CDMTCS
Research
Report
Series

Graph Minor Embedding for Adiabatic Quantum Computing

Tony Liu
Zongyou (William) Li
Department of Electrical, Computer and Software Engineering,
University of Auckland,
Auckland, New Zealand

Michael J. Dinneen
Department of Computer Science,
University of Auckland,
Auckland, New Zealand

CDMTCS-549
January 2021

Centre for Discrete Mathematics and
Theoretical Computer Science

ABSTRACT

In recent years, adiabatic quantum computing (AQC) has established itself as a useful computing paradigm which utilizes the underlying physical process of quantum annealing (QA) to help solve NP-hard problems. The main enabler of this field has been the introduction of QA hardware by D-Wave Systems, the latest of which has up to 5640 qubits arranged in the topology of a Pegasus graph. Solving NP-hard problems using QA hardware typically involves a compilation (embedding) step which is equivalent to the graph minor embedding problem; itself an NP-hard problem. This has spurred the development of heuristic algorithms which can find valid minor embeddings for a given problem instance in a reasonable amount of time. This paper presents extensions to two papers: Clique Overlap Embedding implements an existing simulated annealing algorithm [1] with an improved guiding pattern and shifting rule; Fault Tolerant Template Embedding extends an integer programming formulation [2] to allow embedding on Chimera graphs with faulty qubits. Benchmark results show a marked improvement in embeddability for both approaches when compared to their original implementation, with each approach also showing distinct areas where they outperform the state of the art.

Contents

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Adiabatic quantum computing (AQC) is a novel computing paradigm which applies the adiabatic theorem to a system of qubits in superposition [3]. AQC has been shown to have useful real-world applications ranging from NASA rover planning [4] to machine learning [5]. There is also evidence that AQC can solve a certain class of problems faster than classical optimization [6]. In order for a problem to be solved using AQC, it must be formulated in quadratic unconstrained binary optimization (QUBO) form, and then the QUBO must be 'embedded' into the underlying annealer topology of a Chimera graph. The second step is equivalent to solving the graph minor embedding problem and it is itself an NP-hard problem. Furthermore, real-world Chimera topology contains faulty nodes which makes minor embedding more difficult. Algorithmic improvements enabling larger and more complex QUBO graphs to be embedded directly increases the practical usefulness of AQC and has significant research value. Therefore, finding more robust graph minor embedding heuristics will serve as the primary motivation for our research.

2. Background

The total energy of an adiabatic quantum system is described by a time-dependent Hamiltonian operator, $H(t)$, which is itself the (time-proportional) summation of the initial Hamiltonian and the final Hamiltonian:

$$H(t) = \left(1 - \frac{t}{T}\right) H_{initial} + \left(\frac{t}{T}\right) H_{final} \tag{1}$$

where T is the total time used for adiabatic evolution. $H_{initial}$ is the ground state of the system where all qubits are in superposition of 0 and 1. H_{final} corresponds to the 'program' under execution and is specially designed such that its ground state encodes the solution to some computational problem. By the adiabatic theorem if the energy gap between $H_{initial}$ and the first excited eigenstate is sufficiently large and the evolution of t is slow, then the system $H(t)$ eventually arrives at the ground state for H_{final} [3]. Once this process is complete, the state of all qubits can be observed thereby providing a solution to the 'program'. Performing these steps allow adiabatic quantum computers to solve optimization problems; this is termed quantum annealing (QA).

2.1. Quantum Annealing on D-Wave Machines

Programming QA algorithms on D-Wave machines is a multi-step process. Firstly, a problem must be formulated using the Quadratic Unconstrained Binary Optimization (QUBO) model:

$$E = h^T \sigma + \sigma^T J \sigma \tag{2}$$

where $h \in \mathbb{R}^n$, $J \in \mathbb{R}^{n \times n}$ and $\sigma \in \{0,1\}^n$. Here, h and J are constant coefficients and σ are decision variables. QA then equates the QUBO energy, E, to the Hamiltonian end state, H_{final}. Under the previously mentioned adiabatic theorem, the system evolves in its ground state and therefore converges towards:

$$E_{min} = \min_{\sigma} E \tag{3}$$

allowing us to find the solution values for σ which globally minimise our QUBO model. By intelligently constructing h and J in the QUBO model we can use QA to solve a variety of NP-hard and NP-complete problems, including all of Karp's 21 NP-complete problems [7]. Prior work in the formulation of problems as QUBO models is also demonstrated by [8].

It is also important to note that the QUBO model can naturally be represented as a weighted graph $G_{qubo} = \{V, E\}$ where $V = \{\sigma_1, \sigma_2 ... \sigma_n\}$. Each vertex $\sigma_i \in V$ has weight h_i and there exists an edge between σ_u and σ_v with weight $J_{u,v}$ iff $J_{u,v} > 0$. The next step in QA is to embed this graph into a D-Wave quantum annealer. The latest commercially available quantum annealer is the D-Wave 2000Q which has 2048 qubits arranged in a Chimera graph. A Chimera graph $C_{l,m,n}$ is a $m \times n$ grid of *cells*, where each *cell* is a complete bipartite graph (also known as biclique) denoted as $K_{l,l}$, where the right partition is connected horizontally to neighbouring cells and the left partition is connected vertically to neighbouring cells. A small scale Chimera graph is shown in Figure 1, the D-Wave quantum annealer contains a much larger $C_{4,16,16}$ Chimera graph.

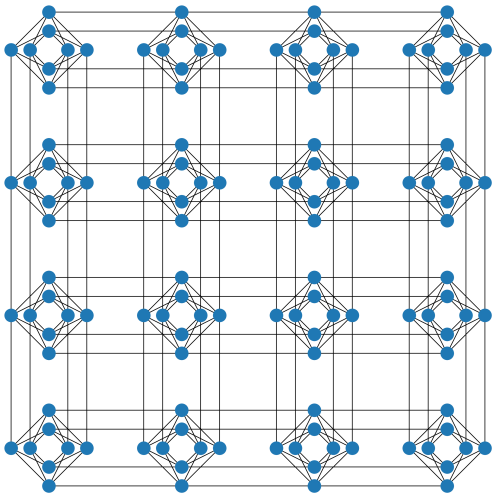
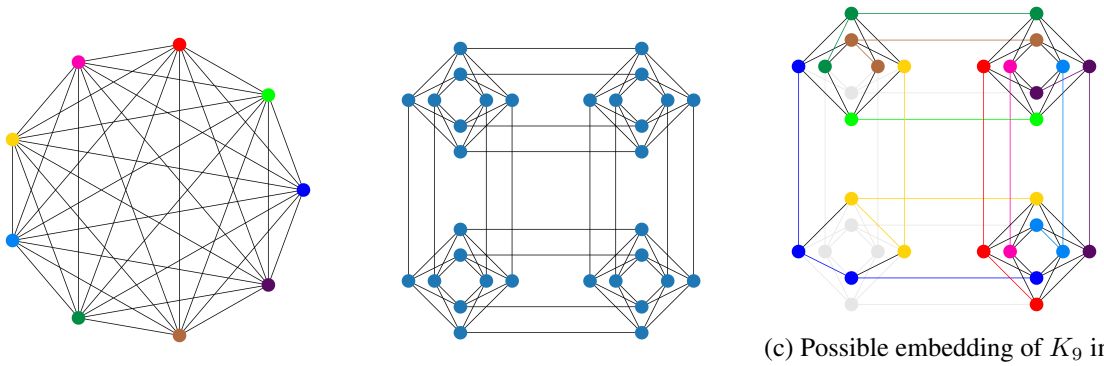


Figure 1: An example Chimera graph $C_{4,4,4}$

If the QUBO graph is an induced subgraph of the Chimera graph then the embedding would be straightforward, since every QUBO vertex can be assigned directly to a qubit. More concretely, the node weights in

h correspond to qubit biases, the edge weights in J correspond to couplings between individual qubits and $\sigma \in \{0, 1\}$ correspond to qubit spin. However, we find this is rarely the case in practise as the QUBO graph may potentially be a clique [9] or a random sparse graph [10]. In either case, [11] has demonstrated that it is still possible to embed the QUBO by assigning each vertex not to a single qubit but to a *chain* of qubits. Here, a *chain* is defined as any induced subgraph of the Chimera graph. This problem is known as the Graph Minor Embedding (henceforth abbreviated to minor embedding) problem and it is NP-hard.

There exists exact algorithms for solving minor embedding on a fixed host graph, but they are intractable for large graphs and are also non-constructive [12]. Furthermore, the Chimera is not a fixed host graph because certain vertices and edges can be disabled due to poor manufacturing yield. Therefore, existing research has focused primarily on heuristic methods. We seek to investigate existing heuristic methods and improve them in some measurable way.



(a) A clique K_9 guest graph

(b) A Chimera $C_{4,2,2}$ host graph

(c) Possible embedding of K_9 into $C_{4,2,2}$ host

Figure 2: Demonstration of a valid graph minor embedding

3. Problem Definition

In this section we formally define the graph minor embedding problem. Given a guest graph G and a host graph H, a minor embedding involves finding a mapping commonly denoted as $\phi : V(G) \to 2^{V(H)}$ such that:

1. For each vertex $v \in V(G)$, the subgraph induced by $\phi(v) \in H$ is connected.
2. For all $(u, v) \in E(G)$, we have $\phi(u) \cap \phi(v) = \emptyset$.
3. For all $(u, v) \in E(G)$, there exists $(x, y) \in E(H)$ where $x \in \phi(u)$ and $y \in \phi(v)$.

Here, ϕ is referred to as a *minor embedding* and each $\phi(v)$ where $v \in V(G)$ is referred to as a *chain*. Figure 2 demonstrates a minor embedding and highlights the chains using colors. For a problem instance it is possible there does not exist a valid embedding; it is also possible there exists multiple valid embeddings. In the second scenario, we prefer embeddings which minimise the maximum chain length. There exists experimental

evidence suggesting that long chains reduce the size of the energy gap during annealing leading to sub-optimal solutions [13], [14].

