Quantum Information Processing (2021) 20:234
https://doi.org/10.1007/s11128-021-03168-z

Embedding of complete graphs in broken Chimera graphs

Elisabeth Lobe[1] · **Lukas Schürmann**[2] · **Tobias Stollenwerk**[3]

Received: 24 December 2020 / Accepted: 25 June 2021 / Published online: 10 July 2021

Abstract

In order to solve real-world combinatorial optimization problems with a D-Wave quantum annealer, it is necessary to embed the problem at hand into the D-Wave hardware graph, namely Chimera or Pegasus. Most hard real-world problems exhibit a strong connectivity. For the worst-case scenario of a complete graph, there exists an efficient solution for the embedding into the ideal Chimera graph. However, since real machines almost always have broken qubits, it is necessary to find an embedding into the broken hardware graph. We present a new approach to the problem of embedding complete graphs into broken Chimera graphs. This problem can be formulated as an optimization problem, more precisely as a matching problem with additional linear constraints. Although being NP-hard in general, it is fixed-parameter tractable in the number of inaccessible vertices in the Chimera graph. We tested our exact approach on various instances of broken hardware graphs, both related to real hardware and randomly generated. For fixed runtime, we were able to embed larger complete graphs compared to previous, heuristic approaches. As an extension, we developed a fast heuristic algorithm which enables us to solve even larger instances. We compared the performance of our heuristic and exact approaches.

Keywords Graph minor embedding · Chimera · Integer linear programming · Bipartite matching · Quantum annealing

✉ Elisabeth Lobe
elisabeth.lobe@dlr.de

[1] Institute for Software Technology, German Aerospace Center (DLR), Lilienthalplatz 7, 38108 Brunswick, Germany

[2] Institute of Computer Science, Rheinische Friedrich-Wilhelms-Universität Bonn, Regina-Pacis-Weg 3, 53113 Bonn, Germany

[3] Institute for Software Technology, German Aerospace Center (DLR), Linder Höhe, 51147 Cologne, Germany

1 Introduction

1.1 Background

Quantum annealing is a promising new technology which gained attention in recent years due to the development of commercial quantum annealing devices by the company D-Wave Systems. These machines sample from the low-energy distribution of a tunable system of interacting quantum bits (*qubits*) [6]. The energy of the system can be described by an *Ising model* including local energy fields for single qubits and certain pairwise interactions between the qubits. However, not every qubit interacts with all other qubits. The available couplings can be represented as edges in a graph where every qubit corresponds to a vertex. For currently operating D-Wave hardware, these graphs are the so-called *Chimera* and *Pegasus* graph [1].

In practice, no ideal Chimera or Pegasus graphs can be realized. Usually, there are some qubits or rarely couplings which are taken offline because they do not behave as expected after calibration. In the following, we refer to the corresponding vertices as *broken vertices* and to graphs containing them as *broken graphs*. Since calibrations are repeated in the order of months or years, a broken hardware graph is of practical relevance for the same amount of time.

Typical applications, however, need much more couplings than the hardware graphs provide [16,19,20,22]. Thus, before the user even can calculate on the annealing machines this problem needs to be mitigated by a so-called *embedding*: one vertex of the original graph, also referred to as *logical vertex*, is mapped to several qubits, also called *physical* vertices, of the hardware graph such that the induced subgraph is connected. For each edge in the original graph, there needs to exist at least one edge connecting the corresponding subgraphs of the two concerned logical vertices. See, e.g. [4] for more details. Each set of physical qubits representing a single logical vertex is grouped together by coupling them strongly.

In general, given two arbitrary graphs G and H, to decide whether G can be embedded into H is NP-hard. It is unclear but assumed that this still holds if we fix H to the broken Chimera graph. This means the embedding problem is as hard as the actual problem we want to solve on the D-Wave machine.

Due to the physical limitations, we cannot expect the production of a hardware architecture with a fully connected graph structure in the close future. Additionally, with the ongoing scaling of the annealing devices it is very unlikely that the problem of broken qubits will be eradicated completely. Nevertheless, we assume to see a decreasing ratio of broken qubits. All in all the embedding problem will remain relevant when dealing with quantum annealing devices in the long term.

For a few well-structured graphs, like the complete or the complete bipartite graph, the embedding problem is trivial if H is an ideal Chimera graph, due to its regular lattice structure. Although it is the worst-case scenario having a complete graph to be embedded, it allows the efficient embedding of all subgraphs of the complete graph. However, if there exist just a few inconveniently placed broken vertices, the scheme for the ideal Chimera cannot be applied. An important question of practical relevance is therefore: given a broken Chimera graph, what is the largest complete graph that

can be embedded? This is an optimization problem we refer to in the following as *largest complete graph embedding* (LCGE) problem.

If a graph is embeddable into another, it is a so-called *minor* of the second graph. Therefore, the LCGE problem is equivalent to the search for the *largest clique minor* [2], as the complete graph can be supplemented by the vertices which are not used for the embedding of the complete graph, forming a larger graph. The largest clique of this minor corresponds to the maximal embeddable complete graph.

In this work, we focus on the Chimera graph and leave the extension to the Pegasus graph, which has a larger connectivity for the same number of vertices [1], to future research. As Pegasus is derived from the Chimera we are confident that our results are transferable.

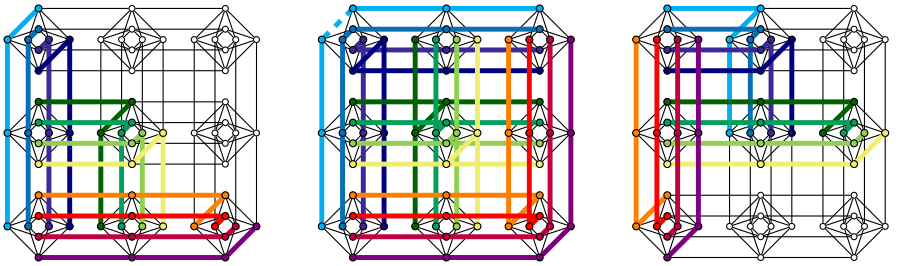
1.2 Related work

Graph minors have been a research topic of high interest even before the D-Wave machine was released. Particularly, the work of Robertson and Seymour has mainly influenced the developments in this area. For instance, in [17] they show among others that for every fixed graph G, there is a polynomial algorithm to decide whether graph G is a minor of H for some input graph H. For the reverse case, as we deal with here, there are no comparable results known, even for such a well-structured graph as the Chimera. Nevertheless, as the embedding is the first step to be able to run experiments on the D-Wave quantum annealing machine it is studied broadly in this context.

Apart from problem specific approaches, as e.g. in [16], current research mainly splits up into two directions: on the one hand, the goal is to develop an efficient generic heuristic that can embed as many graphs as possible. The first polynomial algorithm was shown by Cai et al. in [3] and is based on finding shortest paths in the hardware graph H. As it considers both, G and H, to be arbitrary input graphs, broken vertices in a non-ideal Chimera are already taken into account. It is still the standard algorithm the package `minorminor` of the D-Wave API is based on [7]. An improvement of this algorithm is suggested in [15] and just recently compared to two new algorithms of Zbinden et al. [24], which show even better performance.

Those heuristic approaches work well in practice, especially for sparse input graphs. However, they have a drawback: if the heuristic fails to embed a graph, it remains unclear whether an embedding is not possible at all or the heuristic just could not find it. There is no guarantee that an embedding can be found or how often the heuristic needs to be repeated until we find one if it exists. As the problem is most likely NP-hard, we will always have to deal with the tradeoff between quality and runtime.

Thus, the second strategy is to insert an intermediate step in the embedding process by using a template with a precomputed fixed embedding acting as a 'virtual hardware' graph. This template has a much simpler structure than the Chimera graph. Thus, on the one hand the computational resources needed to calculate an embedding are decreased, and once it is found, it can be reused for the whole operational period of the machine. On the other hand, simple certificates can be formulated whether a graph is embeddable or not.



(a) standard triangular **(b)** extended triangular **(c)** permutation of diagonal elements

Fig. 1 Different versions of complete graph embeddings in ideal Chimera graph. Each colour represents a single logical vertex (Color figure online)

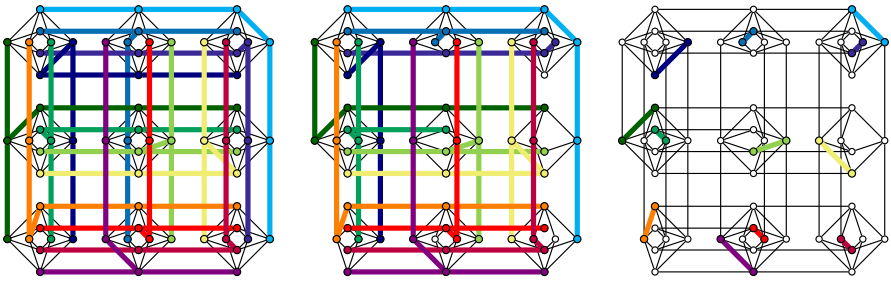
A universal template is the complete graph enabling to embed all graphs with the same or a smaller number of vertices or edges. It completely circumvents the necessity of calculating an embedding for each individual instance but rather provides it straightforwardly. Due to physical restrictions, the Chimera graph of D-Wave was designed to be sparse but nevertheless yield an efficient embedding of the complete graph [4]. The TRIAD layout, presented in [4], forms the basis for the triangle embedding structure of the complete graph in the (ideal) Chimera graph as shown in Fig. 1a for K_{12}. There the set of qubits representing single logical vertex forms a so-called *chain*.

