Pegasus: The second connectivity graph for large-scale quantum annealing hardware

Nike Dattani[*]
Harvard-Smithsonian Center for Astrophysics and National Research Council of Canada

Szilard Szalay[†]
Wigner Research Centre for Physics

Nicholas Chancellor[‡]
Joint Quantum Centre, Durham University

Pegasus is a graph which offers substantially increased connectivity between the qubits of quantum annealing hardware compared to the graph Chimera. It is the first fundamental change in the connectivity graph of quantum annealers built by D-Wave since Chimera was introduced in 2009 and then used in 2011 for D-Wave's first commercial quantum annealer. In this article we describe an algorithm which defines the connectivity of Pegasus and we provide what we believe to be the best way to graphically visualize Pegasus in order to see which qubits couple to each other. As supplemental material, we provide a wide variety of different visualizations of Pegasus which expose different properties of the graph in different ways. We provide an open source code for generating the many depictions of Pegasus that we show.

The 128 qubits of the first commercial quantum annealer (D-Wave One, released in 2011) were connected by a graph called Chimera (first defined publicly in 2009 [1]), which is rather easy to describe: A 2D array of $K_{4,4}$ graphs, with one 'side' of each $K_{4,4}$ being connected to the same corresponding side on the $K_{4,4}$ cells directly above and below it, and the other side being connected to the same corresponding side on the $K_{4,4}$ cells to the right and left of it (see Figure 1). The qubits can couple to up to 6 other qubits, since each qubit couples to 4 qubits within its $K_{4,4}$ unit cell, and to 2 qubits the $K_{4,4}$ cells above and below it (or to the left and right of it). All commercial quantum annealers built to date follow this graph connectivity, with just larger and larger numbers of $K_{4,4}$ cells (See Table 1).

Table I: Chimera graphs in all commercial quantum annealers to date.

	Array of $K_{4,4}$ cells	Total # of qubits
D-Wave One	4×4	128
D-Wave Two	8×8	512
D-Wave 2X	12×12	1152
D-Wave 2000Q	16×16	2048

In 2018, D-Wave announced the construction of a (not yet commercial) quantum annealer with a greater connectivity than Chimera offers, and a program (NetworkX) which allows users to generate certain Pegasus graphs. However, no explicit description of the graph connectivity in Pegasus has been published yet, so we have had to apply the process of reverse engineering to determine it, and the following section describes the algorithm we have established for generating Pegasus.

I. ALGORITHM FOR GENERATING PEGASUS

A. The vertices (qubits)

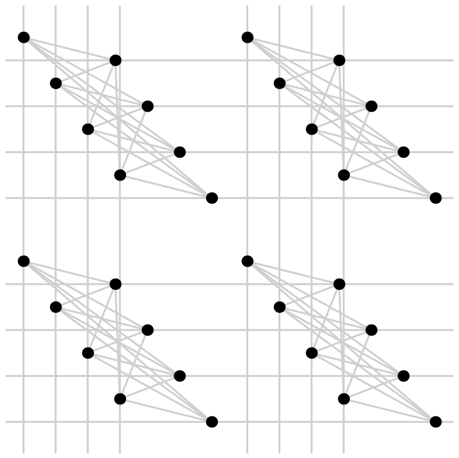
Start with Z layers of Chimera graphs, each being an $X \times Y$ array of $K_{4,4}$ cells (therefore we have an $X \times Y \times Z$ array of $K_{4,4}$ cells). The indices (x, y, z) will be used to describe the location of each cell along the indices corresponding to the dimension picked from (X, Y, Z). The values of X and Y have no restriction, but $Z = 3$ in Pegasus. Each $K_{4,4}$ cell has two 'sides', labeled $i \in \{0, 1\}$, so that there are 4 qubits (vertices) for every combination: (x, y, z, i). We will arbitrarily label these 4 qubits using two more labels: $j \in \{0, 1\}$ and $k \in \{0, 1\}$. Therefore every qubit is associated with 6 indices: (x, y, z, i, j, k), with their ranges and descriptions given in Table II.

