Template-based Minor Embedding for Adiabatic Quantum Optimization

Thiago Serra[1], Teng Huang[2], Arvind Raghunathan[3], and David Bergman[4]

[1]Bucknell University
[2]Lingnan (University) College, Sun Yat-sen University
[3]Mitsubishi Electric Research Laboratories
[4]University of Connecticut

Abstract

Quantum Annealing (QA) can be used to quickly obtain near-optimal solutions for Quadratic Unconstrained Binary Optimization (QUBO) problems. In QA hardware, each decision variable of a QUBO should be mapped to one or more adjacent qubits in such a way that pairs of variables defining a quadratic term in the objective function are mapped to some pair of adjacent qubits. However, qubits have limited connectivity in existing QA hardware. This has spurred work on pre-processing algorithms for embedding the graph representing problem variables with quadratic terms into the hardware graph representing qubits adjacencies, such as the Chimera graph in hardware produced by D-Wave Systems. In this paper, we use integer linear programming to search for an embedding of the problem graph into certain classes of minors of the Chimera graph, which we call *template embeddings*. One of these classes corresponds to complete bipartite graphs, for which we show the limitation of the existing approach based on minimum Odd Cycle Transversals (OCTs). One of the formulations presented is exact, and thus can be used to certify the absence of a minor embedding using that template. On an extensive test set consisting of random graphs from five different classes of varying size and sparsity, we can embed more graphs than a state-of-the-art OCT-based approach, our approach scales better with the hardware size, and the runtime is generally orders of magnitude smaller.

1 Introduction

Quantum Annealing (QA) is a technique based on a system of quantum particles in which the Hamiltonian—an operator corresponding to the sum of potential and kinetic energies—is modeled after the objective function of a binary optimization problem (Finnila et al., 1994; Kadowaki and Nishimori, 1998). Adiabatic Quantum Computation (AQC) is a particular form of QA that exploits the adiabatic theorem of quantum physics (Farhi et al., 2000, 2001), which states that a quantum system in the ground state evolves predominantly in the ground state—i.e., at the lowest possible energy level—if the rate at which the system changes is sufficiently small. Hence, AQC can approximately solve unconstrained optimization problems by slowly evolving a quantum system from a configuration in known ground state toward a configuration in which the ground state corresponds to a solution of minimum value for the optimization problem.

It is possible to model many combinatorial optimization problems with such a technique (Lucas, 2014). However, it is not known if the quantum system can always evolve in polynomial time on the inputs in order to reach a quantum speedup—i.e., solve a problem much faster than any classical algorithm would (Rønnow et al., 2014). For example, there are cases in which a linear transformation between quantum states requires exponential time (Van Dam et al., 2001). Nevertheless, for some problems the quantum system evolves efficiently (Farhi et al., 2000) or at least faster than classical algorithms (Lucas, 2018). When compared with commercial optimization software, AQC has been found to perform better in one case (McGeoch and Wang, 2013) and comparable in another (Coffrin et al., 2019). Since the solutions obtained through AQC are sometimes suboptimal, Dash (2013) observes that AQC is arguably comparable to heuristics that can find optimal solutions with high probability. Hence, as this technology matures, its competitive advantage will depend on the speed of convergence as well as on being more reliable than such heuristics.

AQC has been used to approximately solve Quadratic Unconstrained Binary Optimization (QUBO) problems of the form

$$\min_{x\in\{0,1\}^n} x^T Q x \qquad \left(\text{or} \qquad \max_{x\in\{0,1\}^n} x^T Q x\right)^{1},$$

where $Q \in \mathbb{R}^{n \times n}$ are input data and x are decision variables.[2] Without loss of generality, let Q be an upper-triangular matrix. In this paper, we will refer to any device implementing AQC as a *quantum annealer*, or a QA hardware.

Although existing QA hardware may potentially be used on problems with as many as 2048 variables, in practice only problems that are much smaller or substantially sparse can be directly formulated. To circumvent this limitation to some extent, we may use a surrogate formulation in which the optimal solutions can be mapped to optimal solutions of the original problem. For defining such formulation, it is often necessary to analyze the structure of the QA hardware and of the optimization problem that we want to solve. That entails determining if the graph associated with the problem is a minor of the graph associated with the QA hardware, which is an NP-complete problem in general (Johnson, 1987).

In this paper, we show that Integer Linear Programming (ILP) can be effectively used for this preprocessing step, in which we determine how a QUBO problem can be modeled in QA hardware if the qubits have limited connectivity. We propose ILP formulations based on particular minors of the QA hardware, which we denote *template embeddings*.

We introduce the minor embedding problem for QA hardware and previous approaches in Section 2, and then contextualize our contribution in Section 3. We describe the template embeddings, their properties, and corresponding ILP formulations in Sections 4 and 5. We present experimental results in Section 6 and final remarks in Section 7.

2 Background

In this section we describe the graph embedding problem associated with QA hardware, in particular with Chimera graphs, and the previous approaches to this problem.

[1] We regard the maximization version, which is preferred by some authors, as interchangeable with the minimization version by negating Q. For that reason, we only consider the minimization version for the rest of the paper.

[2] According to Boros and Hammer (2002), QUBO is also known as a quadratic pseudo-Boolean function in the literature since Kalantari (1986).

2.1 Hardware and Problem Graphs

The quantum annealer solves the Ising formulation

$$\min_{y \in \{-1,1\}^n} y^T J y + h^T y,$$

where $J \in \mathbb{R}^{n \times n}$ and $h \in \mathbb{R}^n$ are input data and y are decision variables. Each binary variable y_i is associated with a *qubit* i, the basic unit of quantum information, and its value corresponds to the *magnetic spin* of the quantum transistor that physically implements the qubit (D-Wave Systems, 2019). The linear coefficient h_i for each variable y_i corresponds to the *bias* of qubit i. The quadratic term J_{ij} for each pair of variables y_i and y_j, if nonzero, implies the existence of a *coupler* between qubits i and j, and the value of J_{ij} represents the *strength* of the coupler. For each QUBO formulation, there is a corresponding Ising formulation for which the variables with values -1 and 1 in an optimal solution of the Ising problem correspond to variables with values 0 and 1 in an optimal solution of QUBO, which is such that the zero elements in Q are also zero in J (Choi, 2008). Hence, we may assume that matrices Q and J have the same nonzero elements.

In practice, the hardware graph has a sparse structure. Due to engineering limitations, each qubit can only be coupled with a limited set of other qubits (Choi, 2008). That implies that we can only directly solve problems with as many variables as the number of qubits if these problems follow the same sparsity structure as the QA hardware.

A QA hardware can be modeled as an undirected graph in which the vertices correspond to qubits and the edges to pairs of coupled qubits. We denote it as the *hardware graph* H. Similarly, we consider a *problem graph* G in which the vertices correspond to decision variables of an Ising formulation and the edges to pairs of variables with a quadratic term in the objective function. Let $G = (V(G), E(G))$, where $V(G) := \{v_1, v_2, \ldots, v_n\}$ is the set of vertices of G and $E(G)$ is a set of edges in which $\{v_i, v_j\} \in E(G)$ if $J_{i,j} \neq 0$. Similarly, let $H = (U(H), F(H))$, where vertex $u_i \in U(H)$ corresponds to qubit i and $\{u_i, u_j\} \in F(H)$ implies that qubits i and j are coupled. For convention, we will use V_i for a subset of $V(G)$ and U_i for a subset of vertices of H or any of its minors.

A problem can be directly solved in QA hardware if there is a subgraph H' of H that is *isomorphic* to G, i.e., there is a bijective mapping between the vertices of G and H' such that adjacent vertices in G are mapped to adjacent vertices in H'. However, this greatly limits the class of problems that can be solved if the vertices of H have small degrees.

The class of solvable problems can be enlarged by allowing each vertex of the problem graph G to be mapped to possibly multiple vertices in the hardware graph H. This imposes the additional requirement that qubits associated with a vertex in G have the same spin in the ground state. For example, two coupled qubits corresponding to vertices u_i and u_j can be induced to have the same spin in the ground state if J_{ij} is negative and sufficiently large in absolute value (Kaminsky and Lloyd, 2004; Kaminsky et al., 2004). In that case, those *physical* qubits define a single *logical* qubit. The multiplicity of physical qubits increases the number of neighbors and make it possible to embed a graph G with higher connectivity than that present in H. More generally, we say that G can be embedded in a hardware graph H if G is a *minor* of H (Choi, 2008). A minor of a graph is any graph that can be obtained by a sequence of vertex and edge deletions as well as edge contractions (Bondy and Murty, 2008). In the example above, contracting edge $\{u_i, u_j\}$ produces a graph in which vertices u_i and u_j are replaced by a vertex u' that is adjacent to any vertex that was originally adjacent to either u_i or u_j. Hence, the *embedding* of G in H consists of assigning each vertex $v_i \in V(G)$ to a distinct set of vertices $U_i \subseteq U(H)$ such that the induced subgraph on U_i is connected and, for each edge $\{v_i, v_j\} \in E(G)$, there exists $u_k \in U_i$ and $u_l \in U_j$ such that $(u_k, u_l) \in F(H)$.