By extending the triangle structure, each of the chains becomes cross-shaped; therefore, we call them *crosses* in the following. Additionally, each pair of crosses is now connected by two edges. Due to this redundancy, the embedding can be extended by splitting one of the crosses into its vertical and horizontal part. Thus, we gained one additional logical vertex, as shown in Fig. 1b. According to [2], this scheme yields an embedding for the complete graph with the largest possible number of vertices.

Unfortunately, due to broken physical vertices the shown templates are not applicable in real hardware. Thus, in [13] an algorithm was proposed trying to extract a subgraph of the broken Chimera where the extended triangle embedding can still be applied and has as many chains as possible. In [2], this approach is generalized by breaking up the triangle structure and placing L-shaped blocks such that all of them overlap pairwise. The principle is illustrated in Fig. 1c. As K. Boothby is one of the main contributors of the D-Wave API, we assume this algorithm is implemented in the package `minorminor` to find complete graphs.

Due to the very limited size of the maximal complete graph, there are various other graphs with less connectivity but a larger number of vertices considered, too. Another good template candidate is the complete bipartite graph, whose embedding is closely related to the one of the complete graph. The idea of [9] is to find the smallest number of vertices that have to be split up into the two partitions such that the resulting graph is bipartite and thus can be embedded using this template. In [18], this approach is elaborated and generalized to related partitioned graph structures.

Known minors can then be collected in a lookup table. The authors of [10] suggest to precompute all ‘maximal minors’ of the complete bipartite graph. This means an input



(a) permutation over all rows and columns **(b)** crosses in broken Chimera **(c)** crossroads in broken Chimera to be found

Fig. 2 By permuting the crossroads, we can find complete graph embeddings in a broken Chimera graph

graph is embeddable if it is a subgraph of one of the contained minors. Another template family are, for instance, the Cartesian products of complete graphs as discussed in [23].

1.3 Our approach

The approach of Boothby et al. [2] to solve the LCGE problem shows a significant advantage over [13] regarding graph sizes. In this work, we further generalize both approaches to still allow for crosses of qubits representing a single logical vertex but also open up the triangle structure.

Figure 2a shows a variant of a complete graph embedding in the ideal Chimera graph. In this construction, every row of qubits is connected to a column of qubits like in the extended triangular embedding in Fig. 1b. But in contrast to Fig. 1b, the edges connecting the horizontal and vertical cross parts do not lie on the diagonal of the Chimera but are distributed over the graph. We call those edges *crossroads* in the following. As each of the crosses occupies the full horizontal, respectively, vertical part, every row, respectively, every column of qubits belongs to a specific cross. For each row and column combination, there is a unique crossroad connecting them. Thus, such an embedding is defined by a matching of rows to columns. In turn, each matching of rows to columns defines a complete graph embedding in the ideal Chimera graph. This means there are a factorial number of possibilities to embed the complete graph.

The redundant edges connecting two crosses would again allow for one more logical vertex by spitting one of the crosses at the crossroad. However, we disregard this, as the redundancy offers another opportunity: the ends of the crosses could be cut off to make room for broken qubits as shown in Fig. 2b. There the remaining, shorter crosses still have an edge to every other cross and thus form a complete graph embedding. But given an arbitrary broken Chimera graph, how do we place the crosses such that this is fulfilled?

By choosing a certain edge connecting a row and a column to locate a crossroad there, the corresponding cross is well defined: both the horizontal and the vertical part are extended until we reach the boundary of the Chimera graph or a broken qubit. To place two crossroads, we need to ensure the resulting crosses ‘meet’ each other, meaning there is at least one edge connecting both. Thus, the LCGE problem can

be reformulated as: how do we match rows with columns to form crossroads, like in Fig. 2c, such that all resulting crosses meet each other?

Clearly, the more restricted the graph the smaller is the number of suitable matchings. While just a few broken vertices might still yield a variety of complete graph embeddings, in particular, if the Chimera graph is very broken, none of the originally large number of possibilities might be valid anymore. Hence, we will not be able to embed the same number of vertices as in the ideal case. Leading to the question, which matching results in the largest possible complete graph? This question is an optimization problem, whose construction we show in the following sections.

For simplification of the notation, we show the construction for the standard symmetric form of the Chimera graph, like in current hardware. But this approach can be extended to arbitrary dimensions. We will further concentrate on finding just a single solution rather than enumerating all possibilities of the same, optimal size as one embedding is sufficient to start calculating on the annealing machine.

The next step after the pure graph embedding in the process of obtaining an embedded Ising model is the distribution of the original problems weights over the various physical vertices. The final weight distribution does depend not only on the original Ising model but also on parameters of the embedding and might influence the performance of the annealing process significantly. For instance, one of the factors that is assumed to have a relevant influence is the so-called *chain length*, here more precise the *cross size*, thus the number of vertices in the crosses.

As we aim for plain embeddability, no further parameters apart from the complete graph size are part of the optimization. To get the final embedding, the crosses are extended until the boundary of the Chimera whether this introduces redundancy in the connecting edges or not. However, exploiting this redundancy might lead to better solutions in terms of the embedding parameters: by cutting off unnecessary vertices from the end of the crosses, the cross size can be reduced. Another option is to select the embedding from the full set of equivalent optimal solutions, where, for example, the cross size is minimal. Both of the mentioned options introduce a second optimization level which would lead far beyond the scope of this article. We keep this for future work.

In Sect. 2, we start with introducing a certain indexing of the Chimera graph, followed by the actual derivation of the optimization problem formulation in Sect. 3. At the end of this section, the complete ILP is summarized. Afterwards, the problem is analysed theoretically in Sect. 4. The results of the experiments explained in Sect. 6 are evaluated in Sect. 7. Finally, in Sect. 8 we present our conclusion.

2 Description of the hardware graph

In this section, we present the Chimera hardware graph with a specific indexing of the graph vertices, being suitable for the formulation of the LCGE problem, and the variable input parameters.

2.1 Chimera graph indexing

First, we introduce some general notations used throughout this work. For some $n, m \in \mathbb{N}$, let $[m; n] := \{m, m+1, \ldots, n\}$ be the enumeration from m to n, where we have $[m; n] = \emptyset$, if $n < m$. For shortness, we use $[n] := [1, n]$ for enumerating from 1, where we say $[0] = \emptyset$. If a set S is the disjoint union of two sets S_1 and S_2, that means $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$, we use $S = S_1 \dot\cup S_2$.

A Chimera graph is defined by a lattice structure of complete bipartite subgraphs, so called *unit cells*, where the number of rows or columns can vary as well as the amount of vertices in the subgraph partitions. We refer to the latter as the *depth* of the Chimera graph. Based on current hardware, the reference is always the ideal symmetric Chimera graph with the number of rows and columns of unit cells given by size $s \in \mathbb{N}$ and a depth of 4, which we denote by $C_{s,s,4}$. Due to the lattice structure, each vertex is represented as a tuple of indices referring to its row and column. For $C_{s,s,4} = (V_{\text{hori}} \dot\cup V_{\text{vert}}, E_{\text{cell}} \dot\cup E_{\text{inter}})$, using the index sets $S := [s]$ and $N := [n]$ with $n := 4s$, we define

- the horizontally connected vertices

$$V_{\text{hori}} := N \times S$$

  with n rows and s columns, which are in the vertically arranged partitions of the unit cells as illustrated in green in Fig. 3a,
- the vertically connected vertices

$$V_{\text{vert}} := S \times N$$

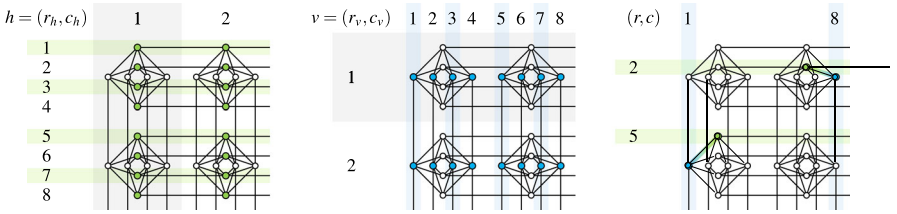
  with s rows and n columns, the horizontal partition illustrated in blue in Fig. 3b,
- the inter unit cell edges $E_{\text{inter}} \subset V_{\text{hori}}^2 \cup V_{\text{vert}}^2$ connecting vertices of different unit cells, which are not needed explicitly in the following and therefore are not specified here, and
- the edges inside of the single unit cells

$$E_{\text{cell}} := \big\{(h, v) = \big((r_h, c_h), (r_v, c_v)\big) \in V_{\text{hori}} \times V_{\text{vert}} \colon r_v = u(r_h), c_h = u(c_v)\big\}.$$

In the latter, we use the function $u \colon N \to S$ with $u(x) = \lceil \frac{x}{4} \rceil$, being the mapping from the *inner row/column* to the *unit cell row/column* index, in the equality constraints to ensure that the paired vertices lie in the same unit cell. Since by this the unit cell rows and columns are given implicitly, we can use the congruent representation

$$\begin{aligned} E_{\text{cell}} &\cong \big\{(r_h, c_v) \colon r_h, c_v \in N\big\} \\ &= \big\{rc \colon r, c \in N\big\} \\ &= N^2 \end{aligned}$$

in the following. In general, we use



(a) horizontal vertices **(b)** vertical vertices **(c)** edge examples $(2, 8)$ and $(5, 1)$

Fig. 3 Specific indexing in Chimera graph (Color figure online)

$$r_{(.)} \colon (x_1, x_2) \mapsto x_1,$$
$$c_{(.)} \colon (x_1, x_2) \mapsto x_2$$

for providing the row, respectively, column for a given vertex, whereas r and c (without further index) always refer to some inner row, respectively, column indices without specifying a certain corresponding vertex. Further, we identify the tuple (r, c) with the non-commutative product rc for shortness to describe an inner unit cell edge. An example for the indexing of edges is shown in Fig. 3c.

