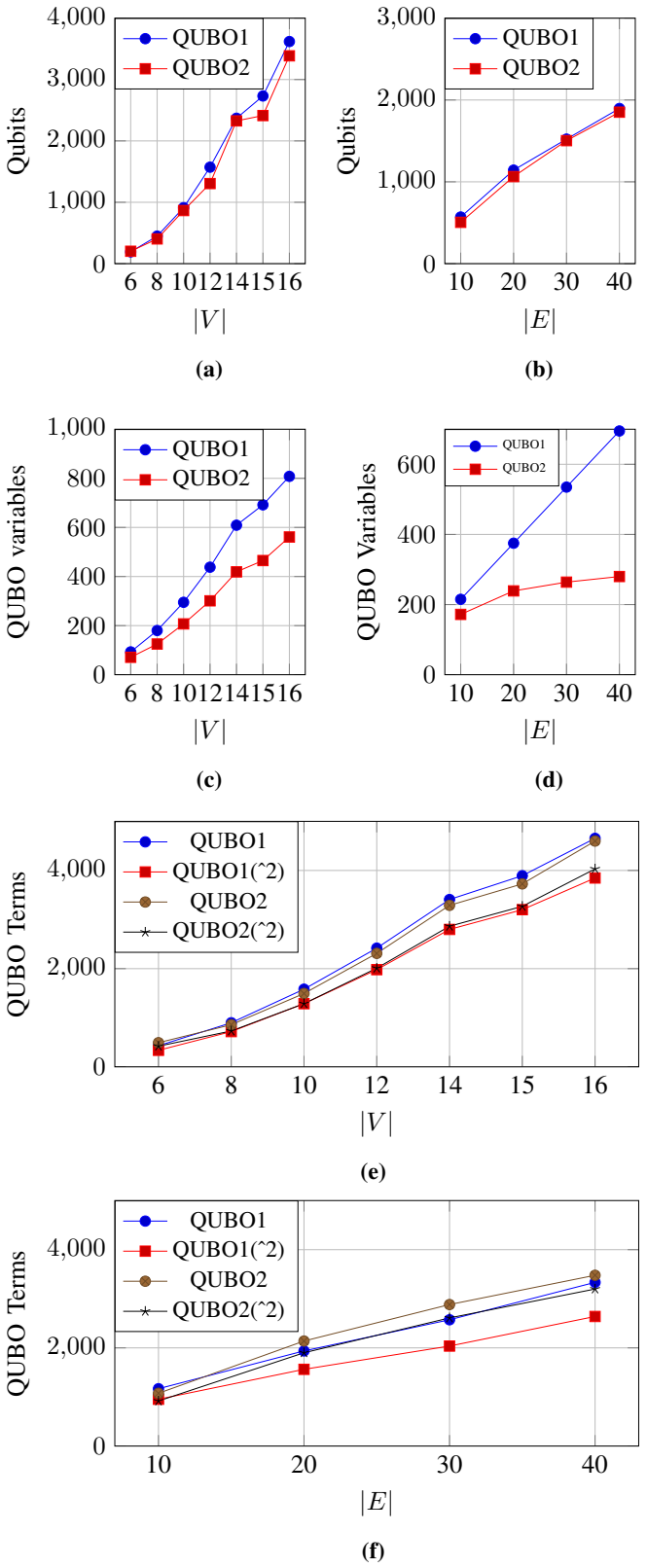

Fig. 1: Comparison of QUBO and embedding properties for QUBO1 and QUBO2 on random graphs of varying numbers of vertices and edges

TABLE I: Comparison of Network Disruption produced by Solvers For Varying $|V|$

		QUBO1			**QUBO2**			**GREEDY**	**RANDOM**	**CPLEX**
		QA	Hybrid	SA	QA	Hybrid	SA	—	—	—
		$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT}{OPT}$
$\|V\|$	5	**1**(0-1)	**0**(0-0)	**1**(0-1)	**1**(0-1)	**0**(0-1)	**1**(0-1)	**0**(0-1)	**1**(0-1)	**0**(0-0)
	10	**21**(11-21)	**4**(3-10)	**21**(6-21)	**15**(11-21)	**10**(3-15)	**21**(10-21)	**4**(3-21)	**21**(9-21)	**0**(0-0)
	15	**52**(50-75)	**42**(25-50)	**50**(33-63)	**50**(26-52)	**33**(17-51)	**51**(25-63)	**33**(19-51)	**50**(26-52)	**3**(3-5)