In the QAs that are produced and that are currently made commercially available by D-Wave Systems, the hardware graph follows the structure of a *Chimera graph*. Let us denote it as a graph $C_{M,N,L}$ such that $2MNL$ vertices are distributed in a grid of MN cells. Each cell contains $2L$ vertices. Each of those cells is a complete bipartite graph $K_{L,L}$, where a left and a right partition each contain L vertices. For ease of explanation, let us number the vertices in each partition from 1 to L. The i-th vertex in each left (right) partition is also adjacent to the corresponding i-th vertex of the left (right) partition in the cell above (to the left) and below (to the right). Figure 1 depicts $C_{2,2,4}$.

The following hardware graphs have been used in QA hardware: $C_{4,4,4}$ in D-Wave One, $C_{8,8,4}$ in D-Wave Two, $C_{12,12,4}$ in D-Wave 2X, and $C_{16,16,4}$ in D-Wave 2000Q (Dattani et al., 2019). In all of those cases, the maximum degree of a vertex is 6, which makes the problem of finding minor embeddings crucial to leverage such AQC hardware. Since $M = N$ in all cases, we will follow the convention of considering Chimera graphs of the form $C_{M,M,L}$.

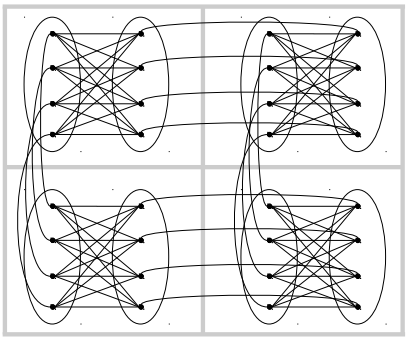

Figure 1: Chimera graph $C_{2,2,4}$ with a $K_{4,4}$ on each cell of a 2×2 grid.

2.2 Minor Embedding Algorithms

Early work on minor embedding into hardware graphs has focused on complete graphs (or *cliques*). A clique K_n is a graph on n vertices in which all vertices are adjacent to one another. If K_n can be embedded into a hardware graph H, then any other graph G with at most n vertices can be embedded in the same hardware graph H, since G is isomorphic to a subgraph of K_n. The TRIAD algorithm was the first technique to embed cliques in hardware graphs with limited connectivity of the qubits (Choi, 2011).

The TRIAD algorithm associates each vertex of the problem graph with a chain of vertices of the hardware graph that is long enough to have at least one vertex that is adjacent to some vertex of all other chains. These chains can be embedded into a Chimera graph, where a clique K_{LM} fits into $C_{M,M,L}$ (Choi, 2011). In fact, it is possible to embed a clique of size $LM + 1$, but no clique larger than $L\,(M + 1)$, in $C_{M,M,L}$ (Klymko et al., 2014). Later work by Boothby et al. (2016) generalized the form by which such clique embeddings can be obtained and consequently showed that there is an exponential number of such embeddings in the Chimera graph, which can be helpful if some qubits are inoperable and thus some vertices of the hardware graph H are missing.

Figure 2 illustrates how K_{32} can be embedded in $C_{8,8,4}$ by dividing 32 vertices into groups of 4 vertices, which are indexed from 1 to 8. The first group of 4 vertices is associated with all left partitions of the first column of unit cells and also the right partition of the bottom unit cell. The second group of 4 vertices is associated with all left partitions of the second column, except the last one, and also with the right partitions of the occupied cells in the second row from the bottom. Similar L-shaped

chains follow for the remaining 6 groups. The vertices in distinct groups are adjacent to one another through the cells in the upper triangle of the grid, and the vertices within each group are adjacent to one another through the cells in the main diagonal of the grid. If we associate the remaining cells with a single additional vertex, then we can embed K_{33} instead. We know from Klymko et al. (2014) that we cannot embed K_{37}, but it is not known if cliques K_{34}, K_{35}, or K_{36} could be embedded in $C_{8,8,4}$.

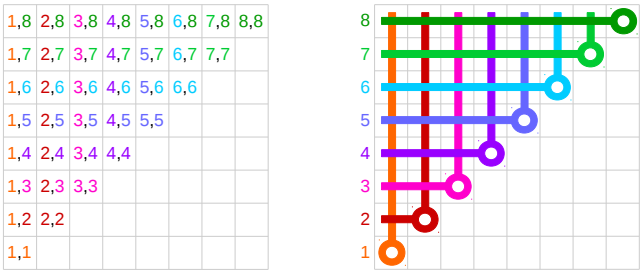

Figure 2: An embedding of K_{32} in $C_{8,8,4}$. Each group numbered from 1 to 8 consists of four connected vertices. On the left, those numbers are associated with left and right partitions of cells in the grid. On the right, vertical and horizontal lines correspond to each group occupying left and right partitions, and circles for both, following the notation in Boothby et al. (2016).

Recent work by Date et al. (2019) has focused on limiting the number of qubits used when embedding graphs with at most ML vertices. By reducing the number of qubits associated with each variable, their approach is able to obtain QUBO solutions that are closer to the optimal value. Another recent line of inquiry concerns embedding the product of graphs, which naturally arise as the problem graph of some formulations (Zaribafiyan et al., 2017). There are also other general-purpose approaches that break a QUBO problem into smaller parts, for example by decomposition (Bian et al., 2016) or fixing some variables to their likely value in optimal solutions (Karimi and Rosenberg, 2017).

The line of work that we will explore in this paper consists of embedding problem graphs with more than ML vertices without decompostion, in particular for the case of dense problem graphs. In sparse problem graphs, heuristics have been quite successful (Cai et al., 2014; Yang and Dinneen, 2016). Among those, one of the most widely used is the CMR algorithm (Cai et al., 2014). In dense problem graphs, the state-of-the-art consists of using a virtual hardware as an intermediary for the embedding. The virtual hardware consists of a particular minor of the Chimera graph $C_{M,M,L}$, which is

chosen to preserve the ability to embed large and dense graphs while making it easy to describe the family of minors that can be obtained from it. This idea was pioneered by Goodrich et al. (2018b) with a complete bipartite graph $K_{ML,ML}$ as virtual hardware, and is currently the state-of-the-art for embedding general problem graphs in QA hardware. Figure 3 illustrates how $K_{64,64}$ can be embedded in $C_{16,16,4}$: each group of 4 vertices is associated with all right partitions of a given row or with all left partitions of a given column.

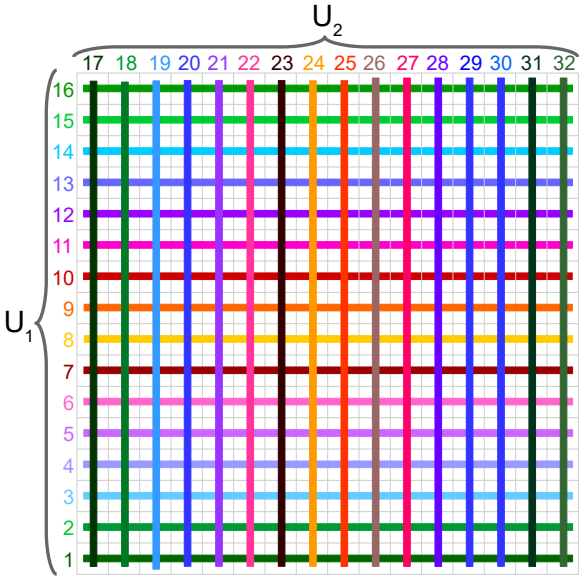

Figure 3: An embedding of $K_{64,64}$ in $C_{16,16,4}$ using groups of 4 vertices, the first 16 are associated with the right partitions of cells in each row (set U_1) and the last 16 with the left partitions of cells in each column (set U_2).