2.2 Broken vertices

With regard to real hardware, we consider some vertices to be unavailable. For the symmetric Chimera graph $C_{s,s,4}$ with $s \in \mathbb{N}$ as described in the previous section, let $B_{\text{hori}} \subset V_{\text{hori}}$ and $B_{\text{vert}} \subset V_{\text{vert}}$ be the sets of different broken vertices and $B := B_{\text{hori}} \cup B_{\text{vert}}$. In our experiments, we vary the ratio of broken vertices to the total number of vertices in an ideal Chimera graph, that is,

$$b := \frac{|B|}{|V_{\text{hori}}| + |V_{\text{vert}}|} = \frac{|B|}{8s^2}.$$

While for an ideal Chimera graph the set of possible crossroads defining the crosses in the embedding is just E_{cell}, the available combinations in a broken Chimera graph are restricted to those inner unit cell edges which do not contain a broken vertex:

$$\begin{aligned} A &:= \left\{(h, v) \in E_{\text{cell}} \colon h \in V_{\text{hori}} \setminus B_{\text{hori}}, v \in V_{\text{vert}} \setminus B_{\text{vert}}\right\} \\ &\cong \left\{rc \in N^2 \colon \big((r, u(c)), (u(r), c)\big) \in E_{\text{cell}} \cap ((V_{\text{hori}} \setminus B_{\text{hori}}) \times (V_{\text{vert}} \setminus B_{\text{vert}}))\right\}. \end{aligned}$$

3 ILP formulation

In this section, we construct an integer linear optimization program (ILP) for the LCGE problem over arbitrary input parameters s, B_{hori}, and B_{vert} as described in the previous section.

3.1 Bipartite matching problem

In general, the LGCE as we consider it here is a matching problem: Which row can be matched with which column to form a crossroad in an optimal embedding following our construction rules?

The decision which of the available combinations is taken can be encoded in binary problem variables $x \in \{0, 1\}^A$ with

$$x_{rc} = \begin{cases} 1, & \text{if row } r \text{ is matched to column } c, \\ 0, & \text{otherwise.} \end{cases}$$

We say a crossroad rc is *activated* if its corresponding binary variable x_{rc} is activated in an optimal solution meaning it is set to 1. For simplification, we use $x \in \{0, 1\}^{N \times N}$ in the following with disabling all unavailable row column pairs by presetting $x_{rc} = 0$ for all $rc \in N^2 \setminus A$, although this extends the model with additional variables.

As the goal is to match as much as possible we want to maximize the number of activated binary variables, the objective is

$$\sum_{rc \in A} x_{rc} = \sum_{rc \in N^2} x_{rc}.$$

Our construction is based on crossroads joining full rows and columns. Therefore, only one crossroad per row and column is allowed. This can be enforced by the matching constraints

$$\begin{aligned} \sum_{r\tilde{c} \in A} x_{r\tilde{c}} = \sum_{r \in N} x_{r\tilde{c}} \leq 1 \quad \forall \tilde{c} \in N, \\ \sum_{\tilde{r}c \in A} x_{\tilde{r}c} = \sum_{c \in N} x_{\tilde{r}c} \leq 1 \quad \forall \tilde{r} \in N. \end{aligned} \tag{1}$$

Those types of constraints are also called *cardinality constraints* as they enforce choosing a certain number of members, here just one, out of a given set. By these restrictions, the optimal value of the objective function corresponds to the size, meaning the number of vertices, of the largest embeddable complete graph. Additionally, they confirm the upper bound on the objective function of $n = 4s$, which is the maximal size of a complete graph in $C_{s,s,4}$ using our construction as explained in Sect. 1.3.

Until now, the constructed problem is just a simple *maximum bipartite matching problem*. In the following, we show further constraints that need to be added.

3.2 Mutually exclusive sets constraints

If a horizontal vertex is broken, it interrupts the horizontal path from the left to the right. This prevents a cross occupying this row to be extended to the boundaries of the Chimera graph. It is analogous for a broken vertical vertex and a cross using the corresponding column. This needs to be taken into account when considering

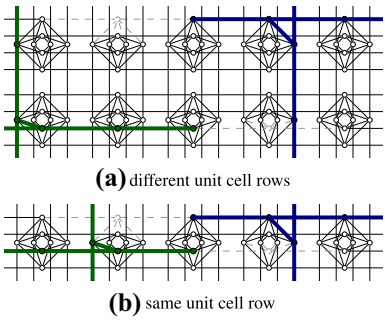


Fig. 4 Examples of crosses that do not meet due to broken vertices (gray, dashed)

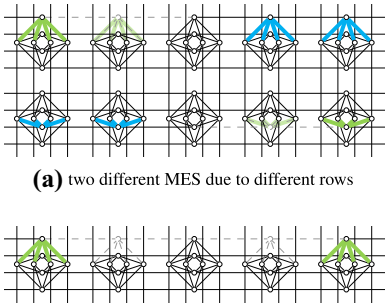
possible crossroad candidates for the embedding. Figure 4 depicts an example of such a situation. Due to the broken vertices, the corresponding crosses for certain pairs of crossroads might not meet. This means there do not exist any edges between the different vertices of the crosses, even if they reach the same unit cell like in Fig. 4b. But at least one edge is needed to provide a valid embedding of two vertices of the complete graph. Therefore, those crossroads cannot be activated together and we need to introduce further constraints enforcing the activation of only one of them.

We will see that there are not only isolated pairs but clusters of crossroads all being pairwise forbidden, which means only one of all of them can be activated. We call those clusters mutually exclusive sets (MES). The construction of those sets differs for certain pairs of broken vertices. We have the following cases, which are handled separately in the next paragraphs:

1. two broken horizontal vertices,
2. two broken vertical vertices,
3. two different broken vertices, one horizontal and one vertical.

1. Let $h = (r_h, c_h) \neq k = (r_k, c_k) \in B_{\text{hori}}$ be **two broken horizontal vertices**, as illustrated in Fig. 5a, b. Due to the horizontal interruption, the crossroads on the left and the right of the two vertices are affected. In Fig. 5b, both broken vertices lie in the same inner row. As the matching constraints already permit only one crossroad per row, no further constraints are necessary in this case and we can restrict to vertices with $r_h \neq r_k$, which is the case shown in Fig. 5a. There we have two MES, which are highlighted in different colours. Each of the blue crossroads in the right top corner cannot be activated together with the others in this corner due to the matching constraints, and they cannot be activated together with the blue in the left bottom corner because their corresponding crosses would not meet. The same holds for the green crossroads in the opposite corners.

For the definition of the MES, we need all crossroads from the left boundary until the leftmost broken vertex and all crossroads from the rightmost broken vertex until the right boundary in the two corresponding inner rows. The incident edges (light green) to the broken vertices are excluded by definition of A, respectively, set to 0, but again are included in the definition of the two sets for simplicity. Let for example h be the top left broken vertex in Fig. 5a and $I_{\text{left}} \subseteq N$ describe the interval of columns on the left and $I_{\text{right}} \subseteq N$ on the right. The set of blue crossroads in the left top corner is then given by combining the row r_h with each of the columns in I_{left}. The remaining blue



(a) two different MES due to different rows

(b) MES in same row already handled by matching constraints

Fig. 5 Sets of mutually exclusive crossroads caused by two broken horizontal vertices (Color figure online)

crossroads combine r_k with I_{right}. This results in the MES $\big(\{r_h\} \times I_{\text{left}}\big) \cup \big(\{r_k\} \times I_{\text{right}}\big)$. For the green crossroads, we get symmetrically $\big(\{r_k\} \times I_{\text{left}}\big) \cup \big(\{r_h\} \times I_{\text{right}}\big)$.