In all of the figures in this publication, the origin will be in the bottom-left corner, the x index will increase in the right-hand direction, the y index will increase in the upward direction, and the z index (which indicates which Chimera layer is being considered) will increase in the direction going upward and rightward simultaneously (since paper and computer screens are still two-dimensional), see Figure 2a. Then $i = 0$ will represent the left side in the classic $K_{4,4}$ depiction, or the horizontal vertices in the diamond-shaped or triangle-shaped depictions, while $i = 1$ will represent the right side in the classic and vertical vertices in the diamond and triangle.

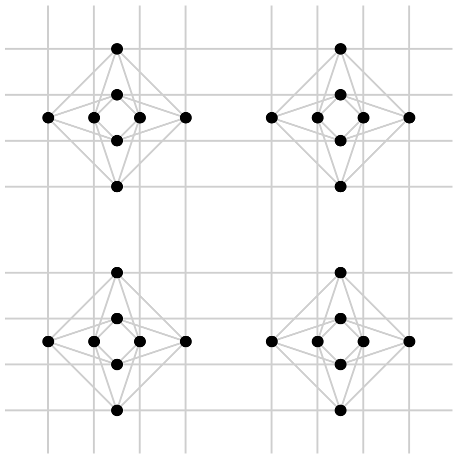
* n.dattani@cfa.harvard.edu
† szalay.szilard@wigner.mta.hu
‡ nicholas.chancellor@durham.ac.uk

Figure 1: Three different depictions of the Chimera graph, with open edges to show that the pattern repeats.

(a) Tilted classic



(b) Diamond



(c) Upper triangle

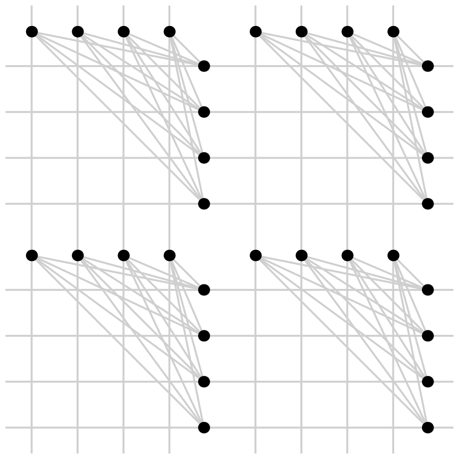
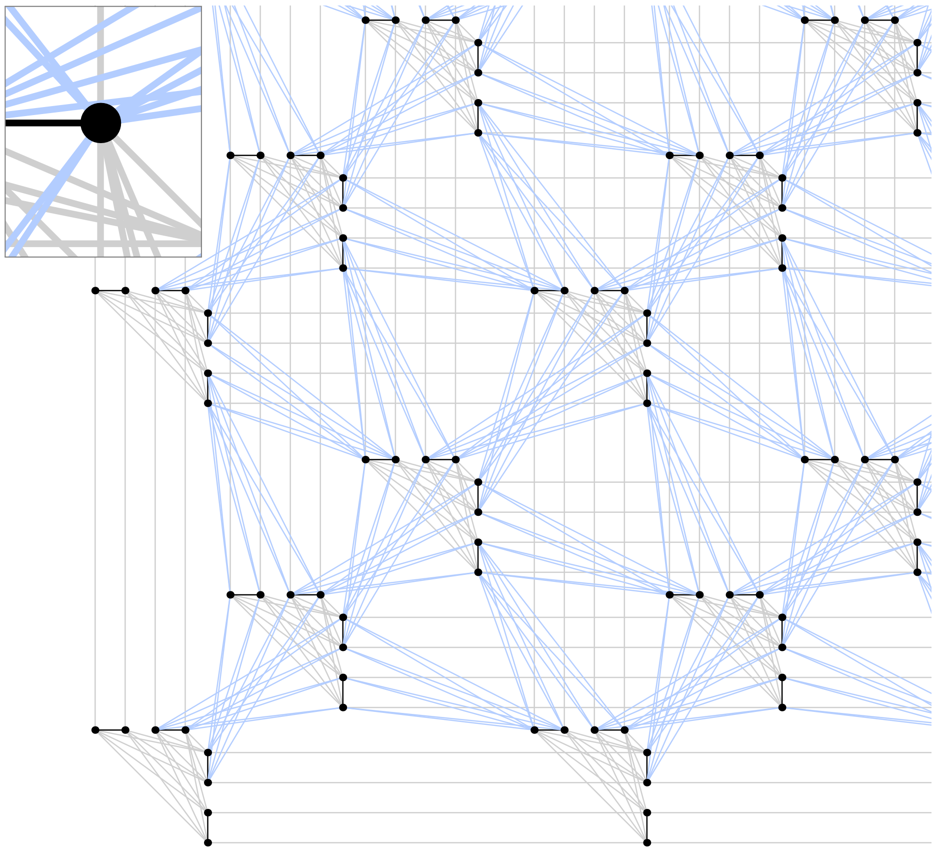
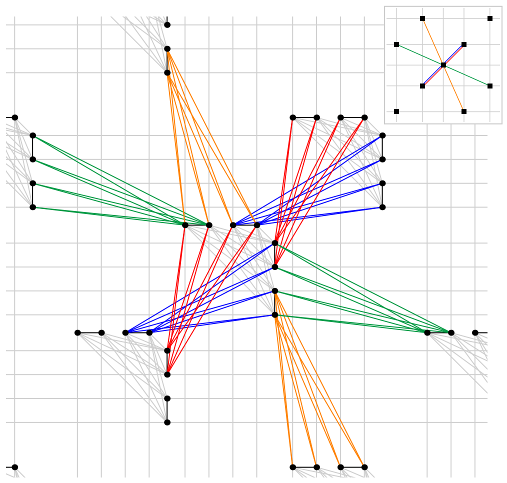