Any minor of $K_{ML,ML}$ is isomorphic to a subgraph of one among ML minors of $K_{ML,ML}$ (Hamilton and Humble, 2017). In essence, each vertex $v_i \in V(G)$ is assigned to vertices in either one or both partitions of $H = K_{ML,ML}$ to obtain an embedding of G, and thus edge $\{u_j, u_k\} \in F(H)$ is contracted if vertex v_i is assigned to both u_j and u_k.

The premise in Goodrich et al. (2018b) is to assign vertices of an Odd Cycle Transversal (OCT) of the problem graph to vertices in both partitions of $K_{ML,ML}$. An OCT of a graph G is a set T such that every odd cycle in G has at least one vertex in T, hence implying that the removal of T results in a bipartite graph, and consequently the remaining vertices are each assigned

to a single vertex of $K_{ML,ML}$. Those authors observed that an OCT of G having minimum size implies the minimum size of a complete bipartite graph in which G can be embedded. If T is such a minimum size OCT and V_1 and V_2 are the resulting partitions of the subset of vertices defined by $V(G) \setminus T$, then it follows that G can only be embedded in complete bipartite graphs having at least $2|T| + |V_1| + |V_2|$ vertices.

Later work in Goodrich et al. (2018a) used ILP formulations to find an OCT of minimum size, hence minimizing $|T|$ as an approach to determine if a given problem graph is embeddable in $K_{ML,ML}$. They report that customized algorithms can find an OCT of smaller graphs faster. However, they also acknowledge that a general-purpose solver is more effective in cases where the problem graphs are harder to embed. We show in this paper that the OCT that embeds G into a complete bipartite graph is not necessarily of minimum size. This serves as a motivation for considering a different optimization approach.

2.3 Related Work

Some authors have been exploring how to solve a broader class of optimization problems with QA hardware. Recent work by Dridi et al. (2018) uses Groebner bases to represent optimization problems involving polynomial functions of higher order on binary domains as QUBO problems, which can then potentially be solved by existing QA hardware. Subsequent work by Alghassi et al. (2019) uses Graver bases to achieve the same with integer non-linear optimization problems, which may also include constraints. A number of other quadratizations that can be applied to such problems is summarized by Dattani (2019).

One can also solve a QUBO using integer linear programming, since the quadratic terms on binary variables can be linearized with an extended formulation (Padberg, 1989). In recent work, Coffrin et al. (2019) uses ILP to verify the solutions generated by QA hardware.

3 Contributions of This Paper

We show how Integer Linear Programming (ILP) can be used as an effective preprocessing step in AQC, especially for problem graphs with more vertices than the largest embeddable cliques. Note that we are not interested in solving minor embedding problems that could be nearly as difficult as the corresponding QUBO problem. We focus instead on how classical optimization algorithms could leverage the potential of quantum optimization

algorithms. Hence, we strive for a balance between computational speed and the ability to embed larger problem graphs by defining simple formulations that exploit the structure of Chimera graphs. In each of these formulations, we cluster the vertices of a minor of the Chimera graph in some partitions and formulate a problem of deciding how to assign vertices of the problem graph to one or more of such partitions. In summary, our main contributions are:

(i) We propose Template Embeddings (TEs) as a generalization of the virtual hardware concept. Each template embedding is a minor of the Chimera graph that can embed a variety of problem graphs with few edge contractions. We study two classes of those: the Bipartite TE (BTE), as the virtual hardware in Goodrich et al. (2018b); as well as the Quadripartite TE (QTE), as a generalization of the former.

(ii) We show that every embedding in BTE is associated with an OCT, but OCTs of minimum size do not certify that a given problem graph cannot be embedded in BTE.

(iii) We present ILP formulations to determine how to embed a problem graph on the minor of each template embedding with competitive results. For BTE, the formulation provides a certificate of embeddability or lack thereof.

4 Bipartite Template Embedding

For a Chimera graph $C_{M,M,L}$, BTE consists of the minor $K_{ML,ML}$ used as virtual hardware by Goodrich et al. (2018b), in which the vertices of the hardware graph are partitioned into sets U_1 and U_2 of size ML each. The construction of BTE is described in Figure 3.

In order to embed a problem graph G in BTE, we need to determine which vertices of $V(G)$ should be assigned to partitions U_1 and U_2. A vertex assigned to a single partition should only be adjacent to vertices assigned to the other partition. If assigning all vertices is proven impossible, then G cannot be embedded in BTE. If all vertices are assigned to at least one partition, then the solution defines a valid embedding.

Before formulating the embeddability of a problem graph G in BTE, we discuss how BTE embeddings relate to OCTs and OCTs of minimum size.

4.1 OCTs and Bipartite Embedding

In this section, we characterize the relationship between OCTs of a graph G and the embedding of G in a complete bipartite graph K_{m_1,m_2}. More specifically, we show that:

(i) The set of vertices $S \subseteq V(G)$ that are assigned to both partitions of K_{m_1,m_2} in any embedding of G is a superset of some OCT $T \subset V(G)$, but T may be a proper subset of S in each of the possible embeddings.

(ii) The largest OCT T contained in S may not be an OCT of minimum size in any of the possible embeddings.

Those results are shown in the following propositions.

Proposition 1. *For any embedding of a graph G in K_{m_1,m_2}, the set of vertices $S \subseteq V(G)$ assigned to both partitions of K_{m_1,m_2} is such that there is an OCT T of G for which $T \subseteq S$. In some cases, $T \subset S$ in every possible embedding.*

Proof. First we show that the set of vertices S assigned to both partitions in any embedding is a superset of an OCT T. Let us suppose, for contradiction, that none of the vertices incident to an odd cycle of G, say $v_1 v_2 \dots v_k v_1$ for $k \geq 3$ and odd, are in S. Since each of those vertices is assigned to one partition of K_{m_1,m_2}, whereas consecutive vertices should necessarily be in different partitions, then one partition is assigned to each v_i with even i and the other partition is assigned to each v_i with odd i. However, since v_k and v_1 are adjacent and assigned to the same partition, then we do not have a valid embedding, and we reach a contradiction. Hence, S contains at least one vertex of every odd cycle in G and is indeed an OCT. This proves the first claim.

Next we show that the embedding of certain graphs implies that we assign to both partitions a vertex that is not incident to any odd cycle. In particular, let us consider a *star graph* $K_{1,m}$, where a single vertex v is adjacent to all the other vertices and consequently the only OCT is an empty set. If $m > \max\{m_1, m_2\}$ but $m \leq m_1 + m_2 - 2$, then we can only embed $K_{1,m}$ in K_{m_1,m_2} if vertex v is assigned to both partitions, in which case each of the remaining m vertices can then be assigned to either partition of K_{m_1,m_2}. □

In other words, there is always an OCT in the set of vertices S that should be assigned to both partitions of K_{m_1,m_2}. However, that set may

also contain other vertices. Proposition 1 shows that some of those vertices may not be incident to any OCT. In what follows, we illustrate that the remaining vertices may define an OCT that is not of minimum size.

Consider the graph G in Figure 4, where vertices v_1 and v_2 are adjacent and each is also adjacent to all vertices in set $V_A := \{v_4, v_5, v_6, v_7\}$, whereas v_3 is adjacent to all vertices in sets V_A and $V_B := \{v_8, v_9, v_{10}, v_{11}\}$. Vertices v_1 and v_2 are incident to all odd cycles of G, such as $v_1v_4v_2v_1$, and consequently we only need to remove one of them to obtain a graph with no odd cycles. In addition, v_3 is incident to some odd cycles, such as $v_1v_4v_3v_5v_2v_1$. Hence, $\{v_1\}$ and $\{v_2\}$ are OCTs of minimum size, but $\{v_1, v_3\}$ is also an OCT.

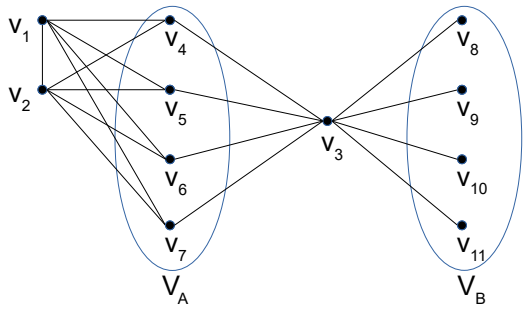

Figure 4: Example of a graph G for which assigning vertices in an OCT of minimum size to both partitions of a bipartite graph H may not suffice to obtain a valid embedding of G in H.