4. State of the Art

4.1. Minorminer

Minorminer (MM) [15] is an implementation of a minor embedding heuristic algorithm proposed by Cai, Macready, and Roy [12]. It is best described as a greedy destroy-and-repair heuristic which repeatedly attempts to find a valid embedding. We present an outline of the algorithm below using an example guest graph G and host graph H.

The algorithm is initialized with an empty embedding. Then the algorithm cycles over the nodes of the guest graph. For each node $g \in V(G)$, it performs the following steps:

1. If $\phi(g)$ already exists in the embedding, remove it.
2. Then attempt to find the optimal root node on the Chimera host graph to grow a tree-shaped chain for an updated $\phi(g)$. Consequently, for every Chimera node $h \in V(H)$, MM uses Dijkstra's algorithm to compute the node-weighted shortest path to all the chains necessary to satisfy the rules of a valid minor embedding (rule 3). These chains are defined as $\{\phi(f) \mid (f, g) \in E(G)\}$. To relax the search space, the algorithm permits the updated $\phi(g)$ chain to temporarily overlap with other chains. However, the weight of overlapping nodes is exponentiated to discourage this from happening.
3. The optimal root node h^* is the Chimera node which minimises the sum of shortest paths to neighbouring chains. Then MM sets $\phi(g)$ to equal a new set of nodes including $h*$ and the paths returned by the Dijkstra's search starting from h^*.

The relaxation mentioned in step 2 means that MM does not have to find a valid embedding in the first pass-through. If it eventually does find a valid embedding, the algorithm then moves to an optimization phase where it repeatedly attempts to reduce the total number of qubits used. This means that MM often returns high quality embeddings with short chains. On the other hand, if no valid embedding is found, the algorithm returns an empty embedding.

Yang and Dinneen also proposed some improvements to MM [16]. The improvements consists mainly of sorting the order of vertices in H by decreasing degree in the first iteration. The paper also proposes reducing the search space by only considering the first degree nodes from neighbouring vertex models as the root instead of all $g \in V(G)$.

From [16] we know that MM can embed a variety of famous guest graphs. However, there is a gap in research for measuring the embeddability of MM on random graphs, especially in terms of how embeddability changes with respect to graph size and graph density. We also notice a weakness of MM mentioned in [12], that is the probability of success is highly dependent on the initial selection of chains in the first pass-through. This problem can be somewhat mitigated by rerunning MM multiple times, each with a random seed. However doing so means that the runtime is increased.

4.2. Probabilistic Swap-Shift Annealing

Probabilistic Swap-Shift Annealing (PSSA) is a new minor embedding heuristic proposed by Sugie et al [1] which uses simulated annealing to explore the search space. Unlike MM which primarily embeds tree-shaped chains, PSSA embeds L-shaped chains. There are many well known clique embeddings for the Chimera graph which use L-shaped chains, one of them being the triangle clique proposed in [11]. PSSA uses two triangle cliques to create an initial embedding, this is termed as a *guiding pattern*. The left of Figure 4 demonstrates this guiding pattern.

The initial assignment of $\phi(g)$ for all $g \in V(G)$ is done by assigning guest nodes to chains in the guiding pattern. First, all the chains in the top triangular clique are assigned. This is then followed by the lower triangular clique. Finally, if there still exists unassigned g, PSSA randomly splits a chain in half and assigns g to the newly created half segment. Once ϕ is initialized PSSA performs operations from the following list following an annealing schedule:

- *Swaps*. If $\phi(i)$ and $\phi(j)$ are disconnected in the current solution but they are supposed to be connected i.e. $(i, j) \in E(G)$, then we perform a random swap so there exists an edge between $\phi(i)$ and $\phi(j)$. Note that this may disconnect other inter-chain connections so this operation isn't guaranteed to improve embedding quality.
- *Shifts along the guiding pattern*. This operation permits shifting a leaf node from one chain to another chain so long as they are within the same minor of the guiding pattern.
- *Shift against the guiding pattern*. Similar to above, this operation further relaxes the search space by allowing shifts between different minors of the guiding pattern.

PSSA tries to maximize the following objective value: $|\{(i, j) \in E(G)\}|$ where there exists $u \in \phi(i)$ and $v \in \phi(j)$ and $(u, v) \in E(H)$. At the start of the algorithm when the annealing temperature is high, PSSA may accept sub-optimal moves which decrease the objective value. This relaxation permits better exploration of the

search space to avoid local optima. Later when the annealing temperature is low, PSSA is more selective and primarily picks good moves which increases the likelihood of finding a valid embedding.

The authors in [1] implemented their own version of MM to benchmark against PSSA. The results show that PSSA performs similarly to MM when tested with random cubic graphs. For random d-regular graphs, PSSA performs slightly worse than MM for 20% edge density but outperforms MM for 50% edge density. However, these results are not entirely representative of real-world performance because the implementation of MM provided by D-Wave [15] is likely far more optimised and therefore more likely to find embeddings.

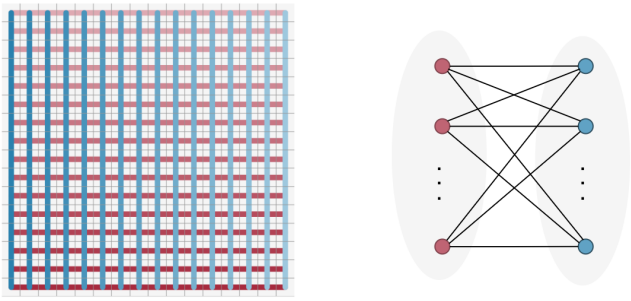
Because PSSA is initialized with triangle clique minors as a guiding pattern, it can trivially embed guest graphs with less than 64 nodes [1]. This initialization strategy also means that, unlike MM, the embedding success rate of PSSA is far less dependent on initial conditions. There are two main drawbacks when comparing PSSA to MM. Firstly, PSSA does not work on Chimera graphs with hard faults since faulty nodes break the triangle clique guiding pattern. Secondly, PSSA does not have an optimization phase which minimises the number of used qubits which could result in long chains and sub-optimal annealing solutions. There also exists a gap in research highlighted by [1] which is the discovery of new guiding patterns which can significantly improve the likelihood of PSSA finding valid embeddings.

4.3. Template Embedding

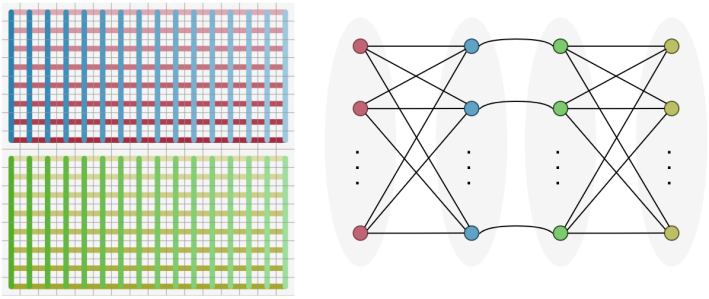
A new framework called *virtual hardware embedding* was first proposed by Goodrich et al [2]. The key concept is the introduction of the *template* graph, which is a graph with a well-defined embedding into a Chimera graph. The necessity for this abstraction is to break the difficult minor embedding problem into two simpler steps. Firstly, a virtual embedding is found between the guest graph and the template graph. Then a hardware embedding is made between the template graph and the Chimera graph. The second step is trivial because the template has a known embedding ϕ.

Figure 3a demonstrates the bipartite template embedding (BTE) model introduced by [2] using the diagrammatic notation from Bootby et al [17] (each colored line represents four distinct chains in the Chimera graph). This *template pattern* partitions the chimera graph into blue vertical chains and red horizontal chains. The induced minor of this template is shown to the right as a biclique $K_{ml,ml}$ and is termed a *template graph*. Given a guest graph G with edges $(i, j) \in E(G)$, a valid virtual embedding must assign i and j to separate partitions. Note that it does not matter which particular node from each partition is used for assignment because all nodes in a partition have the same adjacency. This greatly simplifies the search space. It is also possible for $g \in V(G)$ to be assigned to both partitions, this allows us to trivially obtain embeddings for cliques up to size K_{64}.

Serra et al further introduced the quadripartite template embedding (QTE) model (shown in Figure 3b) [18]. To embed guest graph G in QTE, each $g \in V(G)$ should be assigned to a sequence of adjacent partitions; and for all $(i, j) \in E(G)$, i and j should be assigned to partitions which are fully connected. Furthermore, Serra et al introduced integer programming (IP) formulations for both BTE and QTE which significantly improved runtime, and consequently the success rate.



(a) Bipartite



(b) Quadripartite

Figure 3: Template patterns and their induced minor template graph

Benchmark results from [18] show that BTE and QTE both have high embeddability on a variety of random graphs. The runtime of the algorithm is also fast because modern integer programming solvers are highly optimised. There exists research gaps in template embedding which are similar to PSSA, namely that template embedding does not support Chimera graphs with hard faults and there is no way to optimise the solution in terms of chain length.

4.4. Two Problems

We have identified two gaps in research both related to embeddability:

1. Fault-tolerant template embedding
2. Enhance PSSA guiding pattern to improve its embeddability

From the D-Wave manual, we know that up to 5% of nodes can be faulty [19], therefore a fault-tolerant

template embedding algorithm is needed for it be practically applicable to real-world quantum annealing hardware.

A fault-tolerant version of PSSA is also desirable, however, this is an unlikely research outcome because it breaks the algorithm's fundamental clique-based structure.

Our work will not extend MM because it already has significant research and engineering interest being the default algorithm provided by D-Wave Systems.

5. Methodology

Our methodology consists of three main components:

1. We extend the existing PSSA algorithm with a more robust guiding pattern. The new algorithm is called Clique Overlap Annealing (COA).
2. We generalize the original formulations of BTE and QTE to become fault-tolerant.
3. Finally, we benchmark our algorithmic contributions for various parameters on random graphs.

We have packaged all of our implemented algorithms in a minor embedding toolkit called Ember.

5.1. Probabilistic Swap-Shift Annealing

We implemented PSSA using Python because the language was expressive and allowed for rapid prototyping. Another benefit of Python is that it contains robust graph libraries such as NetworkX [20] and D-Wave NetworkX which simplified the modelling of Chimera graphs. Several design decisions were made in order to ensure our implementation of PSSA had good performance characteristics.

