Embedding Algorithms for Quantum Annealers with Chimera and Pegasus Connection Topologies*

Stefanie Zbinden[1,2], Andreas Bärtschi[2],
Hristo Djidjev[2], and Stephan Eidenbenz[2]

[1] Department of Mathematics, ETH Zürich, 8092 Zürich, Switzerland
[2] CCS-3, Los Alamos National Laboratory, Los Alamos, NM 87545, USA
zbindens@student.ethz.ch, baertschi@lanl.gov

Abstract. We propose two new algorithms – Spring-Based MinorMiner (SPMM) and Clique-Based MinorMiner (CLMM) – which take as input the connectivity graph of a Quadratic Unconstrained Binary Optimization (QUBO) problem and produce as output an embedding of the input graph on a host graph that models the topology of a quantum computing device. As host graphs, we take the Chimera graph and the Pegasus graph, which are the topology graphs of D-Wave's 2000 qubit (first introduced in 2017) and 5000 qubit (expected 2020) quantum annealer devices, respectively. We evaluate our algorithms on a large set of random graph QUBO inputs (Erdős-Rényi $G_{n,p}$, Barabási-Albert and d-regular graphs) on both host topologies against other embedding algorithms. For the Pegasus topology, we find that CLMM outperforms all other algorithms at edge densities larger than 0.08, while SPMM wins at edge densities smaller than 0.08 for Erdős-Rényi graphs, with very similar transition densities for the other graph classes. Surprisingly, the standard D-Wave MinorMiner embedding algorithm – while also getting slightly outperformed by SPMM for sparse and very dense graphs on Chimera – does not manage to extend its overall good performance on Chimera to Pegasus as it fails to embed even medium-density graphs on 175–180 nodes which are known to have clique embeddings on Pegasus.

1 Introduction

Quantum annealers such as the D-Wave 2000Q offer high quality solutions to hard optimization problems, and have a relatively large number of (currently up to 2000) qubits, while the next-generation D-Wave Advantage (due in 2020) will have more than 5000 qubits. Because of the technological challenges in connecting qubits, existing qubit connectivity topologies are far from the desirable all-to-all topology, as a result limiting the sizes of the problems that can be solved

* Research presented in this article was supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory under project numbers 20180267ER / 20190065DR. Los Alamos report number LA-UR-20-22259.

Embedding Method	**QUBO Class**	**Host Graph**
MinorMiner (MM) [9,13]	Erdős-Rényi $G_{n,p}$	Chimera C16
Layout-Aware MinorMiner (LAMM) [29,28]	Barabási-Albert	Pegasus P16
Spring-based MinorMiner (SPMM)	random d-regular	
Clique-based MinorMiner (CLMM)		

Table 1. Study Parameters: We compare the performance of four embedding methods for three different QUBO graphs on the two main D-Wave host graph topologies.

on these devices. In fact, the currently used *Chimera* has degree 6 [7], while the *Pegasus* topology (available in 2020 with D-Wave Advantage) has degree 15 [5]. The programming model for the D-Wave quantum annealer consists of setting the coefficients of a quadratic optimization function on binary variables (called a Quadratic Unconstrained Binary Optimization (QUBO) problem) so that linear terms map to qubits and quadratic terms map to couplers between the corresponding qubits. In practical applications, we are given an input QUBO whose set of linear and quadratic weights does not directly map onto the physical topology of the D-Wave device, so we have to represent each variable by a set of qubits (chain) and decide how to map variables onto chains. This problem is usually modeled as a graph theoretic problem: Finding a *minor embedding* of the input QUBO graph into an input topology host graph, a classical algorithmic problem that is generally NP-hard [25]. The ability to embed practical QUBOs at larger and larger size directly correlates to the success and operational applicability of D-Wave devices when competing with classical devices.

In this paper, we propose and test two new embedding algorithms – Spring-based MinorMiner (SPMM) and Clique-based MinorMiner (CLMM). We study the performance of these algorithms as compared to two previously proposed methods: MinorMiner (MM) [9,13] and a recent adaptation, Layout-Aware MinorMiner (LAMM) [29,28]. All four algorithms are benchmarked on a large set of random input QUBO graphs that need to be embedded onto the Chimera and Pegasus topologies. As random graph classes, we study Erdős-Rényi $G_{n,p}$ graphs, Barabási-Albert graphs, and random d-regular graphs. Each of these graph classes has a density parameter and a graph order (size) that we vary in our experiments. We assess the performance of the four algorithms based on whether they are able to embed graphs. The parameters of our experimental study are given in Table 1. Our main findings are:

– On the Pegasus host graph, our Clique-based MinorMiner (CLMM) is a clear winner with our alternative Spring-Based MinorMiner (SPMM) algorithm edging out both CLMM and MM for very sparse graphs only. The relative ranking of the algorithms is the same across all three QUBO input classes with SPMM's advantage at sparse graphs most pronounced for d-regular graphs. Somewhat surprisingly, a threshold edge density exists that is very similar for all three random graph classes (at about $|E|/\binom{|V|}{2} \approx 0.08$) such

that CLMM and SPMM win at edge densities larger and smaller than the threshold, respectively (E, V denote edges and nodes of the QUBO graph).
– On the Chimera host graph, SPMM wins over MM and LAMM at sparse and dense graphs, whereas MM and LAMM perform slightly better at medium density graphs. Again, SPMM's advantage at large sparse graphs is most pronounced for d-regular graphs.
– On the Chimera host graph, all algorithms easily manage to embed the previously largest known embeddable clique (at 65 vertices), whereas on Pegasus only CLMM finds embeddings of cliques with more than 180 nodes. In fact, using SPMM for Chimera and CLMM for Pegasus we find largest embeddable cliques at sizes 65 and 185 respectively.

The paper is organized as follows: We introduce the concepts of QUBOs, embeddings, host graphs and other background material including related work in more detail in Section 2. We describe the embedding algorithms in Section 3, and give details about the experimental design in Section 4. We present our results for the Pegasus host graph in Section 5 and for the Chimera host graph in Section 6, before concluding in Section 7.

2 Background

2.1 Quadratic Unconstrained Binary Optimization (QUBO)

Quadratic Unconstrained Binary Optimization (QUBO) is the problem of minimizing a quadratic function of binary variables, in one of the forms

$$\min_x \quad \sum_{i=1}^{n} a_i x_i + \sum_{i<j} b_{ij} x_i x_j, \qquad x_i \in \{0,1\} \quad \text{(QUBO formulation)},$$

$$\text{or } \min_z \quad \sum_{i=1}^{n} h_i z_i + \sum_{i<j} J_{ij} z_i z_j, \quad z_i \in \{-1,+1\} \quad \text{(Ising formulation)}.$$

The two formulations are equivalent via bijective relations $h_i = \frac{1}{2}\left(a_i + \sum_j b_{ij}\right)$, $J_{ij} = \frac{b_{ij}}{4}$. Note that J_{ij} is nonzero if and only if b_{ij} is nonzero. Hence QUBO problems are naturally represented by a graph $P = (V_P, E_P)$, where in V_P each variable z_i is represented as a node z_i with weight h_i, and in E_P we have for every pair $i < j$ with nonzero J_{ij} an edge $e = \{z_i, z_j\}$ with edge weight J_{ij}.

We remark that QUBOs are a class of NP-hard optimization problems; as we can use QUBOs to optimize the number of satisfied constraints in an instance of 0/1 Integer Programming – one of Karp's original 21 NP-complete problems [21].