TABLE II: Comparison of Network Disruption produced by Solvers For Varying $|E|$

		QUBO1			**QUBO2**			**GREEDY**	**RANDOM**	**CPLEX**
		QA	Hybrid	SA	QA	Hybrid	SA	—	—	—
		$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT-C(G*)}{OPT}$	$\frac{OPT}{OPT}$
$\|E\|$	10	**11**(6-15)	**2**(1-2)	**2**(1-2)	**6**(1-11)	**2**(1-7)	**2**(1-7)	**2**(1-6)	**7**(6-15)	**0**(0-0)
	20	**20**(15-28)	**15**(14-20)	**15**(14-20)	**20**(15-21)	**20**(9-21)	**20**(14-21)	**15**(14-20)	**20**(15-21)	**1**(0-1)
	30	**17**(17-18)	**17**(12-18)	**17**(11-18)	**11**(3-17)	**11**(7-18)	**17**(12-18)	**17**(12-18)	**17**(12-18)	**4**(3-4)

outweighs the cost in the number of terms and larger coefficients, as the number of qubits was smaller for QUBO2 in every trial, with as much as an 11.7% decrease in for the network tested with 15 vertices, 22 edges.

2) Solution Quality: From Table I, we notice that solution quality degrades for both $QUBO1$ and $QUBO2$ as vertices are added to the network, since $OPT - C(G*)$ increases as $|V|$ increases for all solvers for both $QUBO1$ and $QUBO2$.

While solution quality for QA and Hybrid solvers does degrade for larger graph instances, this result can also partially be explained by having a stochastic process in which there are more variables (and the QUBO size squares quadratically as the number of vertices increases linearly). Let us show this with an example. Suppose we are given a graph, $G = (V, E)$, with optimal solution A which partitions V into $x > 1$ disjoint sets of an equal number of vertices and suboptimal solution B which partitions V into one large, disjoint subset (of size $|V| - K$. If we augment G to G' by adding y vertices to the graph, even if they all go into the same partition for a solution for A, the difference between B in G' and A in G' will be much larger than the difference between B in G and A in G, because the difference between $(\frac{|V|-K}{x} + y)^2$ and $(|V|-K+y)^2$ is larger than the difference between $(\frac{|V|-K}{x})^2$ and $(|V|-K)^2$ for $y > 0$. Hence, we see an example of where the sheer quantity of a suboptimal solution may grow further from the optimal solution because $|V|$ is larger. Nevertheless, the connectivity of the graphs from greedy and random grow larger than CPLEX quickly, and at times, Hybrid solutions can outperform greedy solutions and seem to perform better than random ones. Note that there are some limitations with larger numbers of vertices for the quantum approaches, as with $|V| = 15$, we do see the greedy approach producing better results than the quantum approaches, and for $|V| = 20$, the problem size for QA became too large to embed onto the hardware at all.

Another challenge with increasing the number of vertices is that the penalties used for the QUBO are proportional to $|V|$, and, as the penalties increase, obtaining an exact solution becomes more difficult due to a limit to the precision of D-Wave computers and the stochastic nature of solutions. As a result, 2 of the solutions obtained for the Hybrid approach for QUBO2 are invalid, as they select more then K vertices to remove from the graph.

From Table II, the error for the solutions seems to decrease from the average degree increasing from 4 to 6 for random, greedy, SA, and hybrid. This is due to the fact that for a very dense graph, removing any 3 vertices will yield the same solution; hence every solution is optimal. One noticeable exception above, however, is with QA, which seems to be producing somewhat suboptimal solutions regardless. This is due to the fact that QA is a stochastic process and because the additional edges cause greater difficulty in finding an embedding. As a result, for some of these suboptimal solutions, the QA solver chooses fewer than K vertices.