5.1.1. Objective Delta Computations

As mentioned in section 4.2, PSSA operates by generating candidate swap or shift moves. These moves are then accepted if the below expression evaluates to true. Otherwise, the move is rejected.

$$exp\left(\frac{\Delta E}{T(t)}\right) > random\_float \in [0, 1) \tag{4}$$

Here, ΔE represents the change in objective value. It is positive iff the number of edges in the guest graph satisfied by ϕ increases, and negative vice versa (See section 4.2 for exact formula). $T(t)$ is the current annealing temperature and it is a decreasing function. From this expression we can see that the probability of rejection increases as the algorithm progresses because the temperature decreases and 'good' moves become harder to find. The above observation led us to decompose both swap and shift operations into a two step

process: the first step computes ΔE and the second step actually mutates the data. Because data mutation is typically slow, this optimization provides a substantial speed-up, especially in the later stages of the algorithm where the majority of moves are rejected.

5.1.2. Datastructures

To reduce the computational complexity of the shift and swap operations, careful selection of datastructures was required. For example, the shift operation required frequent insertions and deletions from both ends of a chain, therefore a deque was used to represent a chain because it had constant time push and pop methods. An embedding ϕ was represented using a dictionary where each guest node $g \in V(G)$ mapped to a deque representing its corresponding chain. Furthermore, both shift and swap operations required frequent lookups for which guest node was assigned to a particular Chimera node. This necessitated the creation of an inverse embedding $\phi^{-1} : V(H) \rightarrow V(G)$ which was also implemented as a dictionary. We also implemented a custom weighted graph using node objects containing pointers to other node objects. This graph implementation allowed us to create an abstraction called a contact graph $C = \{V(C), E(C)\}$ where $V(C) = V(G)$ and an edge (u, v) exists in $E(C)$ if there exists an edge between $\phi(u)$ and $\phi(v)$ in the Chimera graph. Furthermore, the weight of (u, v) is equal to the number of edges between $\phi(u)$ and $\phi(v)$. This abstraction serves two purposes. Firstly, nodes in the contact graph can be relabelled whilst preserving their neighbour set to facilitate constant time swap operations. Secondly, we can efficiently compute changes in objective value by 'diffing' two contact graphs in terms of their edge set and edge weights.

5.2. Clique Overlap Annealing

One of the key contributions of our project is Clique Overlap Annealing (COA). The main idea behind COA is to use the highly connected clique overlap template introduced in [18] as a guiding pattern for PSSA to improve its embeddability.

A major limitation of PSSA is that its guiding pattern (shown to the left of Figure 4) contains two triangular cliques which have low inter-clique connectivity. This means that for guest graphs with more than 64 nodes, the likelihood of a successful embedding being found is highly dependent on the topology of the guest graph. For example, if a guest node is assigned to a chain in the left triangular clique, than all its neighbours must likewise be assigned to the left clique. For dense guest graphs, this constraint imposes a severe limit on embeddability.

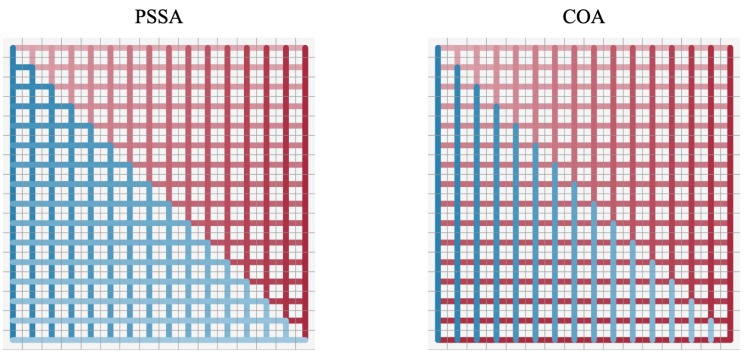


Figure 4: Comparison of the PSSA guiding pattern and the COA guiding pattern

COA instead uses the clique overlap template (shown to the right of Figure 4) as the guiding pattern. Similar to PSSA, initial assignment of guest nodes to chains begins with the right triangular clique highlighted in red. Any remaining nodes are assigned to the I-shaped chains highlighted in blue. This means that COA also trivially embeds guest graphs with less than 64 nodes. However, unlike PSSA, COA has excellent connectivity between the red clique partition and the blue partition.

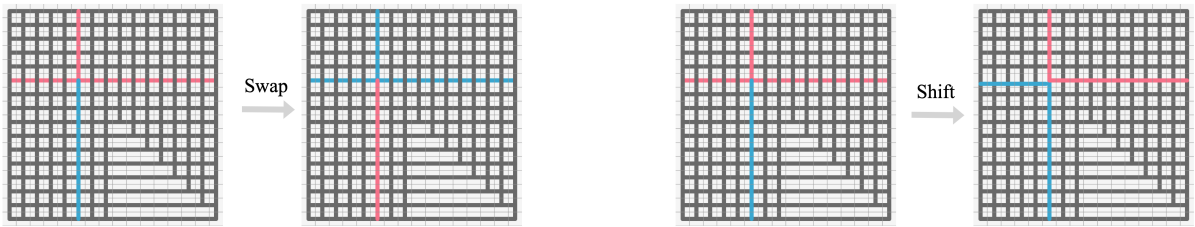


Figure 5: Demonstration of the swap and shift moves of COA

COA uses the same swap moves as PSSA, however, it introduces a new type of shift move. Both are demonstrated in Figure 5. The new shift move allows an I-shaped chain to 'steal' nodes from a T-shaped chain thereby transforming both to L-shaped chains. This provides a large amount of flexibility for minor embedding. For example, guest nodes with few neighbours can be assigned to I-shaped chains which are partially connected to both partitions. Guest nodes with a large number of neighbours can be assigned to T-shaped chains which are fully connected to all chains in the right partition and partially/fully connected to all chains in the left partition. Medium density nodes can be assigned to the L-shaped chains in either partition. The configuration of all chains in both partitions dynamically update throughout the annealing process until a valid embedding is found.

5.2.1. Implementation Details

Because COA uses the same fundamental operations as PSSA, we can reuse much of the code of PSSA. In particular, all datastructures can be reused and the swap method remains unchanged. We have therefore refactored

all common functionality and state between PSSA and COA into an abstract base class. Both PSSA and COA extend the abstract base class and use the template method pattern to override algorithmic specific methods.

5.3. Fault-Tolerant Template Embedding

A second key contribution of our project was the formulation and implementation of fault-tolerant generalizations of both BTE and QTE. We denote these new versions FT-BTE and FT-QTE respectively. These new formulations allow template embedding to be practically applied on real-world Chimera hardware which all have a nonzero number of faulty qubits [19]. Also, because both FT-BTE and FT-QTE are generalizations, they fully decompose into their original formulations for the case where the Chimera is non-faulty. This means that our algorithms can be used in-place of BTE and QTE without adding any additional runtime overhead.

5.3.1. Creating the Biadjacency Matrix

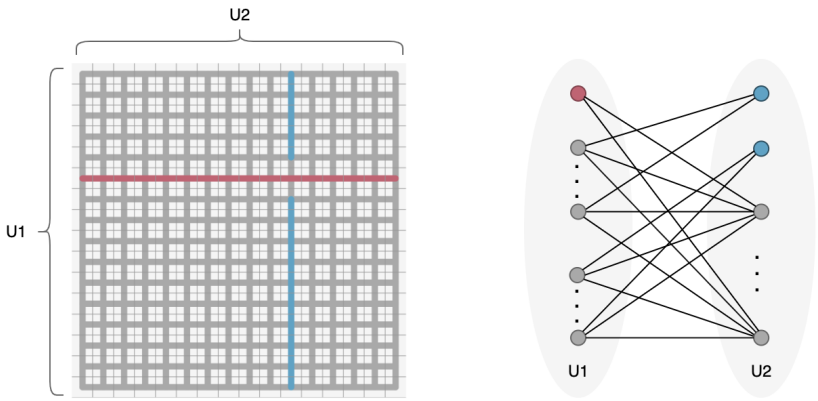
Incorporating information about faulty qubits into template embedding first requires us to reconstruct each template to reflect the removal of these qubits from the Chimera graph. This involves deleting the faulty qubit and all couplers connected to it.

We note that even a single faulty qubit will disrupt a chain in both the bipartite and quadripartite templates. A trivial solution to this problem would be to disable the entire chain containing the faulty qubit and continue with the original formulation. This, however, is very wasteful and restricts the solution space. A better approach is to divide a chain containing a fault into two separate segments whilst preserving as much edge connectivity as possible. We demonstrate this principle in Figure 6a for a faulty node in the blue chain and in Figure 6b for a faulty node in the green chain.

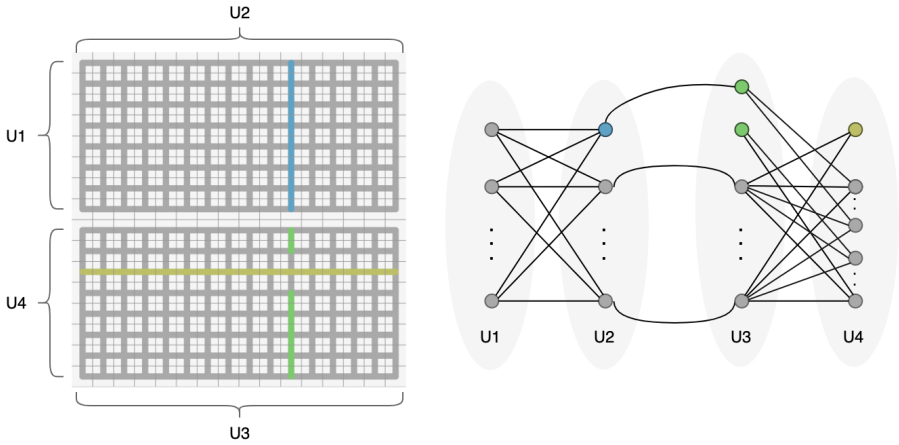
We now present a more rigorous formulation of the above transformation from a graph theoretic perspective. For the bipartite template $K_{ml,ml}$, let us denote the left partition as U_1 and the right partition as U_2. From the example in Figure 6 we have a faulty Chimera node $h \in V(H)$ and a right bipartite node $r \in U_2$ such that $h \in \phi(r)$. The faulty h has divided the $\phi(r)$ chain into a top segment and a bottom segment. We now delete r and insert two new nodes j and k into U_2 shown in Figure 6a as blue nodes. Let $\phi(j)$ be equal to the set of nodes in the top segment and $\phi(k)$ be equal to the set of nodes in the bottom segment. Next, we update the neighbour set of j by inserting an edge (l, j) where $l \in U_1$ if there exists a Chimera edge $(c, d) \in E(H)$ where $c \in \phi(j)$ and $d \in \phi(l)$. We also update the neighbour set of k in identical fashion. Notice that the edge count sum of j and k is less than that of r because certain connections have been removed entirely from the template. These correspond to the edges of h which have been deleted.