2.2 Solving QUBOs on Quantum Annealers

Quantum Annealers such as D-Wave's 2000Q and the upcoming D-Wave Advantage [5] have quantum processors with a set of qubits $\mathcal{Q}$ and a set of couplers $\mathcal{C}$

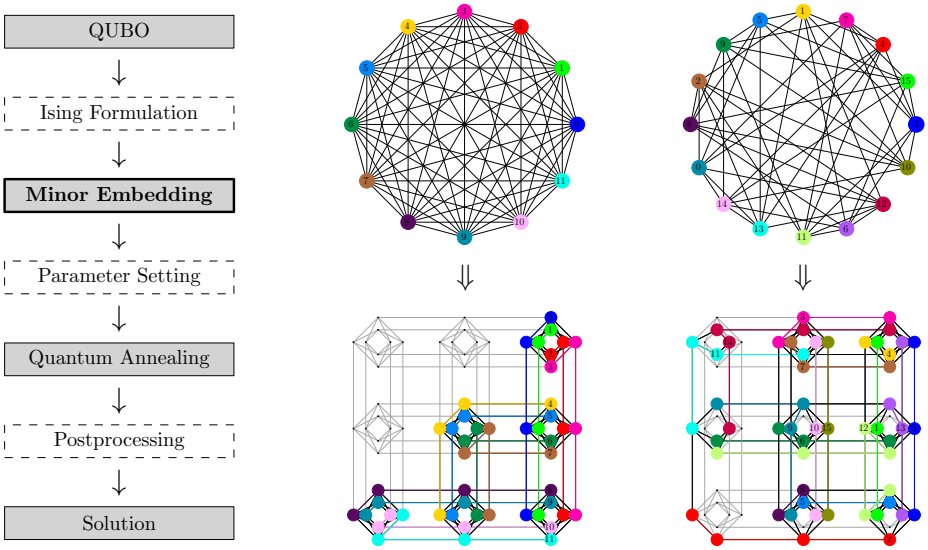

Fig. 1. Schematics of solving a QUBO instance with a Quantum Annealer (cf. [32]) **(left)** Full workflow **(center)** Clique minor embedding of a clique K_{12} on a Chimera graph $C3$ **(right)** Heuristic minor embedding of a 16-node 7-regular graph on host $C3$.

between some pairs of qubits. If we identify the qubits with a node set V_H and the couplers with an edge set E_H, the resulting connected structure is a graph $H = (V_H, E_H)$, called the host graph. The D-Wave programming model lets us set weights h_i for every qubit $q_i \in \mathcal{Q}$ and weights J_{ij} for every coupler $c_{ij} \in \mathcal{C}$. In an actual D-Wave calculation, the device uses quantum annealing to sample from low-energy eigenstates of the Hamiltonian

$$H = \sum_{i=1}^{n} h_i \sigma_z^{(i)} + \sum_{\{i,j\} \in E_H} J_{ij} \sigma_z^{(i)} \sigma_z^{(j)},$$

with Pauli-Z operators $\sigma_z^{(i)}$ acting on qubit q_i.[3] As such, the spin configuration of a groundstate corresponds to an optimum solution of a QUBO in Ising formulation with the same weights h_i, J_{ij}.

However, most users will have QUBO problems from their application domains with corresponding QUBO graphs that are far from being subgraphs of the host graph. In order to be able to solve QUBOs using a quantum annealer, the standard approach (see Figure 1) is to find a minor embedding of the QUBO graph into the host graph [11] and to set the h_i, J_{ij} parameters accordingly [10]; i.e. one chains multiple qubits of the host graph with ferromagnetic couplings $J_{ij} \ll 0$ to represent a single variable of a QUBO (indicated by shared colors in Figure 1 (center)/(right)). The better the embedding algorithm, the more

[3] We have $\sigma_z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$, $Id = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, and tensor product $\sigma_z^{(i)} = Id^{\otimes i-1} \otimes \sigma_z \otimes Id^{\otimes n-i}$.

QUBO problems can be solved by an annealer. Designing and testing capable embedding algorithms that are able to embed a large set of QUBO graphs is thus crucial to expand the set of applications for a quantum device such as D-Wave. The same holds true for CMOS annealers, such as those of Hitachi [36,33].

We note in passing that adiabatic quantum computing [15] – the theoretical inspiration for quantum annealer technology – is equivalent in power to standard gate-based quantum computing [2] that implements arbitrary unitary operations. However, the mapping challenge on gate-based quantum devices differs substantially from quantum annealers as logical variables are mapped only to single qubits and not to chains. To implement a gate between two non-neighboring qubits in a gate device, qubit states are swapped along paths of the host topology, giving a "time-dependent mapping", sometimes called routing. Depending on the application, this can be done heuristically [12], with exact solvers [35], or using a swap network [26]. Comparing state-of-the-art approaches to equality constraints implementation on a quantum annealer [34] and on a gate-based quantum computer [8] shows, on a concrete application, how different the mapping problem is for the two platforms.

2.3 Minor Embeddings

A *minor embedding* of a *pattern* graph $P = (V_P, E_P)$ into a *host* graph $H = (V_H, E_H)$ is a mapping φ of each node in V_P to a subset of nodes in V_H:

$$\varphi\colon V_P \to 2^{V_H},$$

where 2^{V_H} is the set of all subsets of V_H, such that

1. For each node v in V_P, the set of nodes $\varphi(v)$ induces a connected subgraph in H, called the *chain* of v.
2. For every edge $e = \{u, v\}$ in E_P, there exist nodes $\tilde{u} \in \varphi(u)$ and $\tilde{v} \in \varphi(v)$ such that $\{\tilde{u}, \tilde{v}\} \in E_H$.
3. $\varphi(v) \cap \varphi(u) = \emptyset$ for all $u \neq v \in V_P$, i.e., each node $\tilde{v}$ of the host graph H appears in the mapping of at most one node of the pattern graph P.

We call a mapping φ a *chain mapping* if it satisfies Condition 1. A chain mapping φ is called a *semi-valid embedding* if it satisfies Condition 2 and is called a *chain placement* if it satisfies Condition 3. Only if all three conditions are satisfied do we have a minor embedding. Colloquially, we abbreviate minor embedding with just embedding.

Finding a minor-embedding is NP-complete [25] except for (small) fixed pattern graphs [30], and the best known algorithms [1] are exponential in $|V_P|$ and the branch-width or tree-width of H (which is $\Omega(\sqrt{|V_H|})$ for current annealers). Research on minor-embedding for annealers has therefore focused on finding fast and hiqh-quality heuristics. Existing approaches can best be described along one of two trajectories: (i) iteratively modify a semi-valid embedding to reduce the number of multiply used nodes $\tilde{v} \in V_H$ (the approach shared by the algorithms benchmarked in this paper), (ii) iteratively modify a chain placement

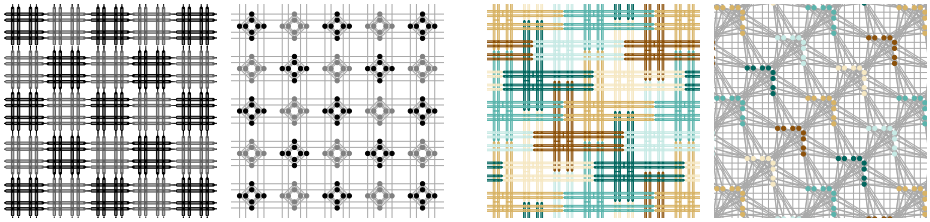

Fig. 2. (left) Chimera topology (D-Wave 2000Q): intersecting axis-parallel rectangles gives rise to a grid of $K_{4,4}$ tiles with vertical/horizontal connections. **(right)** Pegasus topology (D-Wave Advantage): non-bipartite graph & increased connectivity achieved through longer, shifted rectangles and couplers for pairs of neighboring parallel qubits. Rectangle drawings courtesy of Kelly Boothby (D-Wave Systems, Inc.).

to increase the number of represented edges $e \in E_P$ (recently proposed [32] for King's graphs, the topology of Hitachi CMOS annealers [33]).

Furthermore, good minor embeddings are known for highly structured pattern graphs such as cliques [24,6], cartesian products thereof [37], bicliques [19], cubic grids [20] and cylindrical lattices (square-octogonal and triangular) [23].