V. Conclusion and Discussion

The paper shows that using quantum computing to approach the k-CND problem does have some merits, as for large instances of the problem, finding the optimal solution via CPLEX becomes difficult and the code may take many hours to run. In testing the two different ILP formulations, it seems that the theoretical results from Theorem 3 agree with the experimental results that **ILP2** is better suited for transformation into a QUBO in that there are fewer QUBO variables generated for **ILP2**. However, the larger number of terms for QUBO2 and long chains necessary for embedding mean that there is a smaller difference in the number of qubits used for both QUBO formulations in solving the k-CND problem, although the number of qubits was consistently slightly lower for QUBO2 (and as much as an 11.7% decrease in the number of qubits used for $|V| = 15$). The chain lengths necessary for embedding (**QUBO2**) may pose a limitation on this formulation, since longer chain lengths have higher incidences of chain breaks, increasing the chance of producing suboptimal solutions.

There also exist some limitations to the feasibility of using QA to solve the k-CND problem as the QA approach was unable to embed problems of size more than $|V| \geq 20$. The time required to obtain optimal solutions is currently a limitation of using QA for the k-CND problem for any time-sensitive systems.

Future work on the k-CND problem may seek to both find better QUBO penalties and increase the number of vertices and edges used within the graphs, while focusing more on using the Hybrid solver, which is able to support larger graph instances, and may leverage both the benefits of classical computing and quantum phenomena. Future experiments can modify the chain strength when performing QA to mitigate issues with broken chains in embedding (**QUBO2**). Further research can aim to improve on (**QUBO2**) further by reducing the number of variables from constraint (5b).

The approach used in this paper of constructing an alternative ILP formulation in order to reduce the number of QUBO variables in the QUBO formulation between constraints may be applicable to other optimization problems as well; hence future work which converts ILP problems to QUBOs should be sure to consider alternative ILP formulations.

ACKNOWLEDGEMENT

This work was supported in part by NSF CNS-1814614, NSF IIS-1908594, NSF AMPS-2229075, VCU Quest Award, and a Commonwealth Cyber Initiative (CCI) award.

REFERENCES

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.

[2] Duncan S Callaway, Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85(25):5468, 2000.

[3] Thang N Dinh, Nam P Nguyen, and My T Thai. A new approach to the critical node problem in social networks. *Handbook of Optimization in Complex Networks*, pages 3–31, 2012.

[4] Thang N Dinh and My T Thai. Network under joint node and link attacks: Vulnerability assessment methods and analysis. *IEEE/ACM Transactions on Networking*, 23(3):1001–1011, 2014.

[5] Thang N Dinh, Ying Xuan, My T Thai, EK Park, and Taieb Znati. On approximation of new optimization methods for assessing network vulnerability. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[6] Yilin Shen, Nam P. Nguyen, Ying Xuan, and My T. Thai. On the discovery of critical links and nodes for assessing network vulnerability. *IEEE/ACM Transactions on Networking*, 21(3):963–973, 2013.

[7] Jian-Wei Wang and Li-Li Rong. Cascade-based attack vulnerability on the us power grid. *Safety Science*, 47(10):1332–1336, 2009.

[8] Daniel Ortiz-Arroyo and Danish Ali Hussain. Assessing the vulnerability of social networks to breakdowns. *International Journal of Information Technology & Decision Making*, 2(03):449–469, 2003.

[9] Hawoong Jeong, Sean P Mason, Albert-László Barabási, and Zoltán N Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[10] Alexander Veremyev, Konstantin Pavlikov, Eduardo L Pasiliao, My T Thai, and Vladimir Boginski. Critical nodes in interdependent networks with deterministic and probabilistic cascading failures. *Journal of Global Optimization*, 74:803–838, 2019.

[11] Bound and exact methods for assessing link vulnerability in complex networks. *Journal of Combinatorial Optimization*, 28:3–24, 2014.