Let the four partitions of the quadripartite template be denoted as U_1, U_2, U_3 and U_4. As shown in Figure

6, faults occurring in U_1 and U_4 are handled identically to the bipartite template. However, faults in U_2 and U_3 require special treatment because only the top segments of U_3 are connected to the bottom segments of U_2. This is demonstrated in Figure 6b by the single edge between the blue and green nodes. We also note there exists an edge case where a fault occurs on the boundary between U_2 and U_3. Here, the chain-to-chain connection is removed entirely. These types of faults have a large impact on template graph connectivity.



(a) Bipartite



(b) Quadripartite

Figure 6: Impact of a single faulty node on template pattern and graph

Let τ denote a generic template graph, either bipartite or quadripartite. We represent the edges between all adjacent partitions U_x and U_y in τ using a biadjacency matrix defined as follows:

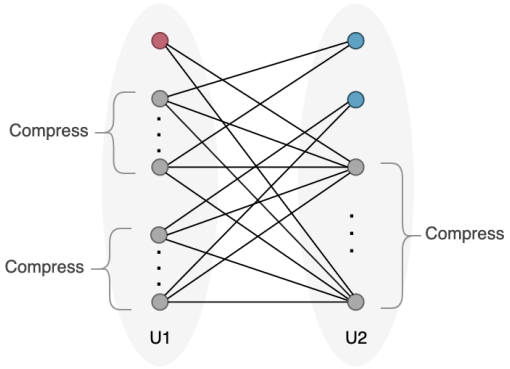
$$A_{i,j}^{x,y} = \begin{cases} 1, & \text{if there exists } (i,j) \in E(\tau) \text{ where } i \in U_x \text{ and } j \in U_y \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

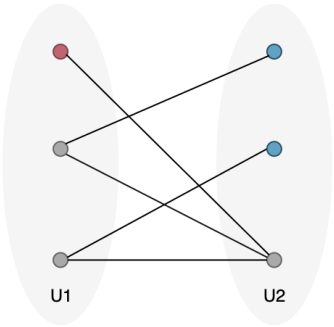
5.3.2. Induced Minor Compression

The next step of FT-BTE and FT-QTE is to compute a minor of the updated template graph which only contains nodes with a unique neighbour set. We call this process *compression*. The motivation for this step is to reduce the total number of decision variables in the formulation of both FT-BTE and FT-QTE. Our experiments show that this can reduce the solver search time by a factor of ten, turning many guest inputs from previously intractable to tractable. An example of compression is shown in Figure 7.

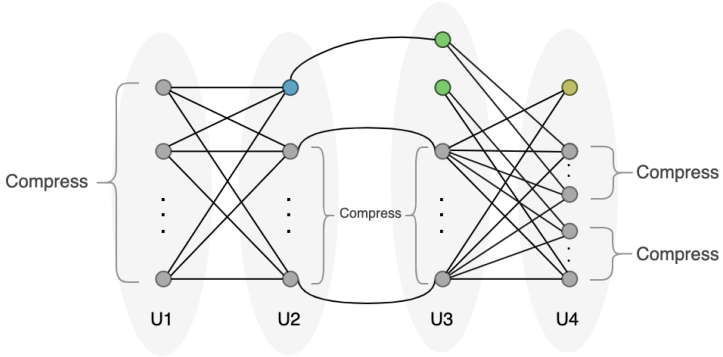
Let $N(n)$ denote the set of neighbours for a template node n. Furthermore, let all nodes $n \in \tau$ have some weight $w(n)$ which is initialized to one. To perform compression on the bipartite template graph we inspect all node combinations i, j where $i \in U_1$, $j \in U_1$ and $i \neq j$. If $N(i) = N(j)$ we remove j from U_1 and the node weight of i, $w(i)$, is incremented by one. We then repeat this algorithm for all node combinations in U_2. Persisting the node weights will be important later on to avoid loss of information in the formulation stage.

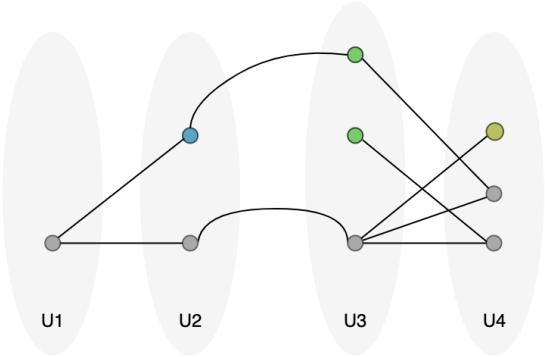
Next, we look at compression on the quadripartite template graph (Figure 7b). Similar to the previous stage, the handling of U_1 and U_4 is identical to the bipartite case. However, the compression of U_2 and U_3 is handled differently because they have two adjacent partitions instead of one. To solve this problem, we perform compression on the edges (i, j) between U_2 and U_3. The neighbour set of an edge (i, j) is defined as $N(i) \cup N(j)$. Now, the compression algorithm presented above can be adapted for U_2 and U_3 with a minor modification: instead of processing combinations of nodes, we process combinations of edges (i, j) as if they were a super-node with a single set of neighbours. An example of quadripartite compression is shown in Figure 7b.





(a) Bipartite





(b) Quadripartite

Figure 7: Demonstrating the before and after of compression on FT-BTE and FT-QTE

Finally, we update the biadjacency matrix $A^{x,y}$ for all partitions x and y by removing the rows and columns corresponding to the nodes in the template graph which have been deleted via the compression algorithm.

5.3.3. Formulation of Fault-Tolerant Bipartite Template Embedding

We now formally define FT-BTE as a constraint programming problem. Furthermore, it is a satisfiability problem where any valid solution is accepted. Therefore, there is no objective function.

Decision Variables: Consider the biadjacency matrix $A^{1,2}$ for the bipartite template with shape $(|U_1|, |U_2|)$. For every guest node $g_i \in V(G)$ and $l \in [1..|U_1|]$, let $y^1_{i,l} \in \{0, 1\}$ be a binary variable indicating whether g_i should be assigned to $n_l \in U_1$. Furthermore, for $r \in [1..|U_2|]$, let $y^2_{i,r} \in \{0, 1\}$ be a binary variable indicating whether g_i should be assigned to $n_r \in U_2$. Note that for binary variables, 1 evaluates to $True$ and 0 to $False$.

Constraints: The first constraint we impose is the connectivity constraint analogous to rule 3 of the definition for a valid minor embedding ϕ (see Section 3). For all edges in the guest graph $(u, v) \in E(G)$, there must be a corresponding edge between the assigned nodes of u and v in the bipartite template graph:

$$\bigvee_{\substack{l\in[1..|U_1|],\\ r\in[1..|U_2|],\\ A^{1,2}_{l,r}=1}} \Big(\big(y^1_{u,l} \wedge y^2_{v,r}\big) \vee \big(y^1_{v,l} \wedge y^2_{u,r}\big) \Big) \qquad \forall (g_u, g_v) \in E(G) \tag{6}$$

Note that the two conjunct terms allow for permutation of u being assigned to U_1, v being assigned to U_2 and vice versa. Next, we consider the case where a single guest node g is assigned to both $n_l \in U_1$ and $n_r \in U_2$. For this scenario, there must exist an edge between n_l and n_r in the bipartite template graph. We therefore apply the following validity constraint:

$$y^1_{i,l} + y^2_{i,r} \leq 1 + A^{1,2}_{l,r} \qquad \forall g_i \in V(G) \quad \forall l \in [1..|U_1|] \quad \forall r \in [1..|U_2|] \tag{7}$$

To avoid redundant assignments, we enforce the constraint that each guest node $g \in V(G)$ cannot be assigned to more than a single template node in each partition:

$$\sum_{l=1}^{|U_1|} y^1_{i,l} \leq 1 \qquad \sum_{r=1}^{|U_2|} y^2_{i,r} \leq 1 \qquad \forall g_i \in V(G) \tag{8}$$

The final constraint is rather complex because it operates on the template node weights w obtained from the compression stage. The key concept is that mapping guest nodes to template nodes is a many-to-one relationship because each template node in our formulation is the representation of several redundant chains. The number of guest nodes allowed to map to a single template node is bounded by the latter's weight. Therefore we have:

$$\begin{aligned} \sum_{g_i \in V(G)} y^1_{i,l} &\leq w(n_l) \qquad \forall n_l \in U_1 \\ \sum_{g_i \in V(G)} y^2_{i,r} &\leq w(n_r) \qquad \forall n_r \in U_2 \end{aligned} \tag{9}$$

5.3.4. Formulation of Fault-Tolerant Quadripartite Template Embedding

We now present a similar formulation for FT-QTE. Note that FT-QTE is a generalization of FT-BTE; any guest graph that can be embedded on the bipartite template should theoretically be embeddable on the quadripartite template.

Decision Variables: The quadripartite template is parameterized by three biadjacency matrices: $A^{1,2}$ with shape $(|U_1|, |U_2|)$, $A^{2,3}$ with shape $(|U_2|, |U_3|)$ and $A^{3,4}$ with shape $(|U_3|, |U_4|)$. For every guest node $g_i \in V(G)$, $p \in \{1, 2, 3, 4\}$ and $n_x \in U_p$, let $y^p_{i,x} \in \{0, 1\}$ be a binary variable for whether guest node g_i should be

assigned to n_x in partition U_p.