2.4 Chimera and Pegasus Topologies

The host graphs of current and upcoming D-Wave annealers can be understood starting from an intersection graph of axis-parallel rectangles (the qubits):

In Chimera [7], 4×4 intersecting orthogonal qubits with internal couplers give rise to biclique $K_{4,4}$ tiles. External couplers to adjacent horizontal respectively vertical qubits arrange these in a grid, where neighboring tiles are connected by 4 edges. All qubits (except those on the border) have degree 6, see Figure 2 (left). The Chimera graph $C16$, such as in the D-Wave 2000Q, has 16×16 tiles for a total of 2048 qubits. We illustrate a smaller $C3$ in Figure 1.

In Pegasus [5], qubit rectangles are longer and connect to 12 orthogonal qubits. Furthermore, horizontal and vertical qubits are shifted asymmetrically, and have additional odd couplers that connect pairs of neighboring parallel qubits, such that qubits have degree 15. This results in cells that are connected by 4, 8, or 16 edges, see Figure 2 (right). The Pegasus graph $P16$, such as in the upcoming D-Wave Advantage, has $15 \times 15 \times 3$ cells, plus some partial cells on the border, for a total of 5640 qubits. We illustrate $P4$ in Figure 3.

2.5 QUBO Random Graph Classes

To extend the range of embeddable QUBOs on current and next-generation devices, we benchmark embedding algorithms based on their performance in finding embeddings. Other metrics such as average or maximum chain lengths [29] are not a focus of this paper; hence the actual values of non-zero QUBO terms do not matter. Similarly, we only consider connected graphs (as one can always

solve connected components independently) and do not consider any divide-and-conquer strategies [27]. We use three classes of random graphs as benchmarks:

(i) Erdős–Rényi graphs $G_{n,p}$ [18], where edges are included in the graph i.i.d with probability p, (ii) Barabási–Albert graphs $BA_{n,m}$ [4,3], in which, starting from m isolated nodes, we insert $m-n$ nodes one by one, connecting each to m existing nodes with preferential attachment proportional to the current degree distribution, (iii) random d-regular graphs, in which each node has degree d. By varying p, d and m, respectively, we generate graphs of various densities.

We chose these three graph classes in order to test our algorithms on a diverse set of graphs: Erdős–Rényi graphs have a binomial (Poisson for small p) degree distribution, Barabási–Albert graphs have a power-law degree distribution (modeling networks), and d-regular graphs have a constant degree distribution.

In the following Section, we briefly present existing algorithms that we either compare to or use as a subroutine in our algorithms, which then follow next.

3 Minor Embedding Heuristics

3.1 Existing Embedding Algorithms

MinorMiner The MinorMiner algorithm (MM), proposed in 2014 [9], is arguably the most prominent embedding algorithm, given its inclusion in D-Wave's Ocean software stack [13]. Given any QUBO graph P and host graph H as an input, it tries to find an embedding; and if not successful after a certain number of steps it returns an empty embedding. The MM algorithm starts from an *initial chain mapping* (with chains empty by default) and *repeatedly loops* over the nodes of P, to determine for each node $v \in V_P$ a (preliminary) chain as follows:

1. Remove the chain $\varphi(v) \subseteq E_H$ from the existing chain mapping.
2. Compute a node-weighted shortest paths tree in H from each non-empty chain $\varphi(u)$, where u is a neighbor of v in P ($\{u, v\} \in E_P$). The node weights in H come with a high penalty term for using nodes in multiple chains.
3. Choose an optimal node $\tilde{v} \in V_H$ that minimizes the sum of distances according to the computed shortest paths trees. Extend $\tilde{v}$ to a chain $\varphi(v)$ by backtracking along the shortest paths trees, and re-add $\varphi(v)$ to the chain mapping.

This naturally splits MinorMiner into two phases: First, MM completes a single loop over the vertices V_P, after which the chain mapping φ is in fact a *semi-valid embedding* (in which chains might still share qubits).

Secondly, MM enters a *fixing phase*, where consecutive loops over nodes in V_P have the goal of fixing this semi-valid embedding. The algorithm restarts when there has been no progression for too many steps in a row[4], with limiting parameters on the total number of steps and number of restarts allowed. Thus,

[4] Even if the algorithm is already in a state with a valid embedding, progression is measured for example in having a smaller maximal chain size.

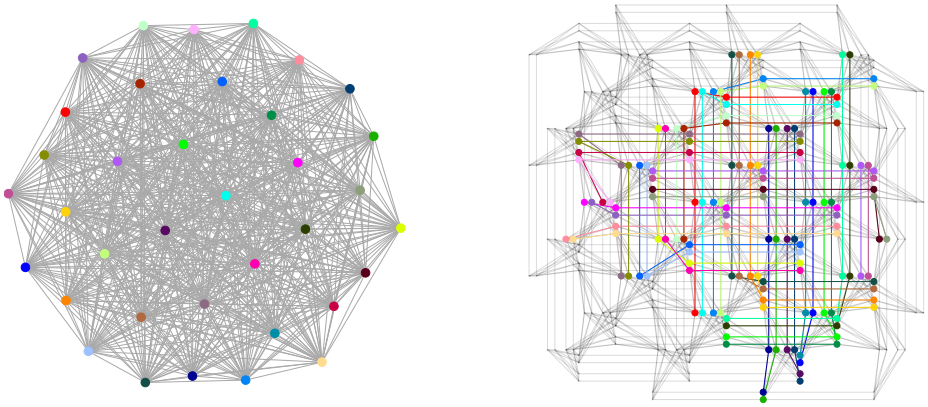

Fig. 3. Minor embedding of a K_{36} on Pegasus $P4$. Horizontal/vertical edges are mainly used to connect chains internally; other edges act as couplers between different chains.

when the algorithm terminates, it might either return a valid embedding when it found one, or an empty embedding if it did not.

MM has a few other controls, such as the `initial_chain` parameter. This parameter can be used to feed the algorithm an initial chain mapping, which is then used in the first phase of finding a semi-valid embedding. However, the algorithm still iterates over all nodes. When it reaches a node which was assigned an initial non-empty chain, it still deletes and replaces that chain with the procedure outlined above.

Layout-Aware MinorMiner A recent contribution to MinorMiner [29] has as its main focus QUBOs that come with a natural graph layout in the plane (think, e.g., of lattices in the simplest case). The implementation [28] takes a QUBO graph and its layout together with the host graph and a plane host graph layout as an input. The algorithm maps each variable node of the QUBO graph layout to the closest (in Euclidean metric) qubit node of the host graph layout. An additional diffusion phase shifts this mapping to achieve an even spreading of initial chains across tiles/cells of the topology, and then starts MinorMiner with the computed `initial_chain` mapping. However, not all QUBOs come with a natural layout; if the graph comes without a layout, their algorithm runs a Fruchterman-Reingold spring embedding algorithm to generate such a layout.

Clique Embedding D-Wave has a host-specific clique embedding algorithm [6], which can quickly embed any clique up to a certain size c_{host} into the Pegasus or the Chimera graph (this also implies an embedding algorithm for any graph with up to c_{host} nodes). For Pegasus $P16$, the maximal clique size embeddable this way is $c_{\text{host}} = 180$, for Chimera $C16$ it is $c_{\text{host}} = 64$. Chains gained from this

embedding have a very special shape: they are all paths which are "L-shaped" if drawn into the 2D-layout of the respective host graph, see Figure 3.

3.2 Our contribution

We propose, implement and compare two new algorithms: Clique-based MinorMiner (CLMM) and Spring-based MinorMiner (SPMM).

For CLMM, we construct an initial chain mapping for a subset of QUBO nodes, able to implement a coupling between any two chains of this node subset. For SPMM, we give an initial chain mapping for *all* QUBO nodes V_P, based on a force-directed graph drawing of P. In the second approach, there are no guarantees for existing couplings between chains. We then pass this initial chain mapping to MinorMiner with the `initial_chain` parameter.

Clique-based MinorMiner (CLMM) For CLMM, we construct an initial chain mapping as follows: We run D-Waves clique embedding algorithm for a clique of size $k = \min(|V_P|, c_{\text{host}})$. The k chains found this way are assigned to k nodes of the QUBO graph, with the assignment depending on the density of P: If $|E_P|/\binom{|V_P|}{2} \geq 0.55$, they are assigned to the k nodes of lowest degree, otherwise to k random nodes. The remaining QUBO nodes are mapped to empty chains.