Figure 2: The Pegasus graph.

(a) A patch of size $(X, Y, Z) = (2, 2, 3)$ cropped out of a Pegasus graph. Grey edges are part of the three layers of Chimera graphs, while black and blue edges form the remainder of the Pegasus graph (the latter connecting different layers).



(b) Groups of $K_{2,4}$ edges connecting cells belonging to different Chimera layers, colored according to Eqs. (12)-(19).



(c) "Compressed" version of the Pegasus graph, with each cell represented by one vertex and each $K_{2,4}$ (8 edges) represented by one edge.

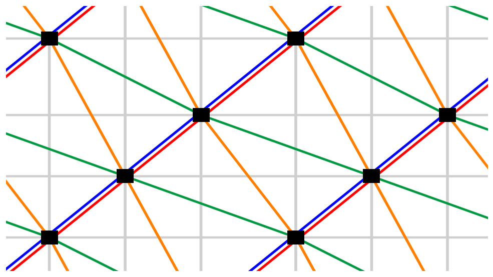


Table II: Indices used to describe each qubit (vertex) in Pegasus.

Index	Range	Description
x	0 to $X-1$	Row within a Chimera layer
y	0 to $Y-1$	Column within a Chimera layer
z	0 to 2	Chimera layer
i	$0, 1$	Side within $K_{4,4}$
j	$0, 1$	First index within each side of $K_{4,4}$
k	$0, 1$	Second index within each side of $K_{4,4}$

B. The edges (couplings) in Chimera

1. *Edges forming each $K_{4,4}$ cell*

The $K_{4,4}$ cells are given by:

$$(x,y,z,0,j,k) \longleftrightarrow (x,y,z,1,j',k'). \tag{1}$$

This means for each $K_{4,4}$ cell, all vertices (j,k) for side $i=0$ are coupled to all vertices (j',k') of side $i=1$. ***For this entire publication, j' can be equal to or different from j (and likewise for $k'$and k).***

2. *Edges connecting different $K_{4,4}$ cells*

The horizontal lines between $K_{4,4}$ cells in Figure 1 can be described by adding 1 to x while keeping all other variables constant and setting $i=1$:

$$(x,y,z,1,j,k) \longleftrightarrow (x+1,y,z,1,j,k), \tag{2}$$

and the vertical lines can be described by adding 1 to y while keeping all other variables constant and setting $i=0$:

$$(x,y,z,0,j,k) \longleftrightarrow (x,y+1,z,0,j,k). \tag{3}$$

For each z, Equations (1)-(3) define edges connecting vertices labeled by x, y, i, j and k. This completes the definition of a Chimera graph. In all figures, these Chimera edges are grey.