Constraints: We first establish a connectivity constraint which forces every guest graph edge to have a corresponding template graph edge. This constraint is only enforced between two fully-connected partitions.

$$\begin{aligned} c_{1,2} &= \bigvee_{\substack{l\in[1..|U_1|],\\ r\in[1..|U_2|],\\ A^{1,2}_{l,r}=1}} \Big(\big(y^1_{u,l} \wedge y^2_{v,r}\big) \vee \big(y^1_{v,l} \wedge y^2_{u,r}\big)\Big) \\ c_{3,4} &= \bigvee_{\substack{l\in[1..|U_3|],\\ r\in[1..|U_4|],\\ A^{3,4}_{l,r}=1}} \Big(\big(y^3_{u,l} \wedge y^4_{v,r}\big) \vee \big(y^3_{v,l} \wedge y^4_{u,r}\big)\Big) \\ & c_{1,2} \vee c_{3,4} \qquad \forall (g_u, g_v) \in E(G) \end{aligned} \tag{10}$$

Similar to FT-BTE, we enforce that if $g \in V(G)$ is assigned to $n_x \in U_p$ and $n_y \in U_q$, then there must exist an edge between n_x and n_y in the template graph.

$$\begin{aligned} y^p_{i,l} + y^q_{i,r} \leq 1 + A^{p,q}_{l,r} \qquad & \forall g_i \in V(G) \\ & \forall (p,q) \in \{(1,2),(2,3),(3,4)\} \\ & \forall l \in [1..|U_p|] \\ & \forall r \in [1..|U_q|] \end{aligned} \tag{11}$$

Following on from the previous constraint, we also enforce that U_p and U_q are pairwise contiguous. Otherwise, there would be no connections between the corresponding chains in the Chimera graph, leading to an invalid embedding.

$$\left.\begin{aligned} & f(1) + f(3) - f(2) \leq 1 \\ & f(2) + f(4) - f(3) \leq 1 \\ & f(1) + f(4) - f(2) - f(3) < 1 \end{aligned}\right\} f(p) = \sum_{x=1}^{|U_p|} y^p_{i,x} \qquad \forall g_i \in V(G) \tag{12}$$

The remaining two constraints are identical to FT-BTE. Firstly, every guest node cannot be assigned to more than one template node per partition.

$$\sum_{x=1}^{|U_p|} y^p_{i,x} \leq 1 \qquad \forall g_i \in V(G) \qquad \forall p \in \{1,2,3,4\} \tag{13}$$

And every template node cannot have more guest nodes assigned to it than its weight, which corresponds to the number of redundant chains represented by that template node.

$$\sum_{g_i \in V(G)} y_{i,x}^{p} \leq w(n_x) \qquad \forall p \in \{1,2,3,4\} \qquad \forall n_x \in U_p \tag{14}$$

5.3.5. Implementation of Fault-Tolerant Template Embedding

The formulations for both FT-BTE and FT-QTE were initially implemented as a quadratic integer programs (IPQ) in Gurobi [21]. The only quadratic constraint was the connectivity constraint, which required interactions between two decision variables. IPQ problems are in general harder to solve than normal integer programs because they are likely to have non-convex objective functions [21]. During tests, we confirmed that quadratic programming had performance issues because embedding small, low-density graphs would take 20 minutes or longer. An attempt to improve performance was made by linearizing the quadratic constraints. However, this introduced too many decision variables which made the problem intractable.

We chose an alternative approach, which was to implement the formulations as constraint-programs using Google OR-Tools [22]. More specifically, we used the CP-SAT solver from OR-Tools which natively supports boolean decision variables and can accept boolean expressions as constraints. This enabled us to encode the connectivity constraint directly without making any additional modifications. CP-SAT operates using a highly optimised SAT solver with lazy-clause generation. This enabled embeddings to be found significantly faster. For the example mentioned in the previous paragraph, we were able to reduce the runtime from over 20 minutes to under a minute.

Another performance concern was that the OR-Tools Python API constructed constraints using protocol buffers before sending the data to the C++ solver. This conversion process was quite slow, often taking five times longer than the time it took to find the actual embedding. To solve this problem, we ported all the code to use the OR-Tools C++ API which directly creates the constraints in-memory. Then we used Pybind11 to invoke the C++ functions from Python as baked-in native extensions. This approach allowed us to combine the high performance of C++ with the ease-of-use of Python.

5.4. Experimental Setup

A series of benchmarks was designed to measure and compare the embeddability of all algorithms discussed so far. These algorithms are: MM, PSSA, COA, FT-BTE and FT-QTE. The benchmark involved running each algorithm on a suite of random test graphs and recording the following statistics regarding each embedding:

- Whether the embedding was successful
- The walltime of finding a valid embedding

- The total number of qubits used
- The median chain length

Since the focus of our project was on embeddability, we were primarily interested in the first two metrics, however, the last two metrics were still useful to gain an overall understanding of the advantages and disadvantages of each algorithm.

The suite of random test graphs included four different types of graphs; these are: Barabasi-Albert graphs, Erdos-Renyi graphs, d-regular graphs and noisy bipartite graphs. Furthermore, each graph was parameterized by edge density and node count. The edge density could be either low (0.25), medium (0.5) or high (0.75) and the node count ranged from 65 to 95. This test suite was also used by [18] and [2] which gives us a useful baseline for comparison. We demonstrate the edge density characteristics of all the aforementioned graphs in Appendix A.

In terms of hosts, we used a mixture of faulty and non-faulty D-Wave 2000Q Chimera graphs, $C_{4,16,16}$. The faulty Chimeras were further split into two variants, a DW-2000Q-6 variant with 7 faulty nodes and DW-2000Q-QuAIL variant with 17 faulty nodes. Both the aforementioned variants are topologies of real quantum annealers maintained by D-Wave [19]. The exact position of all faulty qubits in these graphs are shown in Appendix B. We benchmarked MM, FT-BTE and FT-QTE on faulty Chimera graphs since they were the only algorithms which supported faults.

The procedure for the benchmark was as follows: each algorithm was given 5 attempts to embed a guest graph of a certain type, edge density and size. Each attempt used a different seed for the guest graph; this was kept constant between all algorithms. We set a timeout for both FT-BTE and FT-QTE since both were satisfiability algorithms which could run forever. The timeout for embedding on non-faulty Chimera was set to 30 seconds and the timeout for faulty Chimera was set to 250 seconds. We also limited PSSA and COA to 10^6 iterations. If an embedding was found by an algorithm, we increased the size of the guest graph by one and repeated the trial. Otherwise, we stopped the current trial and moved to the next test configuration. The benchmarks were run on an i5-3570k processor with 8gb of ram.

6. Results and Discussion

6.1. Embedding on Non-faulty Chimera Graphs

Table 1 shows the maximum size of each guest graph configuration embedded by all algorithms on to a non-faulty DW-2000Q host graph. The algorithm which embedded the largest graph for a certain configuration is highlighted in bold. The data shows a clear lead for COA in terms of overall embeddability, with MM and FT-BTE tied as the second best option. The only category where MM out embeds COA is in medium to high

density noisy-bipartite graphs and only by a small margin. We note that PSSA performs significantly worse than COA and it fails to embed certain categories of graphs entirely. This shows strong evidence that the clique overlap guiding pattern used in COA out-embeds the triangular clique guiding pattern used in PSSA.

Table 1: Embeddability of algorithms on non-faulty DW-2000Q

Random graph	Density	FT-BTE	FT-QTE	MM	COA	PSSA
Barabási-Albert	low	79	77	78	**86**	66
	medium	86	86	83	**95**	67
	high	**95**	**95**	**95**	**95**	67
D Regular	low	77	75	80	**82**	65
	medium	72	71	69	**74**	65
	high	68	68	67	**69**	-
Erdős-Reyni	low	79	75	80	**85**	65
	medium	72	72	70	**75**	65
	high	69	68	67	**70**	-
Noisy Bipartite	low	92	75	93	**95**	69
	medium	86	79	**89**	86	66
	high	80	78	**84**	83	-

We note that FT-BTE matches or out-embeds FT-QTE in every category which aligns with results obtained from [18]. The likely justification is that the simpler formulation of FT-BTE increased the likelihood of it finding an embedding before the timeout, even though FT-QTE is a generalization of FT-BTE. An unsurprising result is that FT-BTE embeds significantly more graphs than FT-QTE for the noisy-bipartite graph type, especially for the low density variants. Noisy-bipartite graphs contain a mixture of bipartite edges and non-bipartite edges following a ratio of 5:1. The bipartite edges should be easily embeddable into FT-BTE since it naturally aligns with the topology of the bipartite template and the remaining edges can be assigned to both partitions forming a fully-connected chain.

It is somewhat surprising then to see that MM out-embeds FT-BTE in noisy bipartite graphs where we would expect the latter to have a natural advantage. One possible reason for this is that the noisy-bipartite graph has a comparatively low median neighbour count per node (see Figure 10). This allows MM to find embeddings more easily because the tree-shaped chains it constructs during the search phase has less branches, and therefore it is more likely that a route will be found with no overlapping nodes. The second most embeddable graph type for MM are Barabasi-Albert graphs, which also have a low median neighbour count. This provides some evidence that the above argument is correct.

Both MM and FT-BTE embedded the exact same number of guest graphs. For three of the four graph types, FT-BTE has better embeddability than MM on medium and high density variants. On the other hand, MM generally showed better embeddability on low density variants. This result aligns with our expectations,

since FT-BTE template is designed to have high connectivity whereas MM relies on some degree of randomness to find valid connections. However, unlike FT-BTE, MM is flexible in terms of chain length and shape which provides an advantage on low density graphs.

6.1.1. Runtime Performance Scaling

Although COA provides the best embeddability on non-faulty graphs, our results indicate that it is not always the best algorithm to use in every scenario because the running time of the algorithm rises sharply for large guest graphs. Figure 8 plots the running time of all algorithms as a function of node count for certain combinations of graph types and edge densities. These results are representative of the general trend we see when measuring runtime performance.