The intervals of columns, I_{left} and I_{right}, can be derived from the broken vertices' columns depending on the relational position of the vertices. To describe this more formally, for the fixed size n let

$$I(s_1, s_2) := \begin{cases} [4s_1], & \text{if } s_1 \leq s_2, \\ [4(s_1 - 1) + 1; n] = [4s_1 - 3; n], & \text{otherwise} \end{cases}$$

be the interval to or from s_1 depending on the relation to s_2 for $s_1, s_2 \in S$. The multiplication with 4 is needed for the conversion of unit cell to inner columns. The behaviour of this function is illustrated in Fig. 6 for different relations. By the subtraction of $\frac{1}{2}$, we can circumvent the fact that I only returns the left interval for identical inputs when we need the right one. The resulting sets of mutually exclusive crossroads can then be defined for each combination of h and k with

$$\begin{aligned} X_1(h,k) &:= \big(\{r_h\} \times I(c_h, c_k)\big) \cup \big(\{r_k\} \times I(c_k, c_h - \tfrac{1}{2})\big), \\ X_2(h,k) &:= \big(\{r_k\} \times I(c_h, c_k)\big) \cup \big(\{r_h\} \times I(c_k, c_h - \tfrac{1}{2})\big). \end{aligned} \tag{2}$$

Therefore, we get the cardinality constraints for $i = 1, 2$

$$\sum_{rc \in X_i(h,k)} x_{rc} \leq 1,$$

where the sum, e.g. for $i = 1$, can also be written as

$$\sum_{c \in I(c_h, c_k)} x_{r_h c} + \sum_{c \in I(c_k, c_h - \frac{1}{2})} x_{r_k c}.$$

2. **Two vertical broken vertices**, with $v = (r_v, c_v) \neq w = (r_w, c_w) \in B_{\text{vert}}$, can be handled analogously to the case before by exchanging row and column: We can restrict on $c_v \neq c_w$. Taking all rows from the upper boundary to the uppermost broken vertex

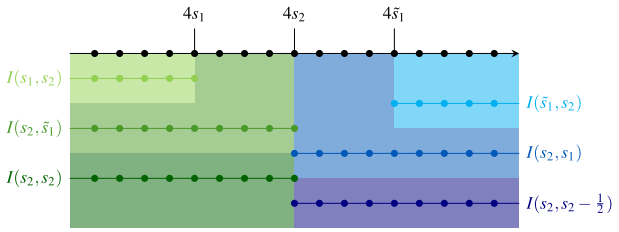


Fig. 6 Representation of interval function

and all from the bottom to the lowest broken vertex results in the sets of mutually exclusive crossroads

$$\begin{aligned} X_1(v, w) &:= \big(I(r_v, r_w) \times \{c_v\}\big) \cup \big(I(r_w, r_v - \tfrac{1}{2}) \times \{c_w\}\big), \\ X_2(v, w) &:= \big(I(r_v, r_w) \times \{c_w\}\big) \cup \big(I(r_w, r_v - \tfrac{1}{2}) \times \{c_v\}\big). \end{aligned} \tag{3}$$

Therefore, we get exemplary the constraint for $i = 1$

$$\sum_{rc \in X_1(v,w)} x_{rc} = \sum_{r \in I(r_v, r_w)} x_{rc_v} + \sum_{r \in I(r_w, r_v - \frac{1}{2})} x_{rc_w} \leq 1.$$

Let us combine the MES for all combinations in

$$\begin{aligned} \mathscr{X}_{\text{hori}} &:= \bigcup \big\{ \{X_1(h, k), X_2(h, k)\} : h, k \in B_{\text{hori}},\ r_h \neq r_k \big\}, \\ \mathscr{X}_{\text{vert}} &:= \bigcup \big\{ \{X_1(v, w), X_2(v, w)\} : v, w \in B_{\text{vert}},\ c_v \neq c_w \big\}. \end{aligned}$$

3. The case with **two different broken vertices** $h = (r_h, c_h) \in B_{\text{hori}}$ and $v = (r_v, c_v) \in B_{\text{vert}}$ is different to the ones above. As shown in Fig. 7a, we have four different cases depending on the relational position of the two vertices, whether the vertical is above or below,

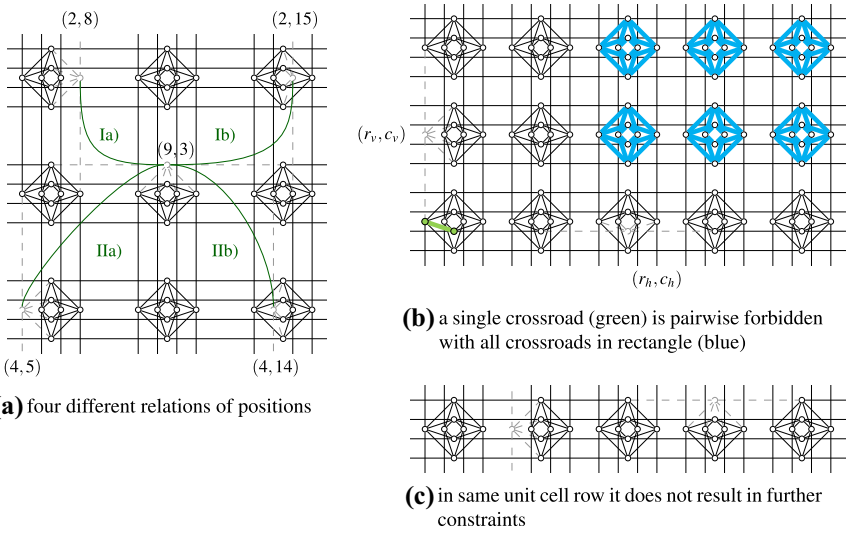
I) $r_v < u(r_h)$ or
II) $r_v > u(r_h)$,

and left or right,

a) $c_h > u(c_v)$ or
b) $c_h < u(c_v)$,

of the horizontal broken vertex.

Figure 7b shows the combination **Ia)** exemplary. The other cases **Ib)**, **IIa)** and **IIb)** can be derived analogously but mirrored to different corners. This is covered by the definition of I, which we use again in the following construction; therefore, this holds for all cases. Further, we can see in Fig. 7c that no additional constraints are provided if both vertices lie in the same unit cell row or column, respectively. Therefore, we can restrict on cases with $u(r_h) \neq r_v$ and $u(c_v) \neq c_h$.



(a) four different relations of positions

(b) a single crossroad (green) is pairwise forbidden with all crossroads in rectangle (blue)

(c) in same unit cell row it does not result in further constraints

Fig. 7 Mutually exclusive crossroads caused by two different broken vertices (Color figure online)

Due to the path interruption by the broken vertices, we get exactly one crossroad, $r_h c_v \cong \big(r_h, u(c_v), u(r_h), c_v\big)$, in the lower left corner of Fig. 7b illustrated in green, which is pairwise forbidden with all the crossroads in a rectangle, which are shown in blue. In the following, we refer to $r_h c_v$ as the *common crossroad*. In the case shown in Ia), the rectangle includes all rows from the upper boundary until the unit cell of the broken horizontal vertex v and all columns starting at the unit cell of the broken vertical vertex v until the right boundary, which are the combinations in $[4r_v] \times [4(c_h - 1) + 1; n]$. More generally, the rectangle is described by

$$I(r_v, u(r_h)) \times I(c_h, u(c_v)).$$

The pairwise constraints

$$x_{r_h c_v} + x_{rc} \leq 1 \qquad \forall rc \in I(r_v, u(r_h)) \times I(c_h, u(c_v))$$

would therefore be sufficient to describe our problem. But taking advantage of the matching constraints (1) again, we can aggregate the crossroads in the rectangle: either all in one inner row or all in one inner column. To keep the optimization problem description as small as possible, we take the smallest amount of resulting MES. This is given by the minimum of the dimensions of the rectangle, hence the number of rows $|I(r_v, u(r_h))|$ or the number of columns $|I(c_h, u(c_v))|$. With

$$X_r(h, v) := \{r_h c_v\} \cup \big(\{r\} \times I(c_h, u(c_v))\big),$$

describing the aggregated MES for a row $r \in I(r_v, u(r_h))$, and analogously

$$X_c(h, v) := \{r_h c_v\} \cup \big(I(r_v, u(r_h)) \times \{c\}\big),$$

for a column $c \in I(c_h, u(c_v))$, we can define

$$\mathscr{X}_{\text{mix}}(h, v) := \begin{cases} \{X_r(h, v) \colon r \in I(r_v, u(r_h))\}, & \text{if } |I(r_v, u(r_h))| \leq |I(c_h, u(c_v))|, \\ \{X_c(h, v) \colon c \in I(c_h, u(c_v))\}, & \text{otherwise,} \end{cases}$$

choosing the set of MES with the smallest cardinality for a certain pair of broken vertices v and h. With

$$\mathscr{X}_{\text{mix}} := \bigcup \{ \mathscr{X}_{\text{mix}}(h, v) : (h, v) \in B_{\text{hori}} \times B_{\text{vert}},\ r_v \neq u(r_h),\ c_h \neq u(c_v) \},$$

we can finally summarize all cardinality constraints to

$$\sum_{rc \in X} x_{rc} \leq 1 \quad \forall X \in \mathscr{X}_{\text{hori}} \cup \mathscr{X}_{\text{vert}} \cup \mathscr{X}_{\text{mix}}. \tag{4}$$

3.3 Embedding ILP in a nutshell

With the definitions of the section before we can summarize the complete embedding problem in the following ILP formulation. If we find an optimal solution to this ILP, its objective value corresponds to the size of the largest embeddable complete graph and the activated variables define the crossroads for a corresponding embedding.

$$\begin{aligned} \max \quad & \sum_{rc \in N^2} x_{rc} && \text{(E)} \\ \text{s.t.} \quad & \sum_{r \in N} x_{r\tilde{c}} \leq 1 && \forall \tilde{c} \in N \\ & \sum_{c \in N} x_{\tilde{r}c} \leq 1 && \forall \tilde{r} \in N \\ & \sum_{rc \in X} x_{rc} \leq 1 && \forall X \in \mathscr{X}_{\text{hori}} \cup \mathscr{X}_{\text{vert}} \cup \mathscr{X}_{\text{mix}} \\ & x_{rc} = 0 && \forall rc \in N^2 \setminus A \\ & x \in \{0, 1\}^{N \times N} \end{aligned}$$

3.4 Embedding extraction

Once a solution $x^* \in \{0, 1\}^{N \times N}$ for the ILP shown in the previous section is found, we can construct the actual embedding from the activated crossroads. For this, we extend the crossroad in the corresponding row to the left and the right and in the

corresponding column to the top and the bottom until we meet a broken vertex or the boundary of the Chimera to obtain the corresponding cross.