C. The *new* edges (couplings) in Pegasus

1. *New edges added to each $K_{4,4}$ cell:*

Pegasus first adds connections within each $K_{4,4}$ cell, given by simply coupling each vertex labeled as $k=0$ to its $k=1$ counterpart with all other variables unchanged:

$$(x,y,z,i,j,0) \longleftrightarrow (x,y,z,i,j,1). \tag{4}$$

These edges are drawn black in the figures.

2. *Edges connecting different $K_{4,4}$ cells:*

The rest of the new connections in Pegasus come from connecting the $K_{4,4}$ cells between different layers (different z) of Chimera graphs. The qubits of a $K_{4,4}$ cell located at coordinates (x,y,z) will be connected to 6 different $K_{4,4}$ cells on the other Chimera layers, with 64 edges in the form of 8 different $K_{2,4}$ graphs: 1 $K_{2,4}$ graph (8 edges) for each of 4 different connecting $K_{4,4}$ cells, and 2 $K_{2,4}$ graphs (16 edges) for the other 2 connecting $K_{4,4}$ cells.

Figure 2b shows these 64 edges, and the 8 groups of 8-edge $K_{2,4}$ graphs connecting the central cell to 6 others are shown with 4 different colors. This is because we have found a set of 4 convenient rules that can be repeated to generate the entire Pegasus graph, and it is found that 4 of the $K_{2,4}$ graphs can be generated by applying these 4 rules to the central cell, and the other 4 of the $K_{2,4}$ graphs can be generated by applying these 4 rules to other $K_{4,4}$ cells (or by applying the 4 rules to the central cell again, but in reverse). We will now describe the rules.

First, all edges are between vertices of one $K_{4,4}$ side i and its complementary side $\bar{i}$ in a different $K_{4,4}$ (so $i=0$ vertices are coupled to $i=1$ vertices of a $K_{4,4}$ in a different layer). In fact all connections will be of the form $(i,j,k) \longleftrightarrow (\bar{i},j',k')$ where j' and k' can be any value in $\{0,1\}$. ***All edges can actually be described using just a one-line rule:***

$$\begin{gathered}(x,y,z,i,j,k)\\ \longleftrightarrow\\ \left(x-j\bar{i}+\delta_{z2},y-ji+\delta_{z2},(z+1)\bmod 3,\bar{i},j',k'\right).\end{gathered} \tag{5}$$

For the layers labeled $z=0$ or 1 (meaning that $\delta_{z2}=0$), we have:

$$(x,y,z,i,j,k) \longleftrightarrow (x-j\bar{i},y-ji,z+1,\bar{i},j',k'). \tag{6}$$

Substituting $j=0$ into Eq. (6) tells us that all $j=0$ vertices of a $K_{4,4}$ cell are connected to all vertices of the opposite side $\bar{i}$ in the $K_{4,4}$ cell with the same (x,y), but next layer $z+1$:

$$(x,y,z) \longleftrightarrow (x,y,z+1), \text{ for } z \in \{0,1\}, j=0. \tag{7}$$

Substituting $j=1$ into Eq. (6) tells us that all $j=1$ vertices are also connected to all j' and k' vertices in the opposite side $\bar{i}$ in a $K_{4,4}$ cell in the next layer $z+1$, but with x and y coordinates shifted by 1 in the following way:

$$(x,y,z) \longleftrightarrow (x-\bar{i},y-i,z+1) \text{ for } z \in \{0,1\}, j=1. \tag{8}$$

For the $z=2$ layer ($\delta_{z2}=1$), we have from Eq. (5):

$$(x,y,2,i,j,k) \longleftrightarrow (x-j\bar{i}+1,y-ji+1,0,\bar{i},j',k'). \tag{9}$$

Substituting $j=0$ into Eq. (9) tells us that all $j=0$ vertices of a $K_{4,4}$ cell are connected to the $K_{4,4}$ cells in the $z=0$ layer, but with x and y coordinates *both* shifted by 1:

$$(x, y, 2) \longleftrightarrow (x+1, y+1, 0), \text{ for } j = 0. \quad (10)$$

Substituting $j = 1$ into Eq. (9) tells us that all $j = 1$ vertices are connected in the following way:

$$(x, y, 2) \longleftrightarrow (x+i, y+\bar{i}, 0), \text{ for } j = 1. \quad (11)$$