FT-BTE presents the best runtime out of all the algorithms due to its (comparatively) straightforward formulation and the highly-optimised implementation of the CP-SAT solver. In fact, the runtime appears to be constant which suggests that FT-BTE has exhaustively searched all possible node assignments. FT-QTE, in comparison, has some spikes in runtime probably due to its more complex constraints and greater number of decision variables.

COA's runtime generally performs on par, or even slightly better, compared to MM up to a certain guest graph size; then it hits an inflection point and rises sharply. There does not appear to be any way to determine where the inflection point is located, as it can range anywhere between 75 to 90 nodes. We note that the increase in runtime is sometimes an acceptable trade-off because it allows COA to embed large graphs which other algorithms cannot embed. This is the case for medium density Barabasi-Albert graphs and low density Erdos-Renyi graphs. However, for low density d-regular graphs and high density noisy bipartite graphs, it would be smarter to select MM because it can embed graphs of the same size as COA up to 2 minutes faster.

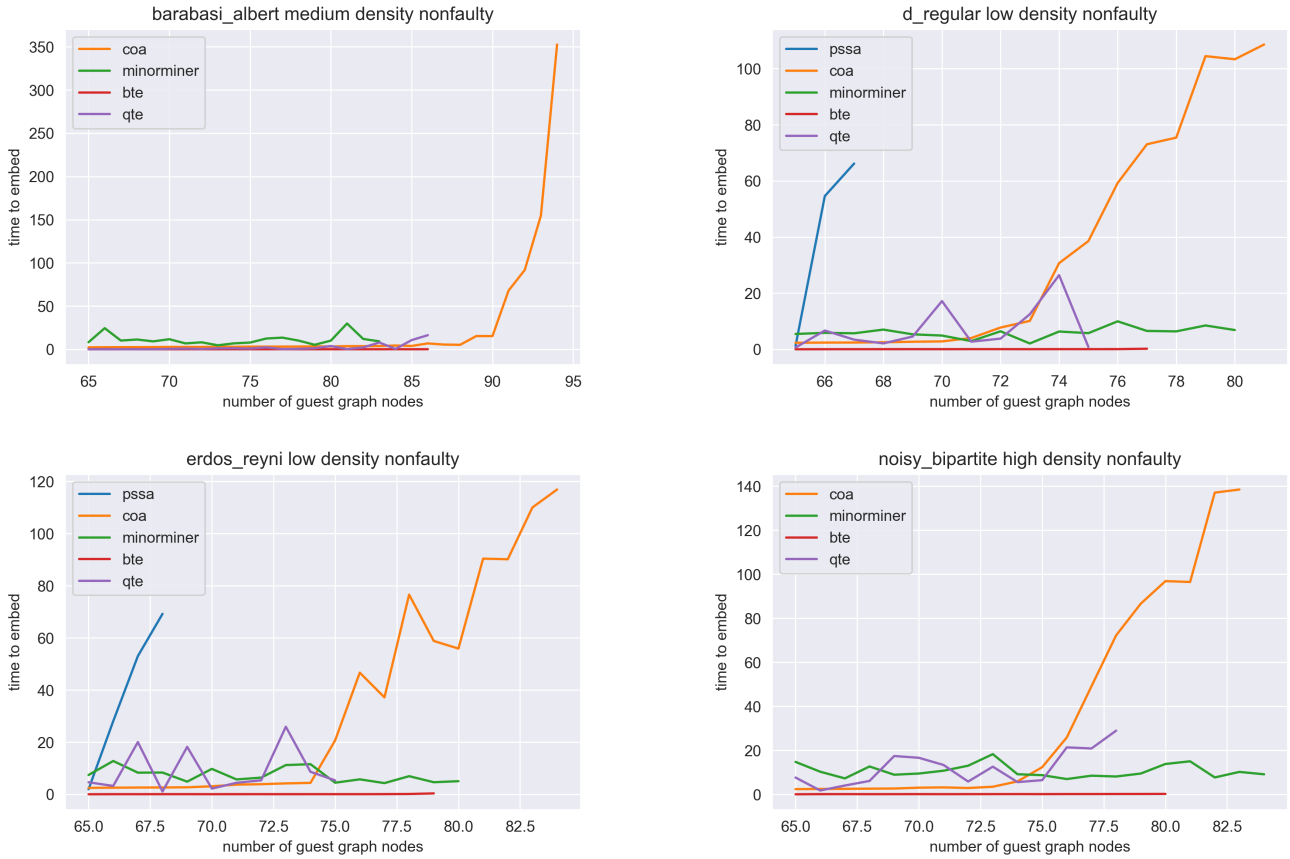


Figure 8: Running time (in seconds) versus size for a selection of guest graphs embedded on non-faulty Chimera hosts

6.2. Embedding on Faulty Chimera Graphs

The embeddability results of all fault-tolerant algorithms are presented in Table 2. Again, the algorithm which embedded the largest guest graph configuration for a certain host topology is highlighted in bold.

We can compare data points between Table 1 and Table 2 to analyze the impact of faulty nodes on embeddability. We note that for seven faults, FT-BTE embeds 27 less graphs, FT-QTE embeds 36 less graphs and MM embeds just six less graphs. This provides strong evidence that MM is far less impacted by faulty nodes than FT-BTE and FT-QTE. The above phenomenon is even more exaggerated on the DW-2000Q-QuAIL host with 17 faults. Here, FT-BTE embeds 61 less graphs, FT-QTE embeds 83 less graphs and MM embeds 15 less graphs. This result is unsurprising because MM's high embeddability on faulty Chimera hosts is one of its inherent strengths. The fact that MM uses Dijkstra's shortest path algorithm to search for valid chains means that it can *route* around faulty nodes. On the other hand, FT-BTE and FT-QTE do not perform any such routing and instead rely on the template graph being mostly intact (in terms of connections between partitions) after removing faults. This strategy does not scale well to a large number of faults because the probability of

Table 2: Embeddability of algorithms on faulty DW-2000Q-6 and DW-2000Q-QuAIL

		D-Wave 2000Q-6 (7 faults)			D-Wave 2000Q-QuAIL (17 faults)		
Random graph	Density	FT-BTE	FT-QTE	MM	FT-BTE	FT-QTE	MM
Barabási-Albert	low	75	74	**76**	74	70	**77**
	medium	**87**	78	81	78	70	**80**
	high	**95**	86	**95**	87	84	**95**
D Regular	low	76	75	**78**	74	69	**79**
	medium	**70**	69	**70**	**70**	68	**70**
	high	**68**	**68**	67	**67**	66	65
Erdős-Reyni	low	77	76	**79**	76	70	**79**
	medium	**70**	**70**	**70**	69	67	**70**
	high	**68**	67	67	**67**	**67**	**67**
Noisy Bipartite	low	82	77	**95**	82	73	**90**
	medium	81	72	**89**	77	67	**87**
	high	79	71	**82**	73	65	**81**

particularly 'harmful' fault sequences is increased: for example, consider two faults sitting on a single chain which divides the chain into three segments instead of just two as demonstrated in Figure 11.

Even though MM has higher embeddability on faulty hosts in general, we notice that for all graph types except noisy bipartite, FT-BTE out embeds MM for both medium and high variants on the D-Wave 2000Q-6. This is unsurprising because we notice the same trend in non-faulty hosts and the explanations remain the same. However, an important trend is that the advantage of FT-BTE over MM in denser variants shrinks as the number of faults increases; in fact, on the D-Wave 2000Q-QuAIL with 17 faults, there does not exist an advantage at all. This provides further evidence that the embeddability of FT-BTE degrades substantially as the number of faults increases, whereas MM is significantly less affected.

6.2.1. Runtime Performance Scaling

We compare the impact of faulty nodes on embedding runtime for two representative guest graph configurations in Figure 9. The runtime of all algorithms follow a similar trajectory when the size of the guest graph is small, however FT-BTE and FT-QTE eventually hit an embedding bottleneck demonstrated by large fluctuating spikes in runtime. Both template-based approaches rely on combinatorial search to find a single valid embedding out of all possible embeddings. Faulty nodes reduce the number of valid embeddings and make combinatorial search more difficult on large graphs, hence, we see an exponential blow-up in runtime resulting in spikes. MM also experiences fluctuations (especially in the bottom right example), however, the effect is not as drastic.