For a crossroad $rc \in N^2$ with $x^*_{rc} = 1$, the vertices to the left of the crossroad are located in the same row r, while the column index ranges from 1 to $u(c) - 1$. More formally, this means the set $\{r\} \times [1; u(c) - 1]$. Remember $u(c)$ defines the unit cell column index corresponding to the inner column index. If there is at least one broken vertex, we need to find the rightmost one among all of them as we cannot extend the cross to left beyond it. The rightmost broken vertex has the largest column index and thus can be found with

$$\max\{\tilde{c} \in [1; u(c) - 1] : (r, \tilde{c}) \in B_{\text{hori}}\}.$$

Since the cross does not include the broken vertex, we need to introduce a shift of one and therefore can obtain the vertices in the left part of the cross with

$$P_{\text{left}}(r, c) = \max\{\tilde{c} \in [1; u(c)] : \tilde{c} = 1 \vee (r, \tilde{c} - 1) \in B_{\text{hori}}\},$$

where the disjunction with $\tilde{c} = 1$ is needed to default to the start value 1 of the index range in case the set of broken vertices is empty. Analogously, we can construct the parts to the right, top and bottom with

$$\begin{aligned} P_{\text{right}}(r, c) &= \min\{\tilde{c} \in [u(c); s] &&: \tilde{c} = s \vee (r, \tilde{c} + 1) \in B_{\text{hori}}\} \\ P_{\text{top}}(r, c) &= \max\{\tilde{r} \in [1; u(r)] &&: \tilde{r} = 1 \vee (\tilde{r} - 1, c) \in B_{\text{vert}}\} \\ P_{\text{bot}}(r, c) &= \min\{\tilde{r} \in [u(r); s] &&: \tilde{r} = s \vee (\tilde{r} + 1, c) \in B_{\text{vert}}\}. \end{aligned}$$

All in all, we obtain the plus-shaped vertex set with

$$\big(\{r\} \times [P_{\text{left}}(r, c); P_{\text{right}}(r, c)]\big) \cup \big([P_{\text{top}}(r, c); P_{\text{bot}}(r, c)] \times \{c\}\big),$$

for all $rc \in N^2$ with $x^*_{rc} = 1$.

As an equivalent to the chain length, an important parameter in the further processing of the Ising model as described in Sect. 1.3, we can observe the *cross size*. The maximum number of vertices in a cross in a Chimera of size s is $2s$, which is only the case when the cross can be extended fully to all boundaries, meaning $P_{\text{left}}(r, c) = P_{\text{top}}(r, c) = 1$ and $P_{\text{right}}(r, c) = P_{\text{bot}}(r, c) = s$.

Clearly, by this construction we obtain embeddings with a larger number of vertices than presented in [2], where the maximum chain length is $s + 1$. However, this allows for larger graph sizes as we will see in the following section.

4 Analysis

In this section, we investigate the structure of the ILP embedding problem by discussing its size, complexity and variations. The solvability of the problem can be estimated by different parameters. The size of the ILP, more precisely the number of variables and constraints, is of interest when directly passing the constructed ILP to ILP solvers

and using them without any further specifications. However, the specific structure of the constraints allows for a deeper complexity analysis of the problem showing fixed-parameter tractability. At the end of this section, we give a short outlook on how the ILP can be extended to more general Chimera graphs.

4.1 Size of the ILP

We estimate the size of the ILP with regard to the input parameters s, B_{hori} and B_{vert}. The number of variables is $n^2 = 16s^2$ if we also take the unavailable combinations in A into account. By removing them, we get $n^2 - |A| \geq n^2 - |B_{\text{hori}}| - |B_{\text{vert}}|$, where the lower bound is achieved only if no two broken vertices meet in one edge.

Apart from the $2n$ matching constraints in (1), we show that the number of additional constraints is also polynomial in the number of broken vertices. We need to count the number of MES that are constructed in the former section for the different combinations of vertices. Taking two unequal vertices out of the broken horizontal vertices B_{hori}, we get

$$\binom{|B_{\text{hori}}|}{2} = \tfrac{1}{2}\,|B_{\text{hori}}|\,(|B_{\text{hori}}| - 1) = \tfrac{1}{2}\,|B_{\text{hori}}|^2 - \tfrac{1}{2}\,|B_{\text{hori}}|$$

combinations. Analogously, for two broken vertical vertices out of B_{vert} we have

$$\binom{|B_{\text{vert}}|}{2} = \tfrac{1}{2}\,|B_{\text{vert}}|^2 - \tfrac{1}{2}\,|B_{\text{vert}}|$$

combinations. On the other hand, the number of combinations for two different broken vertices is $|B_{\text{hori}}|\,|B_{\text{vert}}|$. Those numbers could be slightly but not significantly reduced when taking pairs into account that lie in the same rows, respectively, columns.

For each combination of two broken vertices of the same type we have two constraints. Thus we have

$$\begin{aligned} |\mathscr{X}_{\text{hori}}| &\leq |B_{\text{hori}}|^2 - |B_{\text{hori}}| \\ |\mathscr{X}_{\text{vert}}| &\leq |B_{\text{vert}}|^2 - |B_{\text{vert}}|\,. \end{aligned}$$

For the different broken vertices, the number of constraints depends on the size of the corresponding rectangles. Here, we can only estimate the worst-case scenario which is $n - 1$ constraints, hence

$$|\mathscr{X}_{\text{mix}}| \leq (n-1)\,|B_{\text{hori}}|\,|B_{\text{vert}}|\,.$$

Therefore, in total we get

$$\begin{aligned} |\mathscr{X}_{\text{hori}}| + |\mathscr{X}_{\text{vert}}| + |\mathscr{X}_{\text{mix}}| \leq |B_{\text{hori}}|^2 - |B_{\text{hori}}| + |B_{\text{vert}}|^2 - |B_{\text{vert}}| \\ + (n-1)\,|B_{\text{hori}}|\,|B_{\text{vert}}| \end{aligned}$$

additional cardinality constraints in (4).

4.2 Problem complexity

The described problem is a matching problem on a bipartite graph. The simple version, without additional constraints, can be solved in polynomial time, e.g. with the algorithm of Hopcroft and Karp in $\mathcal{O}\left(n^{2.5}\right)$ [11]. Due to the constraints (4), introduced in Sect. 3.2, our ILP corresponds to a so-called *restricted maximum matching problem*. Those problems are NP-hard in general, and this even holds for cardinality constraints with a cardinality of just one like ours [21]. But exploiting the specific structure of those constraints, we can derive that the runtime is mainly dominated by the broken vertices compared to the size of the Chimera graph. More formally, this means the problem is *fixed-parameter tractable* with the number of the broken vertices $|B|$ as the parameter. We show the fixed-parameter tractability by enumerating the decisions that have to be made for removing constraints until the problem is a simple maximum bipartite matching problem.

Considering the constraint for two broken vertices of the same type, we have two MES, (2), respectively, (3). As it is shown in Sect. 3.2, both MES consist of a left and a right part lying in different rows for broken horizontal vertices, respectively, an upper and a lower part in different columns for broken vertical vertices. Since we can only take one of the crossroads in an MES into a solution, this crossroad is either in the left or in the right, respectively, upper or lower, part. Imagine we decide in advance for one part of the MES. Considering, for instance, some $X \in \mathscr{X}_{\text{hori}}$, with $X =: X_{\text{left}} \cup X_{\text{right}}$ for simplicity, we could choose the crossroad to be in X_{left}. Thus, none of the crossroads in X_{right} can be activated in the solution, meaning we have to set $x_{rc} = 0$ for all $rc \in X_{\text{right}}$. The corresponding cardinality constraint reduces to

$$\sum_{rc \in X} x_{rc} = \sum_{rc \in X_{\text{left}}} x_{rc} \leq 1.$$

As X_{left} only consists of crossroads in a certain row, this constraint is weaker than the matching constraint of (1) covering that row fully. Thus, we can remove it and the resulting optimization problem, having less variables and less constraints, is easier to solve.

By considering both exclusive options, disregarding X_{right} or X_{left}, and choosing the best solution, we get the global optimum. This procedure can be applied for every MES in $\mathscr{X}_{\text{hori}}$ and $\mathscr{X}_{\text{vert}}$, especially this can be done iteratively to already simplified versions. With two parts for each MES, this results in total in

$$2^{|\mathscr{X}_{\text{hori}}|} \cdot 2^{|\mathscr{X}_{\text{vert}}|} \leq 2^{|B_{\text{hori}}|^2 - |B_{\text{hori}}| + |B_{\text{vert}}|^2 - |B_{\text{vert}}|}$$

different simplified problems.

For different broken vertices, we have much more constraints, but they all have one crossroad in common that cannot be matched together with the other concerned crossroads in the rectangle. Therefore, we can proceed similarly to above. One option is just taking the single common crossroad into the solution and rejecting all of the rectangle. This means the binary variable corresponding to this crossroad is set to 1,

while those for the rectangle are set to 0. By this not only the constraints are removed but the size of the ILP is appreciably reduced, persisting for all problems resulting from subsequent decisions. However, for an increasing size of the rectangle it gets more unlikely that the common crossroad is part of an optimal solution. Hence, the second option is rejecting this single crossroad. This again results in weaker remaining constraints than the matching constraints for the whole rectangle, and they can be dropped. These two possibilities per broken vertex pair result in total in further

$$2^{|B_{\text{hori}}||B_{\text{vert}}|}$$

options.