We also tested a wide variety of other density- and degree-based assignments, as well as a splitting or a multi-assignment of chains in exploratory runs. In contrast to these approaches, the presented (albeit simpler) settings performed significantly better and were thus used in the final experiments.

Spring-based MinorMiner (SPMM) For SPMM, we construct an initial chain mapping as follows: (i) We use standard D-Wave layout functions to get a drawing of the Pegasus/Chimera host graph in the plane (cf. the host graphs in Figure 2), and a tuned Fruchterman-Reingold algorithm (see below) to get a QUBO graph layout as well. (ii) We rescale both plane layouts to fit into a $[-1, 1] \times [-1, 1]$ square. (iii) We map each of the QUBO nodes v to the closest qubit node in Euclidean metric.

Fruchterman-Reingold [16] is a force-directed graph drawing algorithm that computes a plane layout based on two principles: nodes pairwise repel each other, but nodes connected by an edge at the same time attract each other. The strength with which the latter takes place can be set for each edge individually; smaller weights implying a smaller attraction. For an edge $e = \{u, v\}$ we set $\text{weight}(e) = (2|E_P|/|V_P|)^2 \cdot (\deg(u) \deg(v))^{-1}$, where the weighting by node degrees ensures that neighboring nodes with high degrees are not too close to each other (as, intuitively, their chains need more space in the host graph) and where the first term is a normalization factor (normalizing weights in regular graphs to 1).

While SPMM and LAMM have some similarities, we find a significant performance difference on Pegasus graphs due to SPMM's improved use of edge weights for node attraction, substituting LAMM's consecutive diffusion phase.

4 Experimental Design

We present the results of a large factorial-design experiment to compare our two algorithms Clique-based MinorMiner (CLMM) and Spring-Based MinorMiner (SPMM) with the established MinorMiner (MM) and the recently proposed Layout-Aware MinorMiner (LAMM). We test the algorithms on the random QUBO graph classes $G_{n,p}$, Barabási-Albert, and d-regular. As host graphs, we use the D-Wave Pegasus host graph (used in the 5000 qubit model first out in 2020) as well as the previous Chimera topolgy (used until the 2000 qubit model).

For the Erdős–Rényi $G_{n,p}$ graph model, we generate five random graphs for each combination of values $n = \{1, \ldots, 425\}$ and $p = \{.01, .02, \ldots, 1.00\}$. While this would result in a total of $5 \cdot 450 \cdot 100 = 212,500$ graphs, we actually reduced this number to around $26,000$ graphs by carefully pruning the set of graphs for a specific algorithm once it has become clear – based on results for smaller/larger values of n or p – that the algorithm will always/never find an embedding. $G_{n,p}$ graphs have a sharp threshold of $n \cdot p > \ln n$ of being connected [14].

For the d-regular graph model (on Pegasus), we generate five graphs each for all combinations of $n = \{1, \ldots, 1200\}$ and $d = \{3, \ldots, 183\}$, employing again a pruning mechanism. On Chimera, we also use five graphs and cut off at 380 vertices and maximum $d = 64$ to account for the smaller host graph. Random d-regular graphs can be sampled quickly for $d \leq n/2$ [31] and uniformly at random for $d \in \mathcal{O}(n^{1/3-\varepsilon})$ [22]; we sample $(n-d)$-regular graphs as complements of d-regular graphs. d-regular graphs only exist for $2|E| = n \cdot d$ even and $d < n$.

For the Barabási-Albert graph model, we generate five graphs each for $n = \{1, \ldots, 1200\}$ and $m = \{1, \ldots, 110\}$ and employ pruning. The number of edges in $BA_{n,m}$ is $(n-m) \cdot m \leq (\frac{n-m+m}{2})^2 = n^2/4$ by AM-GM, with equality for $m = n/2$. Hence we get increasing graph density for m up to $n/2$, and we restrict ourselves to this regime. All graphs are constructed with Python's `networkx`.

Our experiments were executed on LANL's Darwin Cluster [17] using a trivially parallel approach. Running times for individual graphs ranged from milliseconds to more than 10 minutes per graph, largely proportional to graph vertex and edge counts. Overall, the study consumed around $100,000$ core hours. We assess the different algorithms on whether they succeed in finding an embedding with the default parameters of MinorMiner, and not by running times, but overall we observed that running times were very comparable for all the tested algorithms.

5 Embeddings on the Pegasus Host Graph

Embedding Erdős–Rényi Graphs on Pegasus. Figure 4 shows our results for $G_{n,p}$ graphs on Pegasus for the four algorithms MM, LAMM, CLMM, and SPMM. The plot structure is as follows: The blue area on the bottom shows where $n \cdot p < \ln(n)$, the region of disconnected QUBOs excluded from this study. The red vertical line displays $c_{\text{host}} = 180$. Heatplot areas are colored using the green-to-white color scheme on the right of the plot. The color assigned to

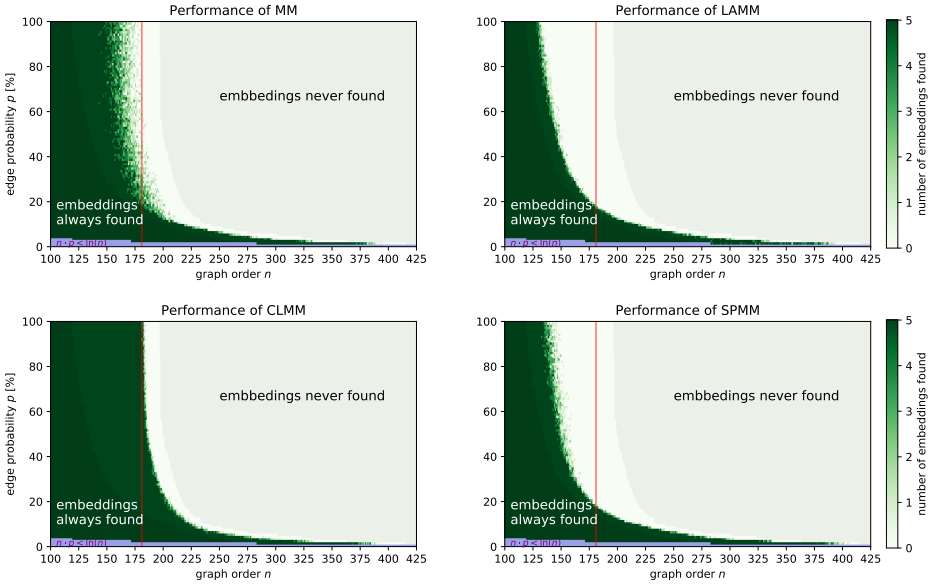

Fig. 4. Embedding performance of all algorithms for Erdős–Rényi graphs on Pegasus: **(top left)** MM, **(top right)** LAMM, **(bottom left)** CLMM, **(bottom right)** SPMM.

a point (n, p) corresponds to the number of times the algorithm succeeds at finding an embedding for the five $G_{n,p}$ graphs tested at point (n, p). The large darker-green area on the left are pruned points, as we can be reasonably sure that the algorithm would always find an embedding since it does find embeddings reliably for larger graphs. Similarly, the light gray area on the right side of the plot represents pruned points, where we are reasonably sure that the algorithm would not find an embedding as it did not find embeddings on smaller and less dense graphs. More precisely, if an algorithm manages to embed a $G_{n,p}$ QUBO with high probability, it is even more likely that it will manage to embed a $G_{n-k,p}$ QUBO graph. Therefore, after testing for each p where the transition from embeddable QUBO to non embeddable QUBO is, we tested a cone of width at least 10 on both sides around them as interesting points before pruning.