Let us first consider what happens if we only assign an OCT of minimum size to both partitions of K_{m_1,m_2}, which would imply $S = T = \{v_1\}$ or $S = T = \{v_2\}$. Since $G - v_1$ and $G - v_2$ are isomorphic, we only consider assigning v_1 to both partitions and the vertices in $G - v_1$ to a single partition each. In that case, vertex v_2 and the vertices in set V_A are necessarily in different partitions. Since vertex v_3 is adjacent to the vertices in V_A, it follows that v_3 is in the same partition as v_2. Since the vertices in V_B are all adjacent to v_3, then the vertices in V_B are in the same partition as those in V_A. Consequently, we need to assign $\{v_2, v_3\}$ to one partition and $V_A \cup V_B$ to the other partition. Therefore, we can only embed graph G in K_{m_1,m_2} by assigning the vertices in a minimum size OCT to both partitions and the rest to a single partition each if $m_1 \geq 3$ and $m_2 \geq |V_A| + |V_B| + 1 = 9$ or vice-versa. This is illustrated on the left of Figure 5.

Now let us consider what happens if we assign a larger OCT to both partitions of K_{m_1,m_2}, say $\{v_1, v_3\}$. In the graph $G \setminus \{v_1, v_3\}$, vertex v_2

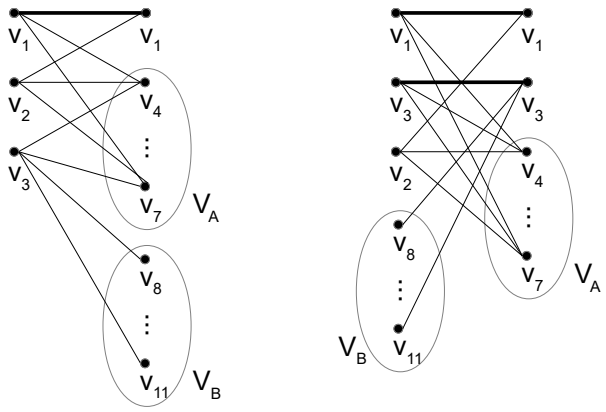

Figure 5: Embeddings of graph G from Figure 4 in a bipartite graph. On the left, we assign only the vertices of a minimum size OCT $\{v_1\}$ to both partitions and consequently minimize the number of vertices used in both partitions to 12. On the right, we assign the vertices of a larger OCT $\{v_1, v_3\}$ to both partitions and consequently reduce the largest number of vertices assigned to either partition from 9 to 7.

and the vertices in V_A are again in different partitions. Since the vertices in V_B are not adjacent to the other remaining vertices, then one possible partitioning of the remaining vertices is $\{v_2\} \cup V_B$ and V_A, hence implying that we can embed G in K_{m_1,m_2} if $m_1 \geq |V_B| + 3 = 7$ and $m_2 \geq |V_A| + 2 = 6$ or vice-versa. This is illustrated on the right of Figure 5.

If $m_1 = m_2 = 8$, then assigning only the vertices of a minimum size OCT $\{v_1\}$ to both partitions and assigning $V_A \cup V_B$ to a single partition is not feasible since no partition has 9 vertices. However, it is possible to assign only the vertices of the OCT $\{v_1, v_3\}$ to both partitions and obtain an embedding, since in such case we only need to assign 7 vertices to one partition of K_{m_1,m_2} and 6 vertices to the other partition.

The rationale for finding an OCT T of minimum size is that it allows G to be embedded in the smallest complete bipartite graph, regardless of the size of each partition in which we are embedding. However, the size of the partitions matter. If we denote the partitions of $G \setminus T$ as the sets of vertices V_1 and V_2, then G can only be embedded with such partitions in K_{m_1,m_2} if $\min(m_1, m_2) \geq |T|$ and $\max(m_1, m_2) \geq |T| + |V_1| + |V_2|$.

The graph in Figure 4 can be embedded with an OCT of minimum size in $K_{3,9}$, which has only 12 vertices. It can also be embedded with a larger OCT in $K_{6,7}$, which has 13 vertices. If we were to consider a Chimera graph

$C_{16,16,4}$ and tried to embed a similar graph in which $|V_A| = |V_B| = 32$, only the second embedding above would be possible in BTE.

4.2 Exact Bipartite Embedding

The formulation below determines if a problem graph G is embeddable in BTE.

Decision Variables For each vertex $v_i \in V(G)$ and $k \in \{1, 2\}$, let $y_{i,k} \in \{0, 1\}$ be a binary variable for whether vertex v_i is assigned to partition U_k and let $y'_i \in \{0, 1\}$ be a binary variable denoting whether v_i is assigned to any partition.

Objective Function The following expression aims at assign as many vertices as possible:

$$\max \sum_{i=1}^{n} y'_i.$$

If the ILP solver reports an upper bound lower than n, then it is not possible to embed G in BTE.

Constraints For each vertex $v_i \in V(G)$, we associate both types of decision variables as follows:

$$y'_i \leq y_{i,1} + y_{i,2}. \tag{1}$$

For each partition U_k, $k \in \{1, 2\}$, no more than ML vertices of $V(G)$ should be assigned to it:

$$\sum_{i=1}^{n} y_{i,k} \leq ML. \tag{2}$$

For each edge $\{v_i, v_j\} \in E(G)$, $i < j$, vertices v_i and v_j should not be assigned to a single and same partition:

$$y_{i,1} + y_{j,1} - y_{i,2} - y_{j,2} \leq 1 \tag{3}$$

$$y_{i,2} + y_{j,2} - y_{i,1} - y_{j,1} \leq 1. \tag{4}$$

The constraints above are the canonical cuts on the unit hypercube defined by the binary variables (Balas and Jeroslow, 1972). Each corresponds to the tightest single inequality on such space because it separates a single

combination of values for those variables and is tight for each combination of values that differs in only one variable. For example, the first inequality above separates $(y_{i,1}, y_{i,2}, y_{j,1}, y_{j,2}) = (1, 0, 1, 0)$ from the feasible set while holding at equality for the adjacent assignments $(y_{i,1}, y_{i,2}, y_{j,1}, y_{j,2}) = (0, 0, 1, 0), (1, 1, 1, 0), (1, 0, 0, 0)$, and $(1, 0, 1, 1)$.

Proposition 2 (Certificate of Embeddability)**.** *Graph G is embeddable in BTE if and only if there is a solution to the ILP formulation with objective value $|V(G)|$.*

Proof. We start with the premise of assigning vertices to partitions. In order to embed G in BTE, every vertex v of G should be assigned to one or more vertices of BTE. If v is assigned to vertices u_1 and u_2 in BTE and those vertices are in the same partition, then u_1 and u_2 have the same neighbors and we can obtain an equivalent embedding by assigning v to only one of them. Therefore, every embedding of G in BTE can be reduced to an embedding in which each vertex of G is assigned to at most one vertex of each partition and the premise of assigning vertices of G to partitions of BTE is valid.

Next we show that any graph G that can be embedded in BTE defines a solution to the ILP formulation. Based on the discussion above, for any such graph we can define a mapping of every one of its vertices to partitions U_1 and U_2 of BTE that corresponds to an embedding of G in BTE, which implies that $y_{i,k} = 1$ if $v_i \in V(G)$ is assigned to partition U_k, $k \in \{1, 2\}$, and $y_{i,k} = 0$ otherwise. By definition, we can infer that three constraints of the ILP formulation are always satisfied with such a mapping. First, no more than ML vertices of G are assigned to either U_1 or U_2 in any embedding in BTE because that is the number of vertices in each of those partitions, and thus any such embedding satisfies constraint (2). Second, no pair of adjacent vertices in G are assigned only to the same partition in BTE because the corresponding vertices in BTE would not be adjacent, and thus any embedding in BTE would not assign any such pair only to partition U_1 as prevented by constraint (3) or to partition U_2 as prevented by constraint (4). In addition, since every vertex $v_i \in V(G)$ is assigned to at least one partition, then it follows that $y_{i,1}+y_{i,2} \geq 1$ and thus there is a feasible solution in which $y'_i = 1\ \forall v_i \in V(G)$ and the objective function value of $|V(G)|$ is attainable because constraint (1) is not violated by such assignments. Therefore, there is a solution of value $|V(G)|$ for every embedding of G in BTE.