We now explicitly write down the 4 rules which lead to the grouping scheme depicted in Figure 2b (these rules are different depending on whether $z \in \{0,1\}$ or $z = 2$, so there are actually 8 of them):

$$(x, y, z, 0, 0, k) \leftrightarrow (x, y, z+1, 1, j', k'), \quad z \in \{0,1\} \quad (12)$$
$$(x, y, z, 1, 0, k) \leftrightarrow (x, y, z+1, 0, j', k'), \quad z \in \{0,1\} \quad (13)$$
$$(x, y, z, 0, 1, k) \leftrightarrow (x-1, y, z+1, 1, j', k'), \quad z \in \{0,1\} \quad (14)$$
$$(x, y, z, 1, 1, k) \leftrightarrow (x, y-1, z+1, 0, j', k'), \quad z \in \{0,1\} \quad (15)$$

$$(x, y, 2, 0, 0, k) \leftrightarrow (x+1, y+1, 0, 1, j', k'), \quad (16)$$
$$(x, y, 2, 1, 0, k) \leftrightarrow (x+1, y+1, 0, 0, j', k'), \quad (17)$$
$$(x, y, 2, 0, 1, k) \leftrightarrow (x, y+1, 0, 0, j', k'), \quad (18)$$
$$(x, y, 2, 1, 1, k) \leftrightarrow (x+1, y, 0, 1, j', k'). \quad (19)$$

II. COMPARISON TO CHIMERA

A. Degree of the vertices

If we look for example at the cell at position $(x, y, z) = (1, 1, 1)$ in Fig. 2a, we can find that vertices have a degree of ***fifteen:*** The 6 grey edges that would regularly be in Chimera (4 to form the $K_{4,4}$ cell and 2 to connect to $K_{4,4}$ cells above/left and below/right), then there is 1 Pegasus edge added *within* one $K_{4,4}$ cell according to Eq. (4), then we have 8 more Pegasus edges for connecting $K_{4,4}$ cells of different Chimera layers z, from Eqs. (12)-(15) (or from Eqs. (16)-(19) if we were starting with a cell in the $z = 2$). Therefore the degree (which is 15) has increased by a factor of 2.5 when compared to the degree of Chimera (which is 6).

In Fig. 2a, the cells at the boundary, such as at position $(x, y, z) = (0, 0, 0)$, do not show a degree of 15, just as the cells of Chimera at the very edge would not show a degree of 6.

B. Non-planarity

We note that certain binary optimization problems forming planar graphs can be solved on a classical computer with a number of operations that scales polynomially with the number of binary variables, with the blossom algorithm [2]. Therefore it is important that the qubits of a quantum annealer are connected by a non-planar graph. The $K_{4,4}$ cells of Chimera are already sufficient to make all commercial D-Wave annealers non-planar. However, if each $K_{4,4}$ cell of a Chimera's physical qubits were to encode just one logical qubit (in for example, an extreme case of minor embedding), then Chimera would be planar. While all added edges in Pegasus that connect different Chimera layers are of the form $K_{2,4}$, which itself is planar; these $K_{2,4}$ edges connect cells of different planes of chimeras in a non-planar way, such that even if each cell were to represent one logical qubit, *these logical qubits would still form a non-planar graph in Pegasus.* This should expand the number of binary optimization problems that cannot yet be solved in polynomially time, that can potentially be embedded onto a D-Wave annealer.

C. Embedding

We have written an entire paper on the minor-embedding of quadratization gadgets onto Chimera and Pegasus. One highlight of that work is the fact that *all* quadratization gadgets for single cubic terms which require one auxiliary qubit, can be embedded onto Pegasus with no further auxiliary qubits because Pegasus contains K_4, which means that all three logical qubits and the auxiliary qubit can be connected in any way, without any minor-embedding. We refer the reader to that paper for more thorough details about the advantage of Pegasus over Chimera for the minor-embedding of quadratization gadgets.

III