Contrasting the performance of the four algorithms, we note the following: The LAMM algorithm does not perform particularly well, perhaps unsurprisingly as $G_{n,p}$ graphs do not have a natural layout that would play to LAMM's core design element; LAMM does show a fairly quick transition from being able to embed all graphs (dark green) to no graphs (white). This transition is in fact more spread-out in the overall better performing SPMM algorithm. The standard MM algorithm sees an even farther spread-out transition when compared to both LAMM and SPMM and clearly outperforms LAMM and SPMM on dense graphs, while being outperformed by SPMM on very sparse graphs. However, MM is remarkably far off from being able to embed a clique of size 180 (the red

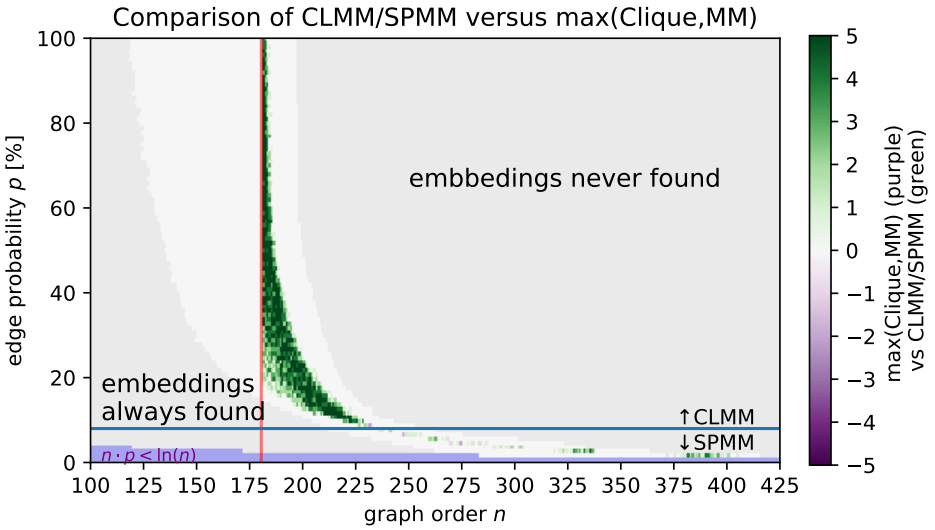

Fig. 5. A combination of CLMM/SPMM outperforms existing methods (host-specific clique and heuristic MinorMiner embeddings) on embedding Erdős–Rényi graphs into Pegasus at every value of p, with a sharp transition from CLMM to SPMM at $p = 0.08$.

vertical line). CLMM easily outperforms MM on dense graphs and interestingly shows a very cleanly defined transition from embeddable to non-embeddable.

We get a more in-depth understanding of performance difference by looking at the difference plot in Figure 5. Its structure is similar to the individual performance plots, except the color bar ranges from green (positive) to white (zero) to purple (negative). A point (n, p) above (below) the blue line at $p = 0.08$ is assigned a color based on the number of embeddings found by CLMM (SPMM, respectively) *minus* the maximum of the number of embeddings found by the clique embedding algorithm or by MM. This way we capture the improvement SPMM gains for sparse graphs and the improvement CLMM gains on dense graphs in one plot. The transition between areas where CLMM and where SPMM are the respective best performing algorithms is sharp, around an edge density value of $|E_P|/\binom{|V_P|}{2} \approxeq p = 0.8$. In combination, our algorithms manage to outperform the already existing algorithms at every value of p, gaining the most around $p = 0.20$, and for $p = 0.02$ where the graphs get sparse enough such that SPMM's advantage over MM starts to get significant.

Embedding Barabási–Albert and d-regular Graphs on Pegasus. Figure 6 (top) shows a similar picture as Figure 5, with CLMM outperforming MM on dense graphs and SPMM taking the lead on sparse graphs. However, as Barabási–Albert graphs for small m are sparser than the sparsest Erdős–Rényi graphs we tested, the improvement of SPMM over MM is much more pronounced, being largest for $m = 2$. We again observe a sharp transition threshold between

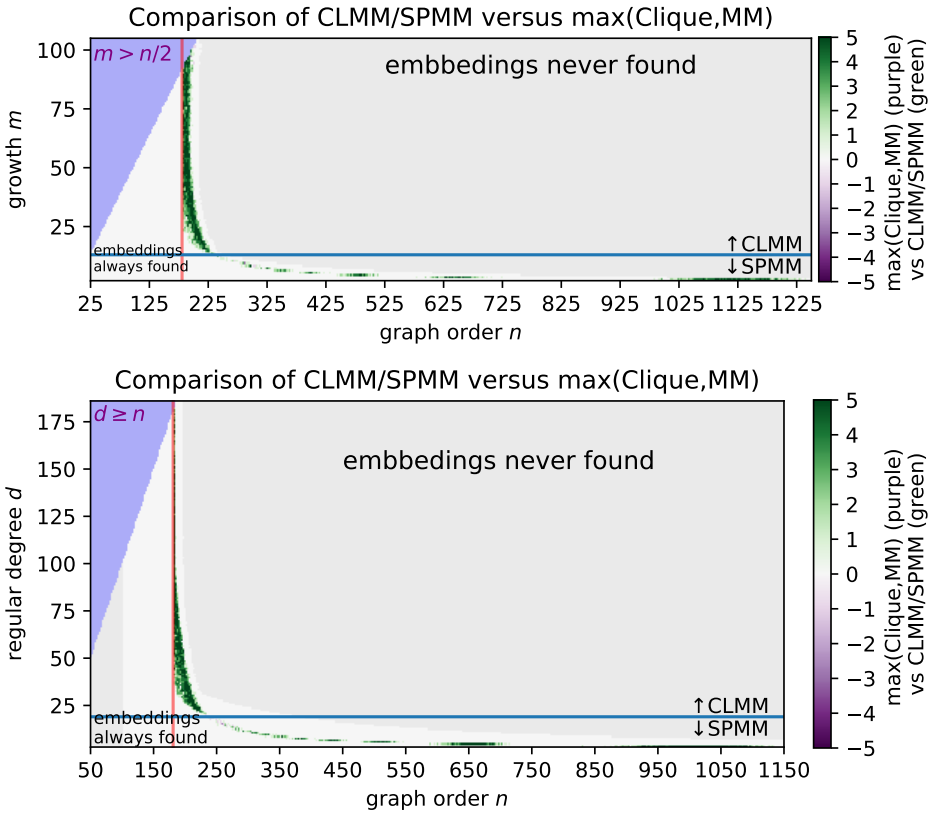

Fig. 6. (top) Performance comparison of CLMM/SPMM vs max(Clique, MM) for Barabási–Albert graphs on Pegasus, transitioning from CLMM to SPMM at $m = 12$. **(bottom)** Performance comparison of CLMM/SPMM vs max(Clique, MM) in embedding d-regular graphs into Pegasus, transitioning from CLMM to SPMM at $d = 18$. The plot omits odd columns to prevent distraction by empty data points for $n \cdot d$ odd.

CLMM and SPMM at $m = 12$ around $n = 240$, corresponding to an edge density of $(n - m)m/\binom{n}{2} \approx 0.095$.

Figure 6 (bottom) shows that on d-regular graphs, performance of CLMM, SPMM and MM mirrors their performance on Erdős–Rényi and Barabási–Albert graphs. Since d-regular graphs only exist for even $n \cdot d$, we omit odd n columns from the plot (but not from the experiments, see the concluding data in Section 7). SPMM again gains the biggest advantage on the sparsest graphs, namely $d = 3$, while CLMM outperforms MM on dense graphs, with a transition threshold at $d = 18$, $n = 233$, corresponding to an edge density of $d/(n - 1) \approx 0.078$.