Finally, we show that any graph G for which we cannot find such a solution to the ILP formulation cannot be embedded in BTE. Since a solution with all variables equal to zero is feasible, we just need to consider the case

in which the optimal value is less than $|V(G)|$. We will prove this by contradiction. Suppose, for contradiction, that there is an optimal solution with value $|V(G)|$ for a graph G that cannot be embedded in BTE. That would imply that each vertex of G is assigned to at least one partition due to constraint (1) and optimality. Further, no more than ML vertices are assigned to each partition due to satisfaction of constraint (2), and adjacent vertices are not assigned solely to vertices in U_1 due to satisfaction of constraint (3) or to vertices in U_2 due to satisfaction of constraint (4). Consequently, such a solution would contradictorily define a valid embedding of G. Therefore, the value of any solution to the ILP formulation associated with a graph G that is not embeddable in BTE is less than $|V(G)|$. □

An important implication of Proposition 2 is that we can terminate the search process and conclude that G is not embeddable in BTE once the search determines that there is no solution with objective value $|V(G)|$. Search algorithms incorporate a number of domain reduction techniques, which lead to considerable pruning of the search space, and this can be favorably exploited to produce a certificate of non-embeddability. Furthermore, any solution with objective value $|V(G)|$ is an optimal solution, and the search can be terminated if one of those is found.

5 Quadripartite Template Embedding

We define the Quadripartite Template Embedding (QTE) as a minor of the Chimera graph with vertices partitioned into sets U_1, U_2, U_3, and U_4. Each vertex in U_1 is adjacent to all vertices in U_2, those in U_2 are also each adjacent to a distinct vertex in U_3, and those in U_3 are also adjacent to all vertices in U_4. In other words, the subgraph induced on $U_1 \cup U_2$ and $U_3 \cup U_4$ are both complete bipartite graphs, and the subgraph induced in $U_2 \cup U_3$ is a perfect matching. In fact, embedding on QTE generalizes embedding on BTE, since BTE is a minor of QTE after contracting all edges between sets U_2 and U_3: $U_1 \cup U_4$ defines one partition of BTE in that case and the other partition is defined by the vertices resulting from contracting the edges between U_2 and U_3. For a Chimera graph $C_{M,M,L}$ with M even and thus $P := M/2$ integer, the size of those partitions are: $|U_1| = PL$, $|U_2| = ML$, $|U_3| = ML$, and $|U_4| = PL$. Figure 6 illustrates that minor of $C_{16,16,4}$.

The set of minors of QTE is a superset of the minors of BTE. QTE's minors include some larger graphs with fewer adjacencies that would not be possible to embed in BTE. In fact, BTE is a minor of QTE in which the edges between vertices in partitions U_2 and U_3 are contracted. QTE offers

the flexibility that two adjacent vertices in U_2 and U_3 can be mapped to distinct vertices in G.

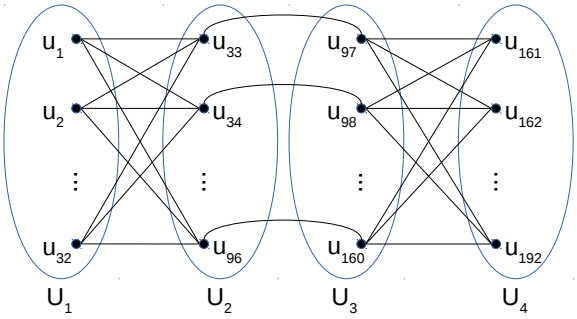

Figure 6: QTE minor of Chimera graph $C_{16,16,4}$.

QTE can be obtained from $C_{M,M,L}$ as follows. Set U_1 consists of P groups of L vertices, each group obtained by contracting the L vertices of the right partitions of one of the top P rows of the $M \times M$ grid. Set U_2 consists of M groups of L vertices, each group obtained by contracting the L vertices of the left partitions in the top P rows of one of the M columns of the grid. Set U_3 consists of M groups of L vertices, each group obtained by contracting the L vertices of the left partitions in the bottom P rows of one of the M columns of the grid. Set U_4 consists of P groups of L vertices, each group obtained by contracting the L vertices of the right partitions of one of the bottom P rows of the grid. Figure 7 illustrates QTE in $C_{16,16,4}$.

In order to embed a problem graph in QTE, each vertex should be assigned to a sequence of adjacent partitions and each pair of adjacent vertices v_i and v_j should be assigned to vertices u_i and u_j that are adjacent in QTE. For simplicity, we only consider that v_i and v_j have been assigned to vertices u_i and u_j that are adjacent in QTE if those vertices are in distinct partitions that together induce a complete bipartite graph, i.e., U_1 and U_2 or U_3 and U_4. Hence, we ignore the possibility of assuming u_i and u_j adjacent if one of these vertices is in partition U_2 and the other vertex is in partition U_3. Otherwise, we would need to explicitly assign the vertices of the problem graph to specific vertices in those partitions instead of merely deciding that the vertices of the problem graph are assigned to some vertex in the partition, which would make the formulation considerably more complex. More specifically, we could potentially decide the particular vertex of the harware graph H that is associated to each vertex assigned to sets U_2 and U_3 in order to leverage the fact that each vertex of U_2 is adjacent to a vertex of

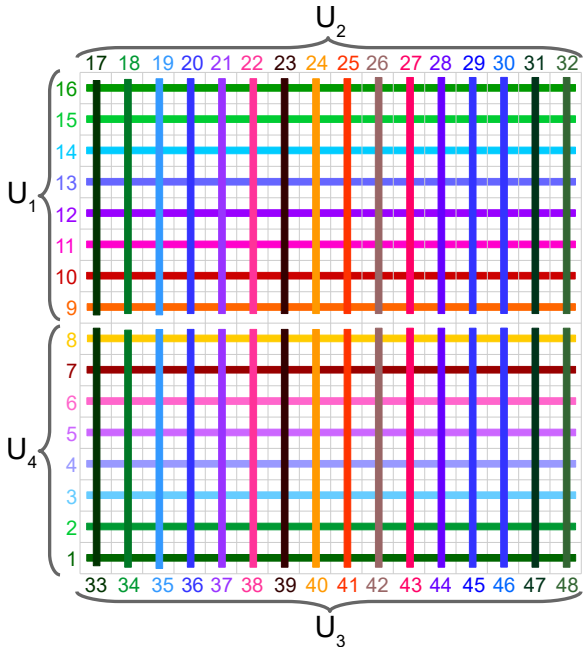

Figure 7: QTE in $C_{16,16,4}$, defining partition U_1 with the top 8 horizontal groups, U_2 with the top 16 vertical groups, U_3 with the bottom 16 vertical groups, and U_4 with the bottom 8 horizontal groups. Each group has 4 vertices.

U_3. However, the number of decision variables would be substantially larger in that case. Given our aim for simplicity, the formulation below does not provide a certificate of embeddability in QTE because we only exploit adjacencies between U_2 and U_3 if a same vertex is assigned to both.

Decision Variables For each vertex $v_i \in V(G)$ and $k \in \{1, 2, 3, 4\}$, let $y_{i,k} \in \{0, 1\}$ be a binary variable for whether vertex v_i is assigned to partition U_k, and let $y'_i \in \{0, 1\}$ be a binary variable denoting whether v_i is assigned to any partition. For each edge $\{v_i, v_j\} \in E(G)$, assuming $i < j$, let $z^k_{i,j} \in \{0, 1\}$ be an auxiliary binary variable implying that the adjacency between vertices v_i and v_j is ensured by assigning vertex v_i to partition U_k and vertex j to the partition in which all vertices are adjacent to those in U_k.

Objective Function We maximize vertices assigned:

$$\max \sum_{i=1}^{n} y'_i.$$

Constraints The first constraint associates the first two types of variables for each vertex $v_i \in V(G)$, as in BTE:

$$y'_i \leq \sum_{k=1}^{4} y_{i,k}.$$

For each partition U_k, $k \in \{1, 2, 3, 4\}$, the number of vertices assigned to U_k is bounded by the size of that partition:

$$\sum_{i=1}^{n} y_{i,k} \leq |U_k|.$$

For each vertex $v_i \in V(G)$, we want the set of partitions to which v_i is assigned to be pairwise contiguous. We formulate that with constraints preventing each possible discontinuity: (i) assigning v_i to U_1 and U_3 implies that v_i is also assigned to U_2; (ii) assigning v_i to U_1 and U_4 implies that v_i is also assigned to U_2 and U_3; and (iii) assigning v_i to U_2 and U_4 implies that v_i is also assigned to U_3. Hence, we use canonical cuts on the unit hypercube in the corresponding subspaces to exclude the assignments $(y_{i,1}, y_{i,2}, y_{i,3}) = (1, 0, 1)$ for (i); $(y_{i,1}, y_{i,2}, y_{i,4)}) = (1, 0, 1)$ and $(y_{i,1}, y_{i,3}, y_{i,4)}) = (1, 0, 1)$ for (ii); and $(y_{i,2}, y_{i,3}, y_{i,4}) = (1, 0, 1)$ for (iii):