[12] Thang N Dinh and My T Thai. Precise structural vulnerability assessment via mathematical programming. In *2011-MILCOM 2011 military communications conference*, pages 1351–1356. IEEE, 2011.

[13] Thang N Dinh and My T Thai. Assessing attack vulnerability in networks with uncertainty. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2380–2388. IEEE, 2015.

[14] Wayne Pullan. Optimisation of a hybrid hub-and-spoke and point-to-point pipeline network. *European Journal of Operational Research*, 240(1):237–248, 2015.

[15] Roberto Aringhieri and Federico Malucelli. An evolutionary approach for the global optimisation of hybrid pipeline networks. *Optimization Methods and Software*, 31(6):1261–1281, 2016.

[16] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[17] Lov K Grover. A fast quantum mechanical algorithm for database search. pages 212–219, 1996.

[18] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[19] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.

[20] Tameem Albash and Daniel A Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018.

[21] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[22] Sergey Bravyi, Andrew W Cross, Jay M Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627(8005):778–782, 2024.

[23] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, Richard Harris, Andrew J Berkley, Jan Johansson, Paul Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.

[24] Hartmut Neven, Vasil S Denchev, Geordie Rose, and William G Macready. Nips 2009 demonstration: Binary classification using hardware implementation of quantum annealing. In *Presented at NIPS 2009: 23rd Annual Conference on Neural Information Processing Systems*, volume 9, pages 1–17, 2009.

[25] Florian Neukart, Gabriele Compostella, Christian Seidel, David Von Dollen, Sheir Yarkoni, and Bob Parney. Traffic flow optimization using a quantum annealer. *Frontiers in ICT*, 4:29, 2017.

[26] Gili Rosenberg, Poya Haghnegahdar, Phil Goddard, Peter Carr, Kesheng Wu, and Marcos Lopez De Prado. Solving the optimal trading trajectory problem using a quantum annealer. *IEEE Journal of Selected Topics in Signal Processing*, 10(6):1053–1060, 2016.

[27] Andrew D King, Jack Raymond, Trevor Lanting, Richard Harris, Alex Zucca, Fabio Altomare, Andrew J Berkley, Kelly Boothby, Sara Ejtemaee, Colin Enderud, et al. Quantum critical dynamics in a 5,000-qubit programmable spin glass. *Nature*, 617(7959):61–66, 2023.

[28] Andrew D King, Alberto Nocera, Marek M Rams, Jacek Dziarmaga, Roeland Wiersema, William Bernoudy, Jack Raymond, Nitin Kaushal, Niclas Heinsdorf, Richard Harris, et al. Computational supremacy in quantum simulation. *arXiv preprint arXiv:2403.00910*, 2024.

[29] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in Physics*, 2, 2014.

[30] Dorabella Santos, Amaro de Sousa, and Paulo Monteiro. Compact models for critical node detection in telecommunication networks. *Electronic Notes in Discrete Mathematics*, 64:325–334, 2018. 8th International Network Optimization Conference - INOC 2017.

[31] Fred Glover, Gary Kochenberger, Rick Hennig, and Yu Du. Quantum bridge analytics i: a tutorial on formulating and using qubo models. *Annals of Operations Research*, 314(1):141–183, 2022.

[32] Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, 7:193–209, 2008.

[33] Satoshi Morita and Hidetoshi Nishimori. Mathematical foundation of quantum annealing. *Journal of Mathematical Physics*, 49(12):125210, dec 2008.

[34] Darrall Henderson, Sheldon H. Jacobson, and Alan W. Johnson. *The Theory and Practice of Simulated Annealing*, pages 287–319. Springer US, Boston, MA, 2003.

[35] Yaroslav Koshka and Mark A. Novotny. Comparison of d-wave quantum annealing and classical simulated annealing for local minima determination. *IEEE Journal on Selected Areas in Information Theory*, 1(2):515–525, 2020.

[36] D. de Caen. An upper bound on the sum of squares of degrees in a graph. *Discrete Mathematics*, 185(1):245–248, 1998.