Discussion. We first discuss MM's poor performance on Pegasus, where the picture is quite bleak: Here, graphs have to be very sparse until MM manages to embed a graph of order 180 nodes, even though there exists a host-specific

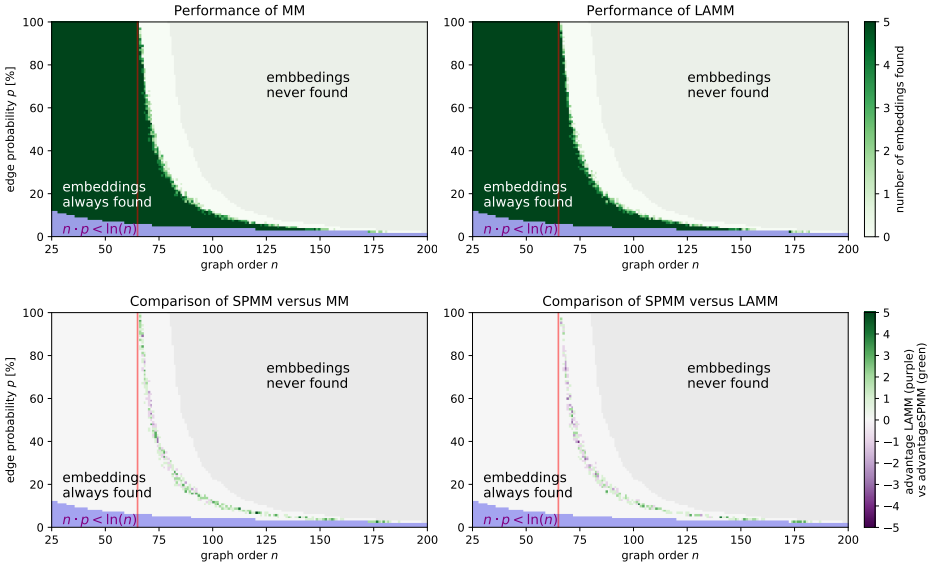

Fig. 7. Performance in embedding Erdős–Rényi graphs on Chimera **(top left)** for MM and **(top right)** for LAMM. **(bottom)** Respective improvements made by SPMM.

embeddable clique of size 180.[5] In trying to find out why MM fails on instances which are still easy embeddable via a host-specific clique embedding, we look at the characteristic pattern given by such a clique embedding. Recall that each QUBO node is mapped to a chain, where the qubit nodes in the chain form a path, linked mostly by edges that are horizontal or vertical in the graph (see Figure 3). Looking at the layout of Pegasus, these are both the sparsest connections between neighboring cells as well as the edges which have the longest length. Therefore, the chains are able to "spread through the graph" using as few qubits as possible, leaving many unused edges suitable as couplers between different chains. However, MM does not distinguish between different types of cell-connecting edges when re-computing a chain of the chain mapping, possibly resulting in a worse solution at the end. In contrast, the edges between tiles of Chimera are all equivalent, so this kind of misstep cannot happen.

Secondly, we look at the link between CLMM and SPMM's performance and the sparsity of the graph. In embeddings for dense graphs, chains often form a path through a large part of the host graph, with few or no nodes of induced degree larger than two. We believe that providing initial "L-shaped" chains such as in CLMM may promote newly built chains to take on such shapes as well. On the other hand, for sparse graphs a well-chosen initial single-qubit chain such as in SPMM can enable short connections to neighboring chains, reducing the qubit footprint of a semi-valid embedding created after the first phase of MinorMiner.

[5] Especially compared to Chimera, where MM manages to find an embedding for K_{65}, the largest embeddable clique, given that treewidth$(K_{65}) = 64 =$ treewidth$(C16)$.

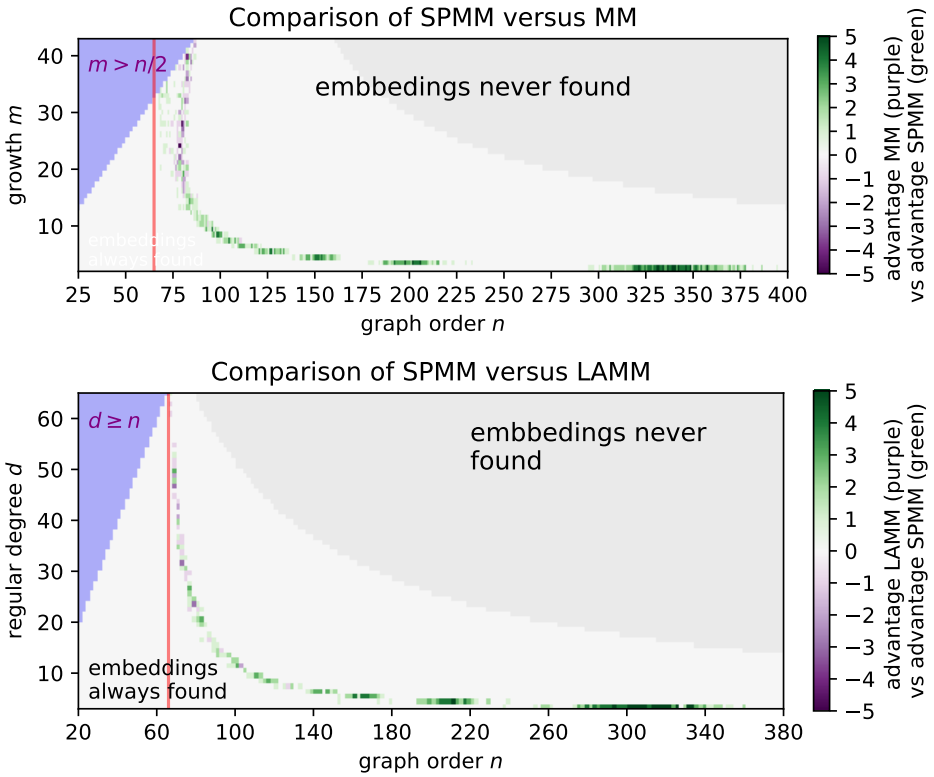

Fig. 8. Embedding performance of SPMM compared to its closest (QUBO graph type specific) competitor on Chimera: **(top)** SPMM vs MM for Barabási–Albert graphs, **(bottom)** SPMM vs LAMM for d-regular graphs, with odd n columns omitted.

6 Embeddings on the Chimera Host Graph

On Chimera, we only compare the three algorithms MM, LAMM and SPMM. We did not test CLMM in great detail, as MM performs very similar, and since preliminary observations could not find any improvements of CLMM over MM.

Embedding Erdős–Rényi Graphs on Chimera. For each non-pruned parameter combination (n, p), we generated five $G_{n,p}$ graphs which we tried to embed using MM, LAMM and SPMM. Figure 7 shows the performance of both MM (left) and LAMM (right) as well as the relative improvements made by SPMM (bottom). Perhaps a bit surprisingly, all algorithms manage to embed cliques of size 65, the largest embeddable clique and one node larger than the maximal clique found by the host-graph specific clique embedder.

SPMM performs better than MM on graphs with $p \geq 0.8$ and graphs with $p \leq 0.3$. However, for $0.3 < p < 0.8$, both algorithms perform comparably well. The performance difference between SPMM and LAMM is similar to the one

	Ranking	Erdős–Rényi	Barabási–Albert	d-regular
Pegasus (dense)	1. **CLMM**	**86,159**	**38,256**	**54,937**
$0.08 < p \leq 1.00$	2. MM	78,230	31,206	44,229
$12 < m \leq n/2$	3. SPMM	70,512	24,435	35,375
$18 < d \leq n-1$	4. LAMM	68,349	23,248	32,998
	Clique	81,530	30,420	49,410
Pegasus (sparse)	1. **SPMM**	**6,150**	**23,334**	**37,210**
$0.01 \leq p \leq 0.08$	2. MM	6,039	21,814	35,866
$2 \leq m \leq 12$	3. CLMM	5,964	21,803	35,490
$3 \leq d \leq 18$	4. LAMM	6,047	18,681	35,374
	Clique	2,700	10,985	12,470
Chimera	1. **SPMM**	**33,793**	**10,874**	**16,506**
	2. LAMM	33,688	10,217	16,132
	3. MM	33,530	10,367	15,972
	Clique	27,860	5,445	8,060

Table 2. Summary of all experiments: We rank Algorithms based on the total number of found embeddings. Pegasus experiments are split into a sparse and a dense QUBO graph regime, given by the observed transition parameters for p, m, d. For comparison, we also give the number of possible embeddings via host-specific cliques.

between SPMM and MM. However, while SPMM still beats LAMM for $p \leq 0.2$, for larger p LAMM outperforms SPMM slightly.