$$y_{i,1} + y_{i,3} - y_{i,2} \leq 1$$

$$y_{i,1} + y_{i,4} - y_{i,2} \leq 1$$

$$y_{i,1} + y_{i,4} - y_{i,3} \leq 1$$

$$y_{i,2} + y_{i,4} - y_{i,3} \leq 1.$$

For (ii), it would suffice to exclude $(y_{i,1}, y_{i,2}, y_{i,3}, y_{i,4}) = (1, 0, 0, 1)$ with $y_{i,1} + y_{i,4} - y_{i,2} - y_{i,3} \leq 1$ because the other cases are covered. However, by summing the two inequalities used for (ii) we obtain the implied inequality $2y_{i,1} + 2y_{i,4} - y_{i,2} - y_{i,3} \leq 2$, which is stronger than $y_{i,1} + y_{i,4} - y_{i,2} - y_{i,3} \leq 1$ on continuous domains in $[0, 1]$ and excludes additional fractional values such as $(y_{i,1}, y_{i,2}, y_{i,3}, y_{i,4}) = (1, 0.5, 0.5, 1)$. Note that a formulation that has a smaller feasible set when the binary variables on $\{0, 1\}$ are relaxed to

continuous variables on $[0, 1]$ is considered as stronger and often is solved faster.

For each edge $\{v_i, v_j\} \in E(G)$, $i < j$, we want vertex v_i assigned to at least one partition U_k such that vertex v_j is assigned to the corresponding partition U_l where the set of vertices $U_k \cup U_l$ induces a complete bipartite graph. In other words, we want v_i and v_j respectively assigned to either (i) U_1 and U_2; (ii) U_2 and U_1; (iii) U_3 and U_4; or (iv) U_4 and U_3:

$$\begin{aligned} y_{i,1} &\geq z^1_{i,j}, & y_{j,2} &\geq z^1_{i,j} \\ y_{i,2} &\geq z^2_{i,j}, & y_{j,1} &\geq z^2_{i,j} \\ y_{i,3} &\geq z^3_{i,j}, & y_{j,4} &\geq z^3_{i,j} \\ y_{i,4} &\geq z^4_{i,j}, & y_{j,3} &\geq z^4_{i,j} \end{aligned}$$

$$\sum_{k=1}^{4} z^k_{i,j} \geq 1.$$

Note that the first two inequalities above imply $y_{i,1} + y_{j,2} \geq 2z^1_{i,j}$, which alone would also be a valid formulation for (i), and the same follows for conditions (ii), (iii), and (iv). However, having a pair of inequalities instead makes the formulation stronger since it excludes fractional solutions such as $(y_{i,1}, y_{j,2}, z^1_{i,j}) = (0.5, 0.5, 1)$. Those constraints also imply that each vertex incident to at least one edge should be assigned to a partition, and thus an non-embeddable problem graph may lead to an infeasible solution.

6 Experiments

We implemented the ILP formulations for the template embeddings in $C_{16,16,4}$ and $C_{20,20,4}$ using Gurobi 9.0.0. We compare our results with those obtained with Fast-OCT-Reduce (FOR) using the source code from Goodrich et al. (2018b), which is the state-of-the-art for embedding general graphs in QA hardware. In other words, we test on the hardware graph currently commercialized by D-Wave Systems ($C_{16,16,4}$) and a larger hardware graph ($C_{20,20,4}$) to compare the scalability of both approaches. We use five random generators of graphs with a density parameter $p = .25$ for Low density, $p = .5$ for Medium density, and $p = .75$ for High density. Four generators are from Goodrich et al. (2018b): Barabási-Albert, Erdős-Rényi, Regular, and Noisy Bipartite. We implement Percolation based on long-range percolation graphs (Coppersmith et al., 2002). For each vertex v_i, we

draw a random number $\chi_i \in [0, 1)$ and we include edge $\{v_i, v_j\}$ with probability $\min\left\{1, \frac{p}{|\chi_i - \chi_j|}\right\}$. The problem graphs used in the experiments and a summary of the results can be downloaded from `ftp://ftp.merl.com/pub/raghunathan/TemplateEmbedding-TestSet/`. The source code can be downloaded from `https://www.merl.com/research/license/TEAQC`.

Since TRIAD can easily generate a clique of size 64 for $C_{16,16,4}$, we generate 5 random graphs from each generator and with each density for number of vertices ranging from 65 to 128. Likewise, TRIAD can easily generate a clique of size 80 for $C_{20,20,4}$, and for that case the number of vertices ranges from 81 to 160 in the experiments. The maximum sizes for the tested graphs come from assuming that each vertex would be assigned to a single partition. In the case of low and medium density, the expected number of edges of the random graphs is smaller than the number of edges of the template embeddings. In the case of high density, the expected number of edges would exceed $\binom{ML}{2}$ if the number of vertices is sufficiently large. Hence, we have limited the experiments with random graphs having high density to at most 105 vertices in the case of $C_{16,16,4}$ and at most 131 vertices in the case of $C_{20,20,4}$. In total, we tried to embed 4,225 graphs in $C_{16,16,4}$ and 5,275 graphs in $C_{20,20,4}$. The time required to embed problems is currently the bottleneck in solving problems on QA hardware. With an eye towards cases that would benefit from our classical approach as a presolve for AQC, we set a time limit of 60 seconds as a satisficing threshold to identify what graphs can be embedded by each approach. We also complement our analysis with plots that compare runtimes in order to identify the most effective approach. All experiments were conducted on a single thread in an Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz and 128 GB RAM.

We begin by comparing the total number of graph instances that can be embedded by the different algorithms in Figure 8. For $C_{16,16,4}$, Fast-OCT-Reduce (FOR) can embed 694 instances while BTE and QTE can embed 717 and 670, respectively. Thus, BTE improves on FOR marginally while QTE embeds fewer instances than FOR. For $C_{20,20,4}$, BTE can embed significantly more instances as compared to FOR (798 vs. 575, respectively). QTE also does much better than FOR in this case (575 vs. 671).

Our next analysis in Figure 9 aims to identify the unique strengths of each algorithm beyond the total number of instances that are successfully embedded. We count the instances that are embedded by only one approach or template embedding, i.e. that no other approach was able to embed. From this analysis, we see that every graph that is embedded by FOR is also successfully embedded by either BTE or QTE. Among the template

embeddings, BTE embeds more instances than QTE.

Tables 1 and 2 show the largest graph that each algorithm could respectively embed in $C_{16,16,4}$ and $C_{20,20,4}$ within the time limit for each type of random generator and density, with the largest numbers of each row in bold. This helps to identify the type of random graph generators and densities for which one approach is superior to the the others.

Table 3 provides a breakdown of the number of graphs embedded in $C_{16,16,4}$ and $C_{20,20,4}$ by the graph classes considered. This is a dissection of the results in Figure 8, which also complements the results reported in Tables 1 and 2 by quantifying the total number of graphs embedded.

Figures 10 and 11 respectively show the performance profiles of each approach as well as the combination of all template embeddings in comparison with the OCT-based approach in $C_{16,16,4}$. In other words, we see the cumulative number of graphs embedded over time for each case. Figures 12 and 13 show the same type of performance profiles with respect to embedding into $C_{20,20,4}$. The performance profiles evidence that the use of template embeddings results in runtimes that are orders of magnitude faster than FOR.

In some of these figures and tables, we respectively denote $C_{16,16,4}$ and $C_{20,20,4}$ as C16 and C20 for brevity.

One noticeable difference between the template embeddings is whether we can determine if the embedding problem is feasible or not by the time limit. In the case of BTE, we were able to find a feasible embedding or determine that none exists by the time limit. In the case of QTE, it was not possible to determine if an embedding was infeasible by the time limit. That speaks to the importance of having models that are as simple as possible, and also highlights the fact that an ILP formulation can reliably determine if a graph can be embedded in a complete bipartite graph for the sizes that we tested.

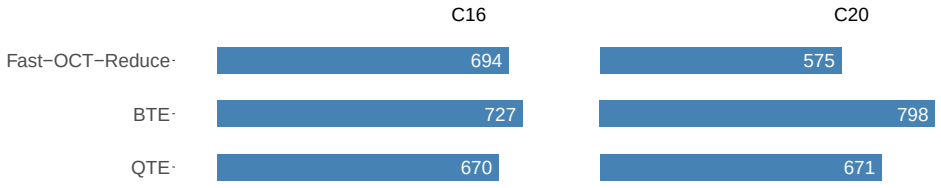

Figure 8: Number of graphs embedded by Fast-OCT-Reduce and each template embedding in C16 and C20.