Finally, we get at maximum

$$2^{|B_{\text{hori}}||B_{\text{vert}}|+|B_{\text{hori}}|^2-|B_{\text{hori}}|+|B_{\text{vert}}|^2-|B_{\text{vert}}|}$$

different simplified versions of our original problem. As we removed all of the additional constraints along the decision tree, they are now simple maximum bipartite matching problems and can be solved efficiently. With

$$\begin{aligned}
&|B_{\text{hori}}|^2 - |B_{\text{hori}}| + |B_{\text{vert}}|^2 - |B_{\text{vert}}| + |B_{\text{hori}}|\,|B_{\text{vert}}| \\
&\quad \leq |B_{\text{hori}}|^2 + |B_{\text{vert}}|^2 + 2\,|B_{\text{hori}}|\,|B_{\text{vert}}| \\
&\quad = \big(\,|B_{\text{hori}}| + |B_{\text{vert}}|\,\big)^2 \\
&\quad = |B|^2
\end{aligned}$$

we get a worst-case runtime in

$$\mathcal{O}\left(2^{|B|^2}\, n^{2.5}\right)$$

and the problem is fixed-parameter tractable in the choice of the broken vertices.

According to current hardware development, we can reasonably assume that $|B|$ is small compared to n. Therefore, considering $|B|$ to be fixed, the problem is efficiently solvable for increasing size n. However, this means at the same time the ratio of broken vertices R is decreasing as it is inversely proportional to n^2. Thus, considering the ongoing development of the annealing machines, the exponential dependency of the runtime on the number of broken qubits will be negligible in contrast to the just polynomial dependency on the size of the Chimera.

Just keeping R fixed, like we did in our experiments for comparison with [2], still provides an exponential runtime. This aspect demands for heuristic solving approaches like presented in the following section. However, once the embedding is computed for a hardware graph, it can be reused during the whole operating period. This justifies a larger runtime than in the operational approach calculating a new embedding for each Ising instance.

4.3 Generalization

In Sect. 3, we show the construction of the ILP for the LCGE problem exemplary for the symmetric Chimera graph with a depth of 4. But the whole setup can also be generalized for arbitrary Chimera graphs C_{s_R,s_C}, d, hence which are rectangular meaning $s_C \neq s_R$ or which have a different depth d. In the following, we briefly mention the adjustments that need to be applied.

If d is different than 4, the unit cell index function changes to $u : x \mapsto \lceil \frac{x}{d} \rceil$. In case of $s_R \neq s_C$ we need to split N^2 up in $R \times C$ with the row, respectively, column sets $R = [ds_R]$ and $C = [ds_C]$. Of course, in this case the maximal number of vertices in an embeddable complete graph, even if it is an ideal Chimera, is just $d \min\{s_R, s_C\}$. As the amount of rows and columns is not equal anymore, we have to adjust the interval function to be able to differ between maximal row or column with

$$I_{R/C}(s_1, s_2) := \begin{cases} [4s_1], & \text{if } s_1 \leq s_2, \\ [4s_1 - 3; ds_{R/C}], & \text{otherwise.} \end{cases}$$

With these modifications, it is possible to construct the analogous matching constraints as well as the MES for the cardinality constraints. Analogously, the extraction of the embedding from a found solution can be adjusted.

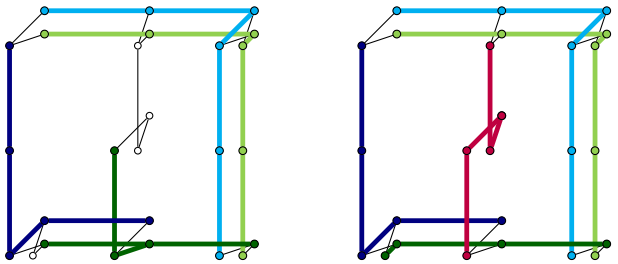
One might also consider to extend the model by adding broken edges, which could possibly be handled in an analogue case differentiation as for the broken vertices. But in contrast to the restriction to broken vertices the implications on the model construction and therefore the size and complexity are not trivial. Further, a broken edge adjoining non-broken vertices is very rare and thus can be handled by marking one of the concerned vertices as broken. Therefore we do not discuss this in more detail here.

4.4 Delineation

In our construction, the embedding corresponding to a single logical vertex is formed by a cross. In the case of many broken vertices, this assumption might be too restrictive. This is shown, for example, in Fig. 8 where the solution to the ILP is not as good as the optimal solution to the LCGE problem. We believe, however, that such corner cases are of less practical relevance since they just seem to occur for a very large ratio of broken vertices.

5 Heuristic ILP

The complexity analysis in Sect. 4.2 has shown a certain structure of the additional constraints. Especially for the case of two broken vertices of different types, there is a strong imbalance: activating the single common crossroad excludes all the crossroad in the corresponding rectangle. Thus, it is very unlikely that this crossroad is part of the optimal solution in particular for growing size of the rectangle. In this section,



(a) ILP solution **(b)** optimal solution

Fig. 8 Very restricted Chimera graph containing larger complete graph than can be found with our ILP

we show the derivation of a simpler ILP whose solution is assumed to be close to the optimal one.

5.1 Reducing size

We decided to test a heuristic approach based on excluding such unlikely common crossroads in advance. This reduces the number of variables and more importantly the number of constraints. Thus, we solve only a certain part of the decision tree constructed in Sect. 4.2, and therefore, it is not clear whether the optimal value can be achieved.

We introduce a parameter defining which common crossroads shall be removed, respectively, kept: the so-called *maximum rectangle ratio*, denoted here by m with $0 \leq m \leq 1$, gives a boundary on the size of the rectangle relative to the Chimera graph size s below which the common crossroad is kept. If the number of unit cell rows times the number of columns of the rectangle exceeds ms^2, the crossroad is excluded. More formally, this means for two different broken vertices $h = (r_h, c_h) \in B_{\text{hori}}$ and $v = (r_v, c_v) \in B_{\text{vert}}$, that we do not use the crossroad $r_h c_v$, hence set $x_{r_h c_v} = 0$ in advance, if we have $M(h, v) \geq ms^2$ with

$$M(h, v) := |I(r_v, u(r_h))| \cdot |I(c_h, u(c_v))|.$$

Thus, a ratio of 1 means all common crossroads are kept, while a ratio of 0 means none of them remain in the resulting optimization problem.

Given m, let the set of unused crossroads be

$$U^m := \left\{ r_h c_v : (h, v) \in B_{\text{hori}} \times B_{\text{vert}},\ M(h, v) \geq ms^2 \right\}$$

and further let

$$\mathscr{X}^m_{\text{mix}} := \bigcup \left\{ \mathscr{X}_{\text{mix}}(h, v) : (h, v) \in B_{\text{hori}} \times B_{\text{vert}},\ r_v \neq u(r_h),\ c_h \neq u(c_v),\ M(h, v) < ms^2 \right\}.$$

be the reduced set of MES. With this, the corresponding constraints can be simplified by replacing $\mathscr{X}_{\text{mix}}$ with $\mathscr{X}^m_{\text{mix}}$ in (E). This can be seen in the following section summarizing the heuristic ILP.

5.2 Heuristic Embedding ILP in a nutshell

With the same definitions as before and a certain choice of m, we can now summarize the heuristically reduced embedding problem in the ILP formulation:

$$
\begin{aligned}
\max \quad & \sum_{rc \in N^2} x_{rc} && && \text{(H)}\\
\text{s.t.} \quad & \sum_{r \in N} x_{r\tilde{c}} \leq 1 && \forall \tilde{c} \in N\\
& \sum_{c \in N} x_{\tilde{r}c} \leq 1 && \forall \tilde{r} \in N\\
& \sum_{rc \in X} x_{rc} \leq 1 && \forall X \in \mathscr{X}_{\text{hori}} \cup \mathscr{X}_{\text{vert}} \cup \mathscr{X}^m_{\text{mix}}\\
& x_{rc} = 0 && \forall rc \in N^2 \setminus A \cup U^m\\
& x \in \{0,1\}^{N \times N}
\end{aligned}
$$

6 Experimental setup

6.1 Random instances

To be able to compare our approach to current state-of-the-art methods, we consider different ratios of broken vertices for growing hardware sizes. We have generated ten instances for each combination of the following values:

- sizes of Chimera graph: $s \in \{4, 6, 8, \ldots 32, 34\}$,
- ratios of broken vertices: $b \in \{0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2\}$.

The ratio of the broken vertices times the total number of vertices in the ideal Chimera graph results in the number of broken vertices for a certain size. For each of the ten instances, we randomly chose this number out of all vertices and marked them as being broken. Due to rounding to whole vertices, the resulting exact ratios differ slightly from the aimed ones above, especially for smaller graph sizes.

As a reference, we like to remark the parameters of two real D-Wave 2000Q systems. First, the solver `DW_2000Q_6`, which we accessed though the Jülich UNified Infrastructure for Quantum computing (JUNIQ), has a size of 16 and 7 broken vertices. This corresponds to a ratio of about 0.0034. Second, the older USRA/NASA chip with

the same size had 17 broken vertices, resulting in a ratio of about 0.0083 [12]. Thus, our experiments are much more exhaustive than current hardware demands.