Embedding Barabási–Albert and d-regular Graphs on Chimera. While SPMM delivers the best overall performance in embedding both Barabási–Albert and d-regular graphs on Chimera, the second place depends on the graph class (MM for Barabási–Albert, LAMM for d-regular graphs). In Figure 8 (top), we show the performance difference between SPMM and MM on Barabási–Albert graphs. While MM outperforms SPMM slightly on $m \geq 20$, the advantage of SPMM on small m is much more apparent, especially at $m = 2$.

Figure 8 (bottom) shows the difference between embedding performances of SPMM and LAMM on d-regular graphs. For $15 \leq d \leq 64$, both algorithms perform comparably well, with a slight advantage to LAMM. For smaller degree, SPMM starts to clearly outperform LAMM (and MM), with the most significant improvement at $d = 3$. Again, we omit odd n columns in the plot.

7 Conclusion

We studied the performance of two new embedding algorithms, Spring-based MinorMiner (SPMM) and Clique-based MinorMiner (CLMM), and contrasted these to existing embedding heuristics for the two different D-Wave host graph

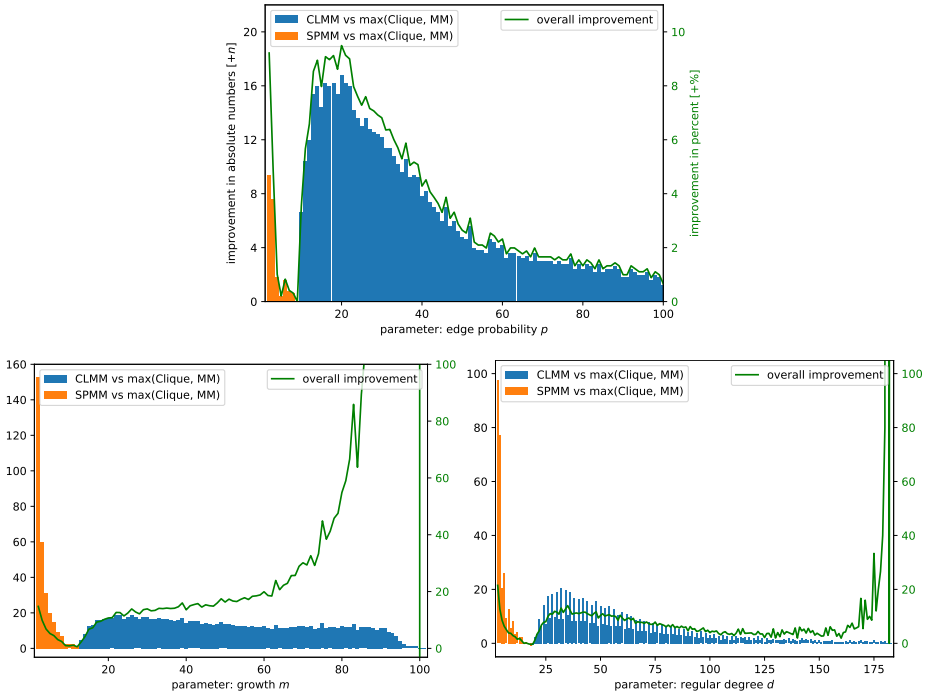

Fig. 9. Improvement of our two algorithms SPMM (orange) and CLMM (blue) compared to the maximal possible embeddability range with a host-specific clique or a heuristic MM embedding algorithm on Pegasus. Results for **(top)** Erdős–Rényi graphs, **(bottom left)** Barabási–Albert graphs, **(bottom right)** random d-regular graphs.

topologies Pegasus and Chimera. To the best of our knowledge, this is the first such study on the upcoming Pegasus topology. While we observed that the existing MinorMiner heuristic does not extend its overall good performance on Chimera to Pegasus, we show how to remedy the situation with our Clique-based and Spring-based MinorMiner variants, see Table 2.

We found that for certain values of the density parameters p, m, d (used in Erdős–Rényi, Barabási–Albert and d-regular graphs, respectively) our algorithms significantly outperform the existing methods, increasing the number of embeddable QUBO graphs by double-digit percentages and enlarging the range of embeddable sparse graphs to graphs with over a hundred additional nodes. Detailed statistics are given in Figure 9, where for each studied value of p, m and d, we show the number of additionally embeddable graphs, both in absolute numbers (bar plots) as well as a percentage increase (line plot). We note that absolute numbers are normalized by the number of sampled graphs per data point (i.e. 5), and that for d-regular graphs, the bar plots show the expected factor 2 difference between odd and even values of d (with the exception of 3-regular graphs, on which SPMM shows an exceptionally massive increase).

In conclusion, we studied different random graphs to represent a wide variety of possible QUBO graphs and gave a detailed analysis of the performance of CLMM, SPMM, MM and the recent LAMM. A relative ranking of the algorithms based on the total number of found embeddings is given in Table 2. While SPMM and CLMM are the clear winners in their respective density domains, the order of the competitors can change depending on the graph class studied.

Though SPMM and CLMM outperform the standard algorithm MM, their simplicity is somewhat remarkable and of course they build upon the work of both the original MinorMiner paper [9] and its implementation [13] as a subroutine. We suggest that the MinorMiner parameter `initial_chain` be extended with '`clique`' and '`spring`' parameters to serve as calls to the respective CLMM and SPMM algorithms presented in this work.

Future Work. Future research directions are three-fold: First, we intend to add case studies of real-world QUBO instance graphs to include them in a full version of this paper, together with plots and results of all our experiments.

Secondly, we would like to study other (CMOS) host graphs [36] and compare our algorithms to simulated annealing-based approaches which were recently proposed in the literature [32] but not yet published as software.

Finally, applying various embedding algorithms to the same QUBO problem will result in embeddings with different characteristics, such as the distribution of chain lengths. These characteristics, in turn, will influence the chance of success and hence the overall time-to-solution of solving QUBO problems with a quantum annealer. Once the Pegasus architecture becomes available, it will be useful to compare embedding algorithms with respect to these metrics, as was done for Chimera before [29].

References

1. Adler, I., Dorn, F., Fomin, F.V., Sau, I., Thilikos, D.M.: Faster parameterized algorithms for minor containment. Theoretical Computer Science **412**(50), 7018–7028 (2011). https://doi.org/10.1016/j.tcs.2011.09.015
2. Aharonov, D., van Dam, W., Kempe, J., Landau, Z., Lloyd, S., Regev, O.: Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation. SIAM Journal on Computing **37**(1), 166–194 (2007). https://doi.org/10.1137/S0097539705447323
3. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Reviews of Modern Physics **74**(1), 47–97 (2002). https://doi.org/10.1103/RevModPhys.74.47
4. Barabási, A.L., Albert, R.: Emergence of Scaling in Random Networks. Science **286**(5439), 509–512 (1999). https://doi.org/10.1126/science.286.5439.509
5. Boothby, K., Bunyk, P., Raymond, J., Roy, A.: Next-Generation Topology of D-Wave Quantum Processors. Tech. Rep. 14-1026A-C, D-Wave Systems (2019), https://www.dwavesys.com/resources/publications?type=white
6. Boothby, T., King, A.D., Roy, A.: Fast clique minor generation in chimera qubit connectivity graphs. Quantum Information Processing **15**(1), 495–508 (2016). https://doi.org/10.1007/s11128-015-1150-6