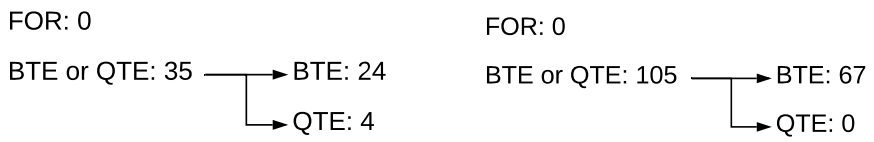

Figure 9: Graphs uniquely embedded by each approach or a specific template embedding in C16 (left) and C20 (right).

Table 1: The largest graph embedded by each algorithm per generator and density in C16.

Random Graph	Density	FOR	BTE	QTE
Percolation	low	67	**68**	**68**
	medium	**66**	**66**	**66**
	high	**66**	**66**	**66**
Barabási-Albert	low	77	**78**	76
	medium	**72**	**72**	71
	high	**69**	**69**	**69**
Erdős-Rényi	low	79	**80**	77
	medium	71	**72**	**72**
	high	**69**	**69**	68
Regular	low	**78**	**78**	77
	medium	**72**	**72**	**72**
	high	**69**	**69**	**69**
Noisy Bipartite	low	**92**	**92**	**92**
	medium	**85**	**85**	82
	high	**81**	**81**	80

Table 2: The largest graph embedded by each algorithm per generator and density in C20.

Random Graph	Density	FOR	BTE	QTE
Percolation	low	81	**84**	83
	medium	**82**	**82**	**82**
	high	**82**	**82**	**82**
Barabási-Albert	low	85	**95**	92
	medium	86	**89**	87
	high	**85**	**85**	84
Erdős-Rényi	low	**97**	**97**	94
	medium	**89**	**89**	88
	high	85	**86**	85
Regular	low	**95**	**95**	94
	medium	**88**	**88**	87
	high	**85**	**85**	84
Noisy Bipartite	low	110	**111**	**111**
	medium	100	**102**	99
	high	99	**100**	96

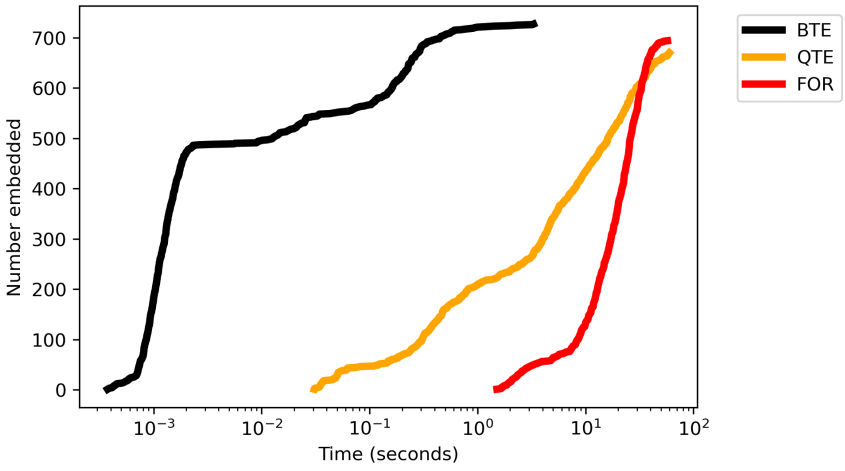

Figure 10: Performance profile of graphs embedded by Fast-OCT-Reduce and each template embedding in C16.

Table 3: Number of graphs embedded by Fast-OCT-Reduce (FOR) and Template Embeddings (TE) in C16 and C20.

	C16		C20	
Random Graph	FOR	TE	FOR	TE
Percolation	35	37	24	38
Barabási-Albert	114	126	67	138
Erdős-Rényi	128	139	90	150
Regular	127	129	112	140
Noisy Bipartite	290	301	282	332
Total	694	732	575	798

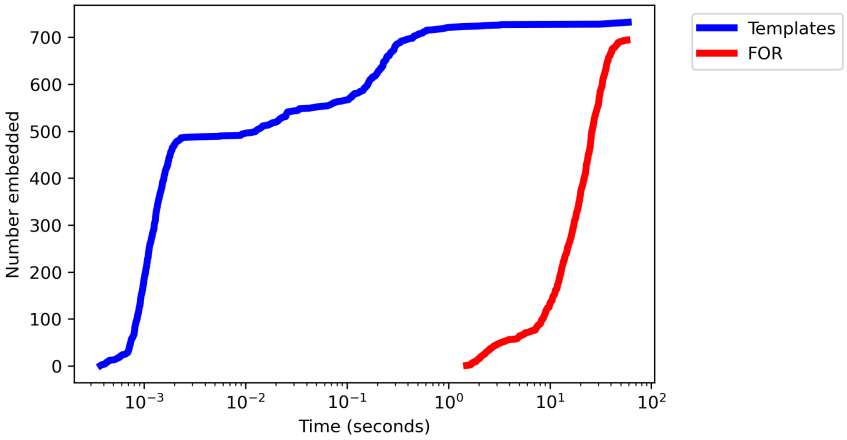

Figure 11: Performance profile of graphs embedded by Fast-OCT-Reduce and any template embedding in C16.

6.1 Analysis

The ILP formulations based on template embeddings can embed 5% and 39% more graphs than the OCT-based approach in $C_{16,16,4}$ and $C_{20,20,4}$, respectively. More importantly, these graphs can be embedded substantially faster. Furthermore, every graph embedded by the OCT-based approach is also embedded by one of the template embeddings. Out of the 30 different combinations of random graph, density, and hardware graph, BTE embeds the largest graph in all 30 of them. Meanwhile, FOR embeds the largest graph in 19, and QTE in 11. The results are particularly favorable with the larger hardware graph, in which we can often embed more graphs

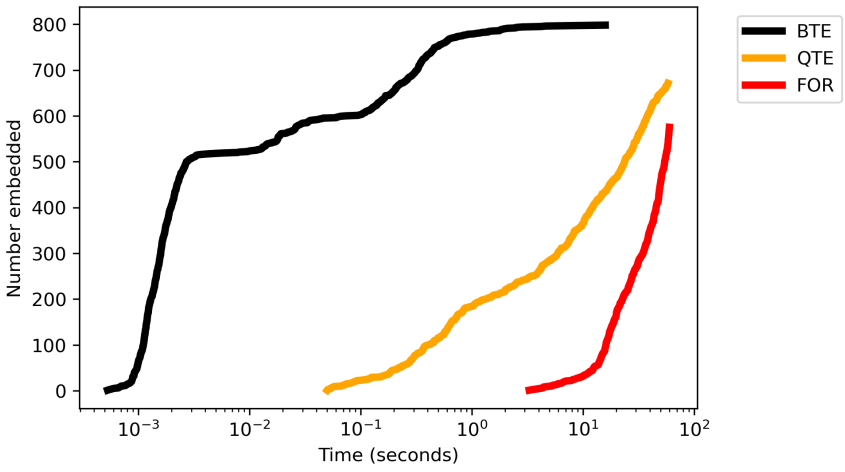

Figure 12: Performance profile of graphs embedded by Fast-OCT-Reduce and each template embedding in C20.

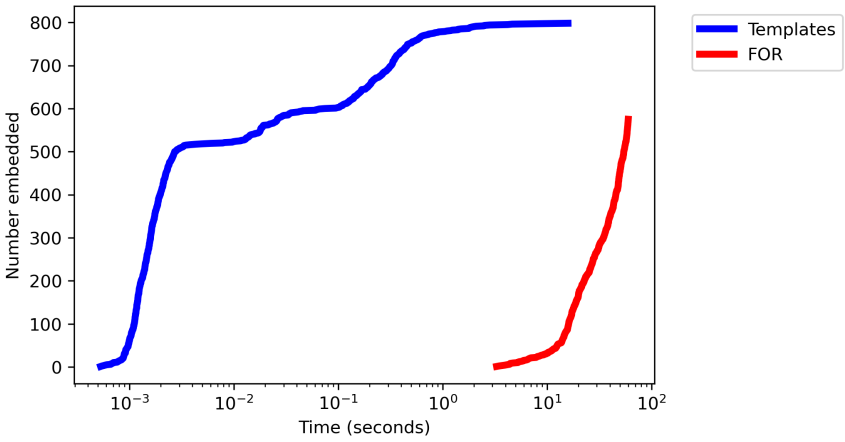

Figure 13: Performance profile of graphs embedded by Fast-OCT-Reduce and any template embedding in C20.

with template embeddings and fewer with the OCT-based approach. That evidences the scalability of the proposed approach. While BTE is a special case of QTE, the simpler formulation allowed BTE to embed more Barabási-Albert, Erdős-Rényi, regular, and noisy bipartite graphs within the time limit in both hardware graphs. QTE performed comparatively better with

Percolation graphs of lower density. While BTE dominates the performance profile curves, we note that QTE helped embedding extra graphs as the runtime increased.