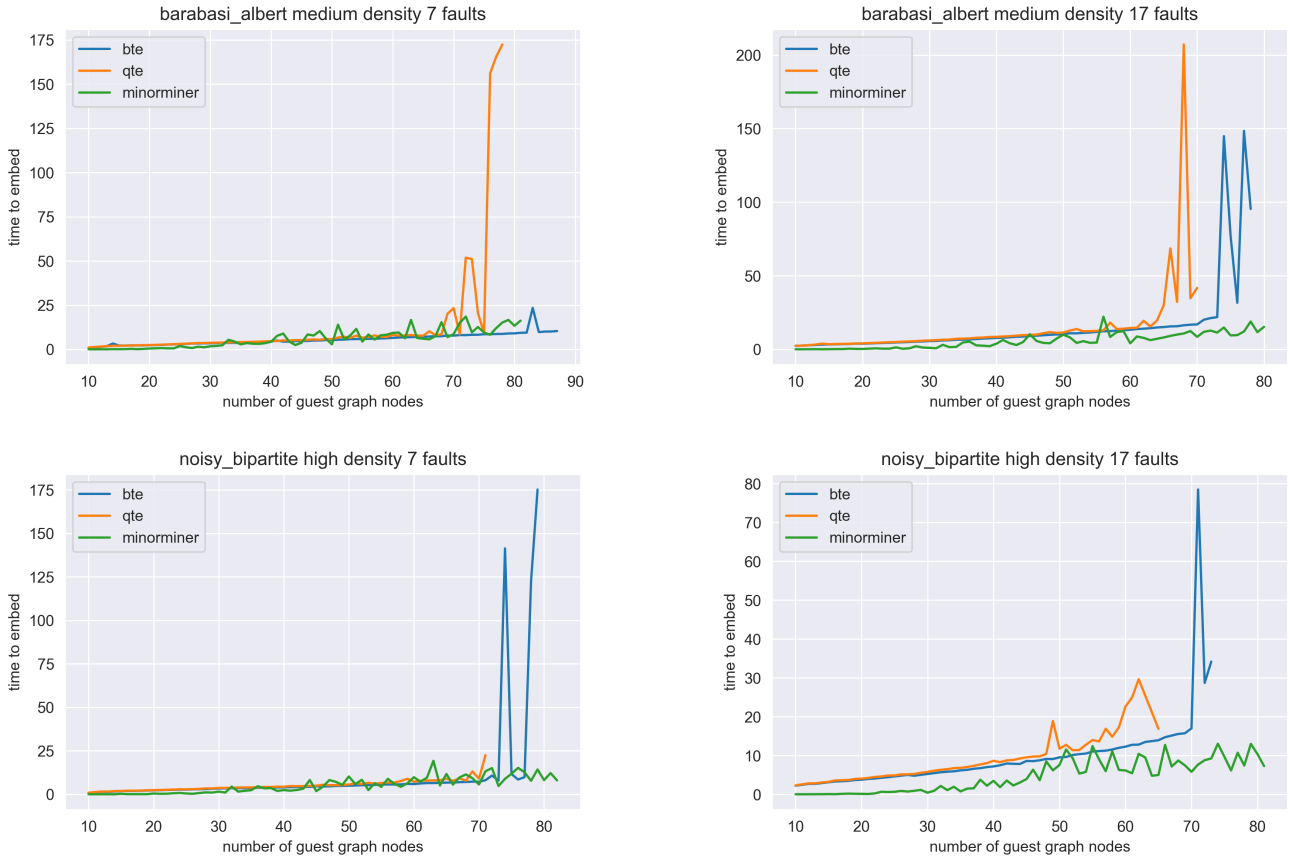


Figure 9: Running time (in seconds) versus size for a selection of guest graphs embedded on faulty Chimera hosts

Furthermore, we note that the initial spike in runtime for both FT-BTE and FT-QTE occurs earlier (at a smaller guest graph size) when there are more faults. In addition, FT-QTE experiences spiking earlier than FT-BTE. Both observations can be explained by an increase in the difficulty of combinatorial search, however, the source of complexity is different. For the former, it is because faulty nodes disrupt the template graph, whilst for the latter, it is because FT-QTE has more complex constraints and more decision variables.

6.3. Guest Graph Differences

The results in Table 1 and 2 show that the input graph type and edge density has a definite impact on an algorithm's embeddability. Interestingly we found that the relationship between edge density and embeddability for Barabási-Albert graphs is inverse to that of the other three. This trend is observed in all algorithms and host graph fault conditions. We investigated this by generating a dataset resembling that used for benchmarking. For each graph, the median neighbourhood size of per node is shown in Appendix A. Observe in the Barabási-Albert chart our input edge density is inversely related to the median neighbourhood size. We discovered NetworkX's random graph generator for Barabási-Albert uses the edge density input to fix the total number of

edges on the graph, these edges are then created using a fixed probability for node preference. Note that both [23] and [18] forked the NetworkX generator to create Barabási-Albert graphs that was closer in its connectivity to an Erdős-Reyni graph.

6.4. Other Embedding Metrics

Embedding quality metrics were measured during benchmarking, however as mentioned in Section 4.4, optimizing embedding quality was not one of our research objectives. Nevertheless, we present a summary of how MM compares to our contributed algorithms in this area because it is useful information for practitioners of QA.

There exists a linear relationship between total number of qubits used and guest graph size for all algorithms. MM, FT-BTE and FT-QTE share a similar gradient for the aforementioned trend. However, for a fixed guest graph size, the embeddings found by MM generally use half as many qubits as the corresponding embeddings found by FT-BTE and FT-QTE. COA has a smaller gradient by a factor of 2, this means for large guest graphs, the difference in total number of qubits used between COA and MM is almost indistinguishable.

The median chain length of FT-BTE, FT-QTE and COA are generally fixed values because all three algorithms use a structured pattern to find embeddings. FT-BTE and FT-QTE both use on average 32 qubits per chain which is unsurprising because this is the size of a plus-shaped chain in the D-Wave 2000Q Chimera topology (a common scenario where a single guest node maps to multiple partitions in the template graph). COA chains are shorter and hover at around 24 qubits which is the length of a T-shaped chain in the above topology. Finally, MM median chain length increases linearly with the size of the guest graph but on average it is shorter than FT-BTE and FT-QTE by a factor of 2.

7. Conclusions

This paper presented several contributions to the field of heuristic graph minor embedding algorithms as applied to the D-Wave Chimera host topology.

Our first major contribution is COA, a modification of the PSSA algorithm first proposed by [1]. There exist two key differences between COA and PSSA. Firstly, COA's guiding pattern was designed to have superior inter-clique connectivity. We also introduced an alternative shift move as part of COA which allows conversion between three different types of chains. These two features combined allow COA to embed guests with up to 28 more nodes than PSSA on a non-faulty D-Wave 2000Q host; this is a significant improvement in embeddability.

Our second major contribution is the formulation and implementation of FT-BTE and FT-QTE which extends the original template embedding [18] for Chimera hosts with faulty nodes. We know that real-world QA

processors have Chimera graphs with faults due to manufacturing defects [19], therefore, our contribution has improved embeddability by allowing template embedding to operate on real-world non-ideal QA processors.

An extensive suite of benchmarks was conducted, comparing all the heuristic algorithms identified in our literature review as well as our own contributions. For idealized, non-faulty Chimera hosts, our results indicate that COA has the best embeddability and FT-BTE has the best runtime. For real-world, faulty Chimera hosts, FT-BTE has better embeddability than MM for certain medium and high density guest graphs, but in general MM embeds more graphs and has a better runtime on particularly hard guest graph instances. MM also produces embeddings with the shortest chains out of all the algorithms we tested.

Although we have accomplished all of our research goals described in Section 4.4, we acknowledge that there still exists weaknesses in our algorithmic contributions: COA, FT-BTE and FT-QTE. None of these algorithms are currently optimised to reduce chain length. Furthermore, COA does not yet support faulty Chimera hosts. These limitations still need to be overcome to facilitate the practical application of our algorithms for minor embedding QUBO graphs on D-Wave machines.

8. Future Work

A key area of further research is to extend COA so that it can operate on faulty Chimera hosts. This work will likely require a further generalization of the guiding pattern as well as the development of more sophisticated simulated annealing moves which can route around faulty nodes.

We did not have the resources in this research project to investigate improving embedding quality by reducing median chain length, however, this is an important area of research because long chains can often lead to sub-optimal solutions being returned by QA [14]. Therefore, we recommend this as future work for COA, FT-BTE and FT-QTE. There is existing research in this area by [2] where chains are shortened by deleting unused edges in the template graph.

Further runtime performance optimizations can be made for all of our proposed algorithms. In the case of COA, porting the code from Python to C++ will gain a significant speedup. For FT-BTE and FT-QTE, the development of problem-specific optimization code and parallelization can improve runtime over OR-Tools.

Finally, our project was limited to testing on the Chimera host topology and we did not consider D-Wave's upcoming Pegasus topology. The Pegasus P16 topology will contain 5640 qubits (compared to the 2048 of D-Wave 2000Q Chimera topology) and will also have a higher treewidth [24]. It would be interesting to see how well our proposed algorithms can be adapted to work on this new architecture.

Acknowledgements

We would like to thank our project supervisor Dr Michael Dinneen for his guidance in the general direction of the project and associate professor Andrew Mason for his expert advice in modelling the fault-tolerant template embedding constraints.

References

[1] Y. Sugie, Y. Yoshida, N. Mertig, T. Takemoto, H. Teramoto, A. Nakamura, I. Takigawa, S.-i. Minato, M. Yamaoka, and T. Komatsuzaki, *Minor-embedding heuristics for large-scale annealing processors with sparse hardware graphs of up to 102,400 nodes*, 2020. arXiv: 2004.03819 [quant-ph].

[2] T. D. Goodrich, B. D. Sullivan, and T. S. Humble, "Optimizing adiabatic quantum program compilation using a graph-theoretic framework," *Quantum Information Processing*, vol. 17, no. 5, Apr. 2018, ISSN: 1573-1332. DOI: 10.1007/s11128-018-1863-4. [Online]. Available: http://dx.doi.org/10.1007/s11128-018-1863-4.

[3] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum computation by adiabatic evolution*, 2000. arXiv: quant-ph/0001106 [quant-ph].

[4] E. G. Rieffel, D. Venturelli, B. O'Gorman, M. B. Do, E. M. Prystay, and V. N. Smelyanskiy, "A case study in programming a quantum annealer for hard operational planning problems," *Quantum Information Processing*, vol. 14, no. 1, pp. 1–36, Dec. 2014, ISSN: 1573-1332. DOI: 10.1007/s11128-014-0892-x. [Online]. Available: http://dx.doi.org/10.1007/s11128-014-0892-x.

[5] H. Neven, G. Rose, and W. G. Macready, *Image recognition with an adiabatic quantum computer i. mapping to quadratic unconstrained binary optimization*, 2008. arXiv: 0804.4457 [quant-ph].

[6] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," *Physical Review X*, vol. 8, no. 3, Jul. 2018, ISSN: 2160-3308. DOI: 10.1103/physrevx.8.031016. [Online]. Available: http://dx.doi.org/10.1103/PhysRevX.8.031016.

[7] A. Lucas, "Ising formulations of many np problems," *Frontiers in Physics*, vol. 2, 2014, ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005. [Online]. Available: http://dx.doi.org/10.3389/fphy.2014.00005.

[8] M. J. Dinneen, M. R. Hooshmandasl, and R. Hua, "Formulating mixed dominating set problems for adiabatic quantum computers," Department of Computer Science, The University of Auckland, New Zealand, Tech. Rep., 2017.