7. Bunyk, P.I., Hoskinson, E.M., Johnson, M.W., Tolkacheva, E., Altomare, F., Berkley, A.J., Harris, R., Hilton, J.P., Lanting, T., Przybysz, A.J., Whittaker, J.: Architectural Considerations in the Design of a Superconducting Quantum Annealing Processor. IEEE Transactions on Applied Superconductivity **24**(4), 1–10 (2014). https://doi.org/10.1109/TASC.2014.2318294
8. Bärtschi, A., Eidenbenz, S.: Deterministic Preparation of Dicke States. In: Fundamentals of Computation Theory. pp. 126–139. FCT'19 (2019). https://doi.org/10.1007/978-3-030-25027-0_9
9. Cai, J., Macready, W.G., Roy, A.: A practical heuristic for finding graph minors. https://arxiv.org/abs/1406.2741 (2014)
10. Choi, V.: Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. Quantum Information Processing **7**(5), 193–209 (2008). https://doi.org/10.1007/s11128-008-0082-9
11. Choi, V.: Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design. Quantum Information Processing **10**(3), 343–353 (2011). https://doi.org/10.1007/s11128-010-0200-3
12. Cowtan, A., Dilkes, S., Duncan, R., Krajenbrink, A., Simmons, W., Sivarajah, S.: On the Qubit Routing Problem. In: 14th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC'19. pp. 5:1–5:32 (2019). https://doi.org/10.4230/LIPIcs.TQC.2019.5
13. D-Wave Systems: minorminer. https://github.com/dwavesystems/minorminer (2017), a heuristic tool for minor embedding.
14. Erdős, P., Rényi, A.: On random graphs I. Publicationes Mathematicae **6**, 290–297 (1959), https://www.renyi.hu/~p_erdos/1959-11.pdf
15. Farhi, E., Goldstone, J., Gutmann, S., Sipser, M.: Quantum Computation by Adiabatic Evolution. https://arxiv.org/abs/quant-ph/0001106 (2000)
16. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. Software: Practice and Experience **21**(11), 1129–1164 (1991). https://doi.org/10.1002/spe.4380211102
17. Garrett, C.K.: The Darwin Cluster. Tech. Rep. LA-UR-18-25080, Los Alamos National Laboratory (2018). https://doi.org/10.2172/1441285
18. Gilbert, E.N.: Random Graphs. Annals of Mathematical Statistics **30**(4), 1141–1144 (1959). https://doi.org/10.1214/aoms/1177706098
19. Goodrich, T.D., Sullivan, B.D., Humble, T.S.: Optimizing adiabatic quantum program compilation using a graph-theoretic framework. Quantum Information Processing **17**(5), 118 (2018). https://doi.org/10.1007/s11128-018-1863-4
20. Harris, R., Sato, Y., Berkley, A.J., Reis, M., Altomare, F., Amin, M.H., Boothby, K., Bunyk, P., Deng, C., Enderud, C., Huang, S., Hoskinson, E., Johnson, M.W., Ladizinsky, E., Ladizinsky, N., Lanting, T., Li, R., Medina, T., Molavi, R., Neufeld, R., Oh, T., Pavlov, I., Perminov, I., Poulin-Lamarre, G., Rich, C., Smirnov, A., Swenson, L., Tsai, N., Volkmann, M., Whittaker, J., Yao, J.: Phase transitions in a programmable quantum spin glass simulator. Science **361**(6398), 162–165 (2018). https://doi.org/10.1126/science.aat2025
21. Karp, R.M.: Reducibility among Combinatorial Problems, pp. 85–103. Springer US (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
22. Kim, J.H., Vu, V.H.: Generating Random Regular Graphs. In: 35th ACM Symposium on Theory of Computing. pp. 213–222. STOC'03 (2003). https://doi.org/10.1145/780542.780576
23. King, A.D., Carrasquilla, J., Raymond, J., Ozfidan, I., Andriyash, E., Berkley, A., Reis, M., Lanting, T., Harris, R., Altomare, F., Boothby, K., Bunyk, P.I., Enderud,

C., Fréchette, A., Hoskinson, E., Ladizinsky, N., Oh, T., Poulin-Lamarre, G., Rich, C., Sato, Y., Smirnov, A.Y., Swenson, L.J., Volkmann, M.H., Whittaker, J., Yao, J., Ladizinsky, E., Johnson, M.W., Hilton, J., Amin, M.H.: Observation of topological phenomena in a programmable lattice of 1,800 qubits. Nature **560**(7719), 456–460 (2018). https://doi.org/10.1038/s41586-018-0410-x

24. Klymko, C., Sullivan, B.D., Humble, T.S.: Adiabatic quantum programming: minor embedding with hard faults. Quantum Information Processing **13**(3), 709–729 (2014). https://doi.org/10.1007/s11128-013-0683-9
25. Matoušek, J., Thomas, R.: On the complexity of finding iso- and other morphisms for partial k-trees. Discrete Mathematics **108**(1), 343–364 (1992). https://doi.org/10.1016/0012-365X(92)90687-B
26. O'Gorman, B., Huggins, W.J., Rieffel, E.G., Whaley, K.B.: Generalized swap networks for near-term quantum computing. https://arxiv.org/abs/1905.05118 (2019)
27. Pelofske, E., Hahn, G., Djidjev, H.: Solving large minimum vertex cover problems on a quantum annealer. In: 16th ACM International Conference on Computing Frontiers. pp. 76–84. CF'19 (2019). https://doi.org/10.1145/3310273.3321562
28. Pinilla, J.P.: embera. https://github.com/joseppinilla/embera (2019), a collection of minor-embedding methods and utilities.
29. Pinilla, J.P., Wilton, S.J.E.: Layout-Aware Embedding for Quantum Annealing Processors. In: High Performance Computing. pp. 121–139. ISC'19 (2019). https://doi.org/10.1007/978-3-030-20656-7_7
30. Robertson, N., Seymour, P.: Graph Minors .XIII. The Disjoint Paths Problem. Journal of Combinatorial Theory, Series B **63**(1), 65–110 (1995). https://doi.org/10.1006/jctb.1995.1006
31. Steger, A., Wormald, N.C.: Generating Random Regular Graphs Quickly. Combinatorics, Probability and Computing **8**(4), 377–396 (1999). https://doi.org/10.1017/S0963548399003867
32. Sugie, Y., Yoshida, Y., Mertig, N., Takemoto, T., Teramoto, H., Nakamura, A., Takigawa, I., Minato, S.I., Yamaoka, M., Komatsuzaki, T.: Graph Minors from Simulated Annealing for Annealing Machines with Sparse Connectivity. In: Theory and Practice of Natural Computing. pp. 111–123. TPNC'18 (2018). https://doi.org/10.1007/978-3-030-04070-3_9
33. Takemoto, T., Hayashi, M., Yoshimura, C., Yamaoka, M.: A 2x30k-Spin Multi-chip Scalable Annealing Processor Based on a Processing-In-Memory Approach for Solving Large-Scale Combinatorial Optimization Problems. In: IEEE International Solid-State Circuits Conference. pp. 52–54. ISSCC'19 (2019). https://doi.org/10.1109/ISSCC.2019.8662517
34. Vyskocil, T., Djidjev, H.: Embedding Equality Constraints of Optimization Problems into a Quantum Annealer. Algorithms **12**(4), 77 (2019). https://doi.org/10.3390/a12040077
35. Wille, R., Burgholzer, L., Zulehner, A.: Mapping Quantum Circuits to IBM QX Architectures Using the Minimal Number of SWAP and H Operations. In: 56th Annual Design Automation Conference 2019, DAC'19. p. 142 (2019). https://doi.org/10.1145/3316781.3317859
36. Yamaoka, M., Yoshimura, C., Hayashi, M., Okuyama, T., Aoki, H., Mizuno, H.: A 20k-Spin Ising Chip to Solve Combinatorial Optimization Problems With CMOS Annealing. IEEE Journal of Solid-State Circuits **51**(1), 303–309 (2016). https://doi.org/10.1109/JSSC.2015.2498601
37. Zaribafiyan, A., Marchand, D.J.J., Changiz Rezaei, S.S.: Systematic and deterministic graph minor embedding for Cartesian products of graphs. Quantum Information Processing **16**(5), 136 (2017). https://doi.org/10.1007/s11128-017-1569-z