For the heuristic approach, we use $m = 0$ and $m = 0.25$ for our experiments as reference points to evaluate the impact of removing a significant number of crossroads.

6.2 Solving strategy

This paper focuses on the presentation of the embedding problem as an ILP. We did not implement an algorithm, yet, which exploits the branching procedure as described in Sect. 4.2. It is not straightforward how the decision tree could be reduced at certain stages. As we do not consider the ratio of broken vertices to be fixed here, there is still an exponential overhead in the number of final simplified problems. Even the simplified heuristic version is still a hard optimization problem.

Thus, using an ILP solver, already taking advantage of implemented branch-and-bound techniques, is a good starting point to evaluate the capabilities of the model. We decided to pass the models (E) and (H) directly to the solver SCIP [8] without further adjustments. It is currently one of the fastest non-commercial solvers for mixed integer programming, which includes ILP.

6.3 Specifications

The experiments were run on a Dell Precision 5820 Tower workstation with a Intel Xeon(R) W-2175 CPU @ 2.50GHz 28, 128 RAM and operating system Ubuntu Linux 18.04.5 LTS. We implemented our code in python and used the python interface package `pyscipopt` [14] to connect to the solver SCIP with version 6.0.1 [8]. As this interface does not support parallel mode, we could just use one core. We set a timeout of 1 hour for solving each instance with SCIP. Building up the model was not included in there. As we first want to evaluate the capabilities of the model and the derived heuristic version themselves, we did not optimize our code regarding performance. Apart from the timeout, we used the default SCIP parameters.

7 Results

A good reference to estimate the quality of a solution is to compare its objective value, the found graph size, to the largest possible size of a complete graph in the ideal Chimera graph. As stated in Sect. 1.3 with our construction using crosses, the largest complete graph in a Chimera graph of size s is K_{4s}. Let $G_{s,b,i}$ be the graph size returned by SCIP within one hour for the ith instance with Chimera size s and ratio of broken vertices b. As we consider ten instances for each parameter combination, the found graph sizes are averaged and we introduce the *averaged solution ratio*

$$\overline{R}_{s,b} := \frac{\frac{1}{10}\sum_{i=0}^{9} G_{s,b,i}}{4s} = \sum_{i=0}^{9} \frac{G_{s,b,i}}{40s}$$

Table 1 Averaged solution ratio $\overline{R}_{s,b}$ for each combination of size s and ratio of broken vertices b

[for exact model (E)]

b \ s	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
0.005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.97	0.91	0.82	0.85
0.02	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.95	0.77	0.68	0.70	0.63	0.60	0.60
0.03	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94	0.63	0.59	0.59	0.58	0.51	0.51	0.52	0.44
0.04	1.00	1.00	1.00	0.99	0.99	0.99	0.88	0.60	0.61	0.56	0.50	0.52	0.43	0.44	0.43	0.39
0.05	0.99	0.99	0.98	0.99	0.98	0.94	0.65	0.59	0.43	0.51	0.44	0.43	0.38	0.37	0.35	0.38
0.1	0.95	0.93	0.86	0.81	0.69	0.52	0.37	0.35	0.30	0.29	0.25	0.25	0.24	0.22	0.21	0.17
0.2	0.72	0.60	0.53	0.48	0.39	0.22	0.20	0.16	0.17	0.17	0.12	0.14	0.11	0.10	–	–

[for heuristic model (H) with $m = 0.25$]

b \ s	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
0.005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.88	0.87	0.86
0.02	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.94	0.83	0.75	0.76	0.77	0.73	0.69
0.03	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.78	0.70	0.67	0.67	0.62	0.62	0.61	0.59
0.04	1.00	1.00	1.00	0.99	0.99	0.99	0.97	0.74	0.70	0.63	0.61	0.60	0.56	0.53	0.49	0.48
0.05	0.99	0.99	0.98	0.99	0.98	0.95	0.72	0.67	0.60	0.57	0.52	0.50	0.49	0.45	0.42	0.41
0.1	0.95	0.93	0.86	0.81	0.71	0.59	0.43	0.41	0.36	0.32	0.31	0.29	0.28	0.23	0.24	0.21
0.2	0.72	0.60	0.52	0.48	0.42	0.36	0.31	0.22	0.17	0.16	0.13	0.12	0.11	0.09	0.09	0.08

[for heuristic model (H) with $m = 0.0$]

b \ s	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
0.005	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.02	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.98	0.97	0.92
0.03	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.96	0.94	0.89	0.86	0.81	0.79	0.74	0.69
0.04	1.00	1.00	1.00	0.99	0.99	0.98	0.96	0.90	0.84	0.79	0.75	0.68	0.62	0.58	0.53	0.49
0.05	0.99	0.99	0.98	0.99	0.96	0.91	0.82	0.78	0.68	0.63	0.57	0.50	0.45	0.39	0.37	0.35
0.1	0.95	0.90	0.77	0.67	0.51	0.43	0.37	0.29	0.26	0.19	0.15	0.14	0.11	0.08	0.06	0.06
0.2	0.66	0.43	0.28	0.22	0.13	0.09	0.06	0.03	0.02	0.01	0.01	0.01	0.00	0.01	0.00	0.00

Colours indicate the number of instances that where solved to optimality (white: all 10, yellow: 1 to 9, red: none)

as a measure for the solution quality. Table 1 shows the resulting ratios for each of the models.

In Table 1a, we can see a clear boundary between the instances which can be solved in one hour and which cannot. The former are either instances with a small Chimera size or with a small ratio of broken vertices. Here, we already see a slight advantage in the achieved graph sizes over [2].

Besides, we also solved the exact model for both versions of the aforementioned D-Wave 2000Q chips with 7 and 17 broken vertices, respectively. Not surprisingly in accordance with the results of Table 1a, we were able to find an embedding for the complete graph with 64 vertices in both cases.

For the unsolved instances, we use the current best solution SCIP provides at the timeout, being a proven lower bound on the actual optimum. As SCIP is a MIP solver, it tries to solve a given model to proven optimality and thus is not made for calculating fast approximate solutions. Therefore, the found solution values for instances with increasing Chimera sizes and ratios decrease significantly, due to the sizes of the models. The instance combinations (32, 0.2) and (34, 0.2) could not be solved at all

Table 2 Difference of rounded averaged solution ratios from heuristic models

b \ s	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
0.005	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.01	−0.12	−0.13	−0.14
0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.06	−0.17	−0.25	−0.23	−0.21	−0.24	−0.23
0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	−0.24	−0.22	−0.19	−0.19	−0.17	−0.08	−0.10
0.04	0.00	0.00	0.00	0.00	0.00	0.01	0.01	−0.16	−0.14	−0.16	−0.14	−0.08	−0.06	−0.05	−0.04	−0.01
0.05	0.00	0.00	0.00	0.00	0.02	0.04	−0.10	−0.11	−0.08	−0.06	−0.05	0.00	0.04	0.06	0.05	0.06
0.1	0.00	0.03	0.09	0.14	0.20	0.16	0.06	0.12	0.10	0.13	0.16	0.15	0.17	0.15	0.18	0.15
0.2	0.06	0.17	0.24	0.25	0.29	0.27	0.25	0.19	0.15	0.15	0.12	0.11	0.11	0.08	0.09	0.08

Colours indicate advantage of a certain model (green: $m = 0.25$, blue: $m = 0.0$)

with the exact model because SCIP ran out of memory. Nevertheless, the remaining instances provide values comparable to those of [2].

Evaluating the heuristic approach, we can see that Table 1b for $m = 0.25$ does not differ significantly from the one for the exact model according to solvability. Having a closer look at the values, we see no decrease for the solved instances. However, there is a slight improvement for the unsolved instances. Thus, the model has a slight advantage regarding runtime but still seems to yield close to optimal values.

Regarding the heuristic approach with $m = 0$, shown in Table 1c, we have a much stronger difference to the exact model. A lot more instances could be solved within one hour of computation time, especially those with larger ratios of broken vertices. For the combination of sizes above 18 and ratios between about 0.02 and 0.05, we see a clear improvement through the heuristic, although a few instances could not be solved, too. This region of parameters, where it is reasonable to use this heuristic, is also clearly recognizable in Table 2, where we show the advantage of the heuristics with either $m = 0$ or $m = 0.25$.

However, in Table 1c we observe that the proportions of graph sizes are much smaller for ratios above 0.1 than for $m = 0.25$. Thus, this heuristic model seems to get easier again with an increasing ratio of broken vertices. We assume a significant number of crossroads is excluded in advance, because of the large number of broken vertices, such that the resulting model has only very few solutions left. In these cases, the heuristic with $m = 0$ is much too restrictive and $m = 0.25$ is advantageous.

In order to compare our approach to previous work by Boothby et al. [2], we similarly plot the found graph sizes for selected ratios of broken vertices in Fig. 9. For larger ratios of broken vertices, e.g. $b = 0.1$ and $b = 0.05$, the maximum over the found graph sizes is comparable to [2] for both $m = 0.0$ and $m = 0.25$. In contrast, for smaller ratios of broken vertices, e.g. $b = 0.01$ and $b = 0.02$ our heuristic approach with $m = 0.0$ is able to embed larger complete graphs than it was reported in [2]. Note that the diagonal corresponds to representing the largest possible complete graph size.