Our main takeaway is that solving a simpler embedding problem yields better results. Because of its relatively small formulation, BTE led to very fast and scalable results. In addition, as conjectured, QTE showed favorable results with lower density graphs.

7 Conclusion and Future Work

We proposed the concept of template embeddings to map quadratic unconstrained binary optimization problems into quantum annealers. Each template embedding corresponds to a minor of the Chimera graph that can embed a variety of large and dense graphs. We also introduced integer linear programming formulations to find such mappings and showed that one of these formulations is exact, and thus certify if a given graph can be embedded in the corresponding minor. Experimental results clearly demonstrate the potential of the proposed approach, especially to embed problems having more variables than the maximum embeddable clique in the Chimera graph corresponding to the QA hardware.

Interestingly, our approach makes a better use of quantum annealers by leveraging classical optimization algorithms as a preprocessing step. The performance of solvers for mixed-integer linear programming has improved by orders of magnitude in the last decades (Bixby, 2012). We believe that there is potential for further coordination between classical and quantum algorithms for discrete optimization.

In future work, we intend to investigate template embeddings that are adaptive to problem sparsity and incorporate knowledge of faulty qubits in the formulations. We should also consider improvements to the formulation of template embeddings and solving strategies to scale up this approach as the size of the hardware graph increases in future releases.

Another line of work consists of extending the template-based approach to minors of Pegasus, the proposed hardware graph for future QA hardware by D-Wave Systems (Dattani et al., 2019). However, we note that the contributions of this work are also valid for Pegasus, as observed in a technical report by D-Wave (Boothby et al., 2019):

> *Much of the embedding support that exists for the Chimera topology may be extended to the Pegasus family of topologies with relative ease. Known constructions for Chimera embeddings of*

structured problems translate to Pegasus without modification, because Chimera occurs as a subgraph of Pegasus.

Acknowledgements The first two authors of the paper, Thiago Serra and Teng Huang, were employed by Mitsubishi Electric Research Laboratories during the initial development of this project. They are the corresponding authors. In addition, the authors would like to thank Claudio Leonardo Lucchesi and Yoshiharu Kohayakawa for their input on random graphs.

References

Hedayat Alghassi, Raouf Dridi, and Sridhar Tayur. Graver bases via quantum annealing with application to non-linear integer programs. *arXiv*, abs/1902.04215, 2019.

E. Balas and R. G. Jeroslow. Canonical cuts on the unit hypercube. *SIAM Journal on Applied Mathematics*, 23:61–69, 1972.

Zhengbing Bian, Fabian Chudak, Robert Brian Israel, Brad Lackey, William G. Macready, and Aidan Roy. Mapping constrained optimization problems to quantum annealing with application to fault diagnosis. *Frontiers in ICT*, 3:14, 2016.

Robert E. Bixby. A brief history of linear and mixed-integer programming computation. In *Documenta Mathematica, Extra Volume: Optimization Stories*, pages 107–121. 2012.

J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer, 2008.

Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. Next-generation topology of d-wave quantum processors. Technical Report 14-1026A-C, D-Wave Systems Inc., 2019.

Tomas Boothby, Andrew D King, and Aidan Roy. Fast clique minor generation in chimera qubit connectivity graphs. *Quantum Information Processing*, 15(1):495–508, 2016.

Endre Boros and Peter L Hammer. Pseudo-boolean optimization. *Discrete applied mathematics*, 123(1-3):155–225, 2002.

J. Cai, W. G. Macready, and A. Roy. A practical heuristic for finding graph minors. *arXiv*, abs/1406.2741, 2014.

Vicky Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, 7(5):193–209, 2008.

Vicky Choi. Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design. *Quantum Information Processing*, 10(3):343–353, 2011.

Carleton Coffrin, Harsha Nagarajan, and Russell Bent. Evaluating ising processing units with integer programming. In Louis-Martin Rousseau and Kostas Stergiou, editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 163–181. Springer International Publishing, 2019.

Don Coppersmith, David Gamarnik, and Maxim Sviridenko. The diameter of a long-range percolation graph. *Random Struct. Algorithms*, 21(1): 1–13, 2002.

D-Wave Systems. Introduction to the D-Wave Quantum Hardware. https://www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware, 2019. Accessed: 2019-05-03.

Sanjeeb Dash. A note on QUBO instances defined on Chimera graphs. *arXiv*, abs/1306.1202, 2013.

Prasanna Date, Robert Patton, Catherine Schuman, and Thomas Potok. Efficiently embedding qubo problems on adiabatic quantum computers. *Quantum Information Processing*, 18(4):117, 2019.

Nike Dattani. Quadratization in discrete optimization and quantum mechanics. *arXiv*, abs/1901.04405, 2019.

Nike Dattani, Szilard Szalay, and Nick Chancellor. Pegasus: The second connectivity graph for large-scale quantum annealing hardware. *arXiv*, abs/1901.07636, 2019.

Raouf Dridi, Hedayat Alghassi, and Sridhar Tayur. A novel algebraic geometry compiling framework for adiabatic quantum computations. *arXiv*, abs/1810.01440, 2018.

Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv*, abs/quant-ph/0001106, 2000.

Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292 (5516):472–475, 2001.

AB Finnila, MA Gomez, C Sebenik, C Stenson, and JD Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chemical physics letters*, 219(5-6):343–348, 1994.

Timothy D Goodrich, Eric Horton, and Blair D Sullivan. Practical graph bipartization with applications in near-term quantum computing. *arXiv*, abs/1805.01041, 2018a.

Timothy D Goodrich, Blair D Sullivan, and Travis S Humble. Optimizing adiabatic quantum program compilation using a graph-theoretic framework. *Quantum Information Processing*, 17(5):118, 2018b.

Kathleen E. Hamilton and Travis S. Humble. Identifying the minor set cover of dense connected bipartite graphs via random matching edge sets. *Quantum Information Processing*, 16(4):1–17, 2017.

David S. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 8(3):438–448, 1987.

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

Bahman Kalantari. Quadratic functions with exponential number of local maxima. *Operations Research Letters*, 5(1):47–49, 1986.

W. M. Kaminsky and S. Lloyd. Scalable architecture for adiabatic quantum computing of NP-hard problems. In A. J. Leggett, B. Ruggiero, and P. Silvestrini, editors, *Quantum Computing and Quantum Bits in Mesoscopic Systems*, pages 229–236. Springer US, Boston, MA, 2004.

W. M. Kaminsky, S. Lloyd, and T. P. Orlando. Scalable superconducting architecture for adiabatic quantum computation. *arXiv*, abs/quant-ph/0403090, 2004.

Hamed Karimi and Gili Rosenberg. Boosting quantum annealer performance via sample persistence. *Quantum Information Processing*, 16(7):166, 2017.

Christine Klymko, Blair D Sullivan, and Travis S Humble. Adiabatic quantum programming: minor embedding with hard faults. *Quantum Information Processing*, 13(3):709–729, 2014.

Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2:5, 2014.

Andrew Lucas. Hard combinatorial problems and minor embeddings on lattice graphs. *arXiv*, abs/1812.01789, 2018.

Catherine C. McGeoch and Cong Wang. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Frontiers*, CF '13, pages 23:1–23:11, 2013.

Manfred Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming*, 45(1):139–172, 1989.

Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. Defining and detecting quantum speedup. *Science*, 345(6195):420–424, 2014.

W. Van Dam, M. Mosca, and U. Vazirani. How powerful is adiabatic quantum computation? In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 279–287, 2001. ISBN 0-7695-1390-5.

Z. Yang and M. J. Dinneen. Graph minor embeddings for d-wave computer architecture. Technical Report CDMTCS-503, Centre for Discrete Mathematics and Theoretical Computer Science, The University of Auckland, 2016.

Arman Zaribafiyan, Dominic J. J. Marchand, and Seyed Saeed Changiz Rezaei. Systematic and deterministic graph minor embedding for cartesian products of graphs. *Quantum Information Processing*, 16(5), 2017.