[9] E. Boyda, S. Basu, S. Ganguly, A. Michaelis, S. Mukhopadhyay, and R. R. Nemani, "Deploying a quantum annealing processor to detect tree cover in aerial imagery of california," *PloS one*, vol. 12, no. 2, e0172505, 2017.

[10] A. Perdomo-Ortiz, J. Fluegemann, S. Narasimhan, R. Biswas, and V. Smelyanskiy, "A quantum annealing approach for fault detection and diagnosis of graph-based systems," *The European Physical Journal Special Topics*, vol. 224, no. 1, pp. 131–148, Feb. 2015, ISSN: 1951-6401. DOI: 10.1140/epjst/e2015-02347-y. [Online]. Available: http://dx.doi.org/10.1140/epjst/e2015-02347-y.

[11] V. Choi, "Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design," *Quantum Information Processing*, vol. 10, no. 3, pp. 343–353, Oct. 2010, ISSN: 1573-1332. DOI: 10.1007/s11128-010-0200-3. [Online]. Available: http://dx.doi.org/10.1007/s11128-010-0200-3.

[12] J. Cai, W. G. Macready, and A. Roy, "A practical heuristic for finding graph minors," *arXiv preprint arXiv:1406.2741*, 2014.

[13] M. J. Dinneen and R. Hua, "Formulating graph covering problems for adiabatic quantum computers," in *Proceedings of the Australasian Computer Science Week Multiconference*, 2017, pp. 1–10.

[14] J. P. Pinilla and S. J. Wilton, "Layout-aware embedding for quantum annealing processors," in *International Conference on High Performance Computing*, Springer, 2019, pp. 121–139.

[15] *Minorminer repository*. [Online]. Available: https://github.com/dwavesystems/minorminer.

[16] Z. Yang and M. J. Dinneen, *Graph minor embeddings for d-wave computer architecture*. Centre for Discrete Mathematics and Theoretical Computer Science, 2016.

[17] T. Boothby, A. D. King, and A. Roy, "Fast clique minor generation in chimera qubit connectivity graphs," *Quantum Information Processing*, vol. 15, no. 1, pp. 495–508, Oct. 2015, ISSN: 1573-1332. DOI: 10.1007/s11128-015-1150-6. [Online]. Available: http://dx.doi.org/10.1007/s11128-015-1150-6.

[18] T. Serra, T. Huang, A. Raghunathan, and D. Bergman, *Template-based minor embedding for adiabatic quantum optimization*, 2019. arXiv: 1910.02179 [cs.DS].

[19] Jun. 2018. [Online]. Available: https://support.dwavesys.com/hc/en-us/articles/360005268633-QPU-Specific-Physical-Properties.

[20] A. Hagberg, P. Swart, and D. Chult, "Exploring network structure, dynamics, and function using networkx," Jan. 2008.

[21] L. Gurobi Optimization, *Gurobi optimizer reference manual*, 2020. [Online]. Available: http://www.gurobi.com.

[22] L. Perron and V. Furnon, *Or-tools*, version 8.0, Google. [Online]. Available: https://developers.google.com/optimization/.

[23] T. Goodrich, E. Horton, and B. Sullivan, "Practical graph bipartization with applications in near-term quantum computing," May 2018.

[24] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, *Next-generation topology of d-wave quantum processors*, 2020. arXiv: 2003.00133 [quant-ph].

Appendix

A. Characteristics of Guest Graphs used for Benchmarking

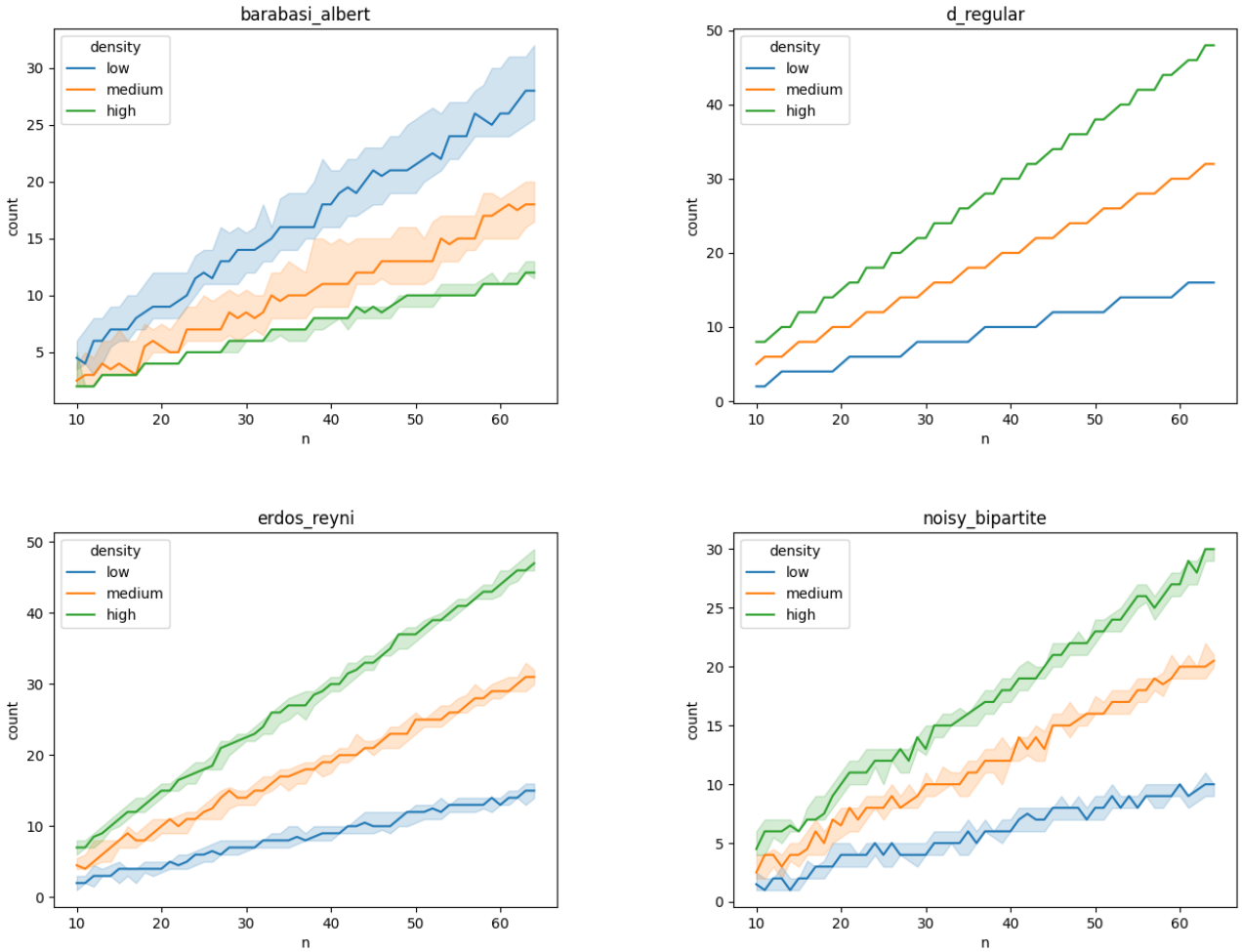
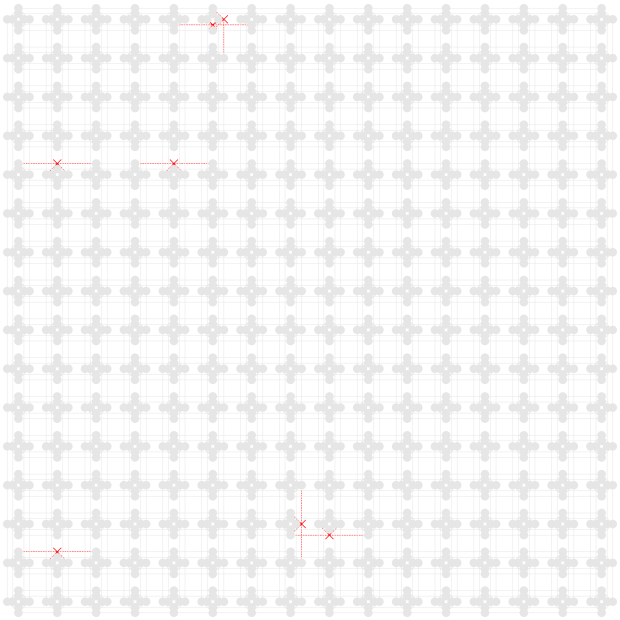


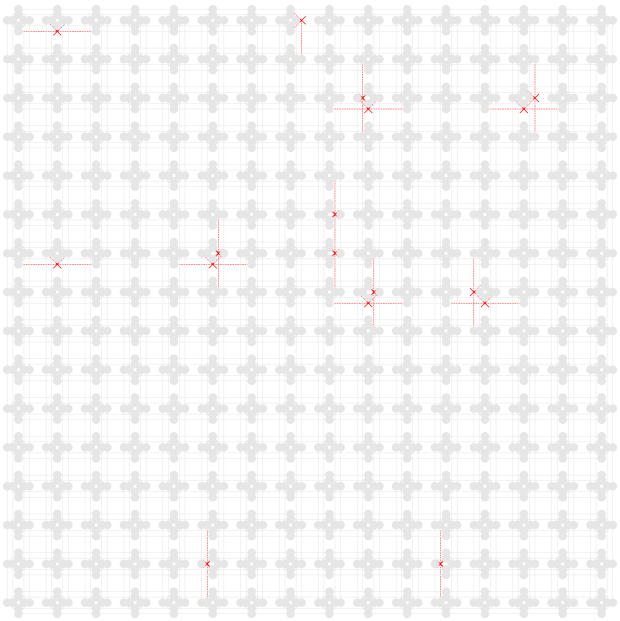
Figure 10: Plots of the median neighbour count per node (Y-axis) by guest graph type and node count (X-axis). The bands represent a 95% confidence interval for the spread of the median.

Appendix

B. Faults of Chimera Hosts used for Benchmarking



(a) D-Wave 2000Q-6



(b) D-Wave 2000Q-QuAIL

Figure 11: The above host topologies were used for benchmarking. The faulty nodes and edges of each respective topology are highlighted in red