All in all, for a ratio of 0.05 or smaller we observe that the proportions from the solved instances with small size and ratio have a value very close to 1.0, meaning most of them yield a maximal or close to maximal complete graph despite the presence of broken vertices. Due to the heuristic results, we expect just a very small decline in the proportions for the exact model for larger Chimera sizes, if we could solve them to the

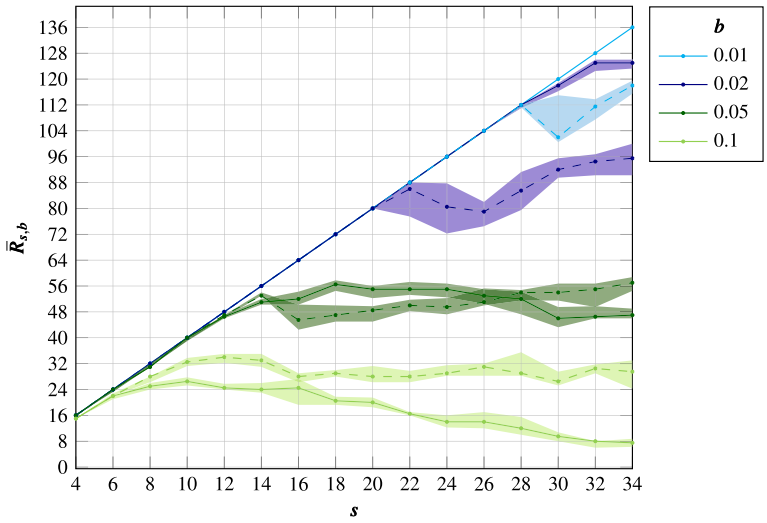


Fig. 9 Complete graph sizes against Chimera sizes s for selected ratios of broken vertices b for our heuristic approach (H). The median of $G_{s,b,i}$ over $i \in [10]$ is depicted by the solid lines for $m = 0.0$ and dashed lines for $m = 0.25$; the shaped regions illustrate the quartiles

end. This is based on the fact that the heuristics provide a lower bound on the actual optimum of the exact model. Thus, despite the shortcomings presented in Sect. 4.4, our model is indeed very powerful.

8 Conclusion

We introduced a novel approach for the problem of embedding a complete graph into a faulty Chimera hardware architecture. It is based on a formulation as a bipartite matching optimization problem with additional constraints. We could show by a detailed analysis that the problem is fixed parameter tractable, where the decisive parameter is the number of broken vertices. The formulated optimization problem (E) can be solved to optimality using state-of-the-art MIP solvers for small Chimera sizes or a small ratio of broken vertices. Especially in these parameter settings, the optimal value of the heuristic version (H) does not differ significantly from the original one. For larger Chimera graphs with a ratio of broken vertices in a certain range, the heuristic performs even better within the given time constraint of one hour, due to the removal of unlikely crossroads and thus several constraints. However, if the ratio is too large, here above 0.1, the heuristic is too restrictive and the solution quality decreases again. Nevertheless, the complete graph sizes we have found exceed the ones from previous approaches [2,13].

Further, regarding the current developments in the area of quantum annealers, larger graphs with less broken vertices, and operational times of over 1 year we can produce reusable templates for complete graphs with a reasonable computational power. Nev-

ertheless, there is some space for improvements. By exploiting the full potential of the branching strategy shown in Sect. 4.2 using dedicated bounding techniques, we could develop a more customized, exact or heuristic, solver.

Another step is to transfer the approach to the just recently released new hardware graph Pegasus. It yields a larger connectivity for the same number of vertices, but at the same time this makes the Pegasus graph less approachable. The shown constructions for the Chimera graph provide a deeper insight into the structure of such lattice-like graphs and the problems dealing with them. Observing the physical realization by specifically arranged overlapping loops, one can see that the Pegasus graph is closely related to the Chimera [5]. Thus, we are confident that our model construction for the Chimera can be transferred to the Pegasus topology. Due to the larger vertex degree we even expect less constraints resulting from broken vertex pairs than for the Chimera.

Acknowledgements The authors gratefully acknowledge the Jülich Supercomputing Centre (JSC) for supporting this work by providing the access to the D-Wave 2000Q through the Jülich UNified Infrastructure for Quantum computing (JUNIQ).

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Boothby, K., Bunyk, P., Raymond, J., Roy, A.: Next-generation topology of D-Wave quantum processors. arXiv:2003.00133 (2020)
2. Boothby, T., King, A.D., Roy, A.: Fast clique minor generation in chimera qubit connectivity graphs. Quantum Inf. Process. **15**(1), 495–508 (2016). https://doi.org/10.1007/s11128-015-1150-6
3. Cai, J., Macready, W.G., Roy, A.: A practical heuristic for finding graph minors. arXiv:1406.2741 (2014)
4. Choi, V.: Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design. Quantum Inf. Process. **10**(3), 343–353 (2011). https://doi.org/10.1007/s11128-010-0200-3
5. D-Wave Systems Inc.: D-Wave Systems documentation—D-Wave QPU architecture: Topologies. https://docs.dwavesys.com/docs/latest/c_gs_4.html. 2020-12-03
6. D-Wave Systems Inc.: Technical description of the D-Wave quantum processing unit. https://docs.dwavesys.com/docs/latest/_downloads/09-1109A-V_Technical_Description_of_DW_QPU.pdf. User Manual 2020-10-06
7. D-Wave Systems Inc.: Minorminer. GitHub repository (2020). Version 0.2.4. https://github.com/dwavesystems/minorminer
8. Gleixner, Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R.L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M.E., Maher, S.J., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J.M., Walter, M., Wegscheider, F., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 6.0. Technical report, Optimization Online (2018). https://optimization-online.org/DB_HTML/2018/07/6692.html

9. Goodrich, T.D., Sullivan, B.D., Humble, T.S.: Optimizing adiabatic quantum program compilation using a graph-theoretic framework. Quantum Inf. Process. **17**(5), 118 (2018). https://doi.org/10.1007/s11128-018-1863-4
10. Hamilton, K.E., Humble, T.S.: Identifying the minor set cover of dense connected bipartite graphs via random matching edge sets. Quantum Inf. Process. **16**(4), 94 (2017). https://doi.org/10.1007/s11128-016-1513-7
11. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**(4), 225–231 (1973). https://doi.org/10.1137/0202019
12. Jünger, M., Lobe, E., Mutzel, P., Reinelt, G., Rendl, F., Rinaldi, G., Stollenwerk, T.: Performance of a quantum annealer for ising ground state computations on chimera graphs. arXiv:1904.11965 (2019)
13. Klymko, C., Sullivan, B.D., Humble, T.S.: Adiabatic quantum programming: minor embedding with hard faults. Quantum Inf. Process. **13**(3), 709–729 (2014). https://doi.org/10.1007/s11128-013-0683-9
14. Maher, S., Miltenberger, M., Pedroso, J.P., Rehfeldt, D., Schwarz, R., Serrano, F.: PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In: Mathematical Software—ICMS 2016, pp. 301–307. Springer International Publishing (2016). https://doi.org/10.1007/978-3-319-42432-3_37
15. Pinilla, J.P., Wilton, S.J.: Layout-aware embedding for quantum annealing processors. In: International Conference on High Performance Computing, pp. 121–139. Springer, Berlin (2019). https://doi.org/10.1007/978-3-030-20656-7_7
16. Rieffel, E.G., Venturelli, D., O'Gorman, B., Do, M.B., Prystay, E.M., Smelyanskiy, V.N.: A case study in programming a quantum annealer for hard operational planning problems. Quantum Inf. Process. **14**(1), 1–36 (2015). https://doi.org/10.1007/s11128-014-0892-x
17. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. J. Combin. Theory Ser. B **63**(1), 65–110 (1995). https://doi.org/10.1006/jctb.1995.1006
18. Serra, T., Huang, T., Raghunathan, A., Bergman, D.: Template-based minor embedding for adiabatic quantum optimization. arXiv:1910.02179 (2019)
19. Stollenwerk, T., Lobe, E., Jung, M.: Flight gate assignment with a quantum annealer. In: International Workshop on Quantum Technology and Optimization Problems, pp. 99–110. Springer, Berlin (2019). https://doi.org/10.1007/978-3-030-14082-3_9
20. Stollenwerk, T., O'Gorman, B., Venturelli, D., Mandrà, S., Rodionova, O., Ng, H., Sridhar, B., Rieffel, E.G., Biswas, R.: Quantum annealing applied to de-conflicting optimal trajectories for air traffic management. IEEE Trans. Intell. Transp. Syst. **21**(1), 285–297 (2019). https://doi.org/10.1109/TITS.2019.2891235
21. Tanimoto, S.L., Itai, A., Rodeh, M.: Some matching problems for bipartite graphs. J. ACM (JACM) **25**(4), 517–525 (1978). https://doi.org/10.1145/322092.322093
22. Venturelli, D., Marchand, D.J., Rojo, G.: Quantum annealing implementation of job-shop scheduling. arXiv:1506.08479 (2015)
23. Zaribafiyan, A., Marchand, D.J., Rezaei, S.S.C.: Systematic and deterministic graph minor embedding for cartesian products of graphs. Quantum Inf. Process. **16**(5), 136 (2017). https://doi.org/10.1007/s11128-017-1569-z
24. Zbinden, S., Bärtschi, A., Djidjev, H., Eidenbenz, S.: Embedding algorithms for quantum annealers with chimera and pegasus connection topologies. In: International Conference on High Performance Computing, pp. 187–206. Springer, Berlin (2020). https://doi.org/10.1007/978-3-030-50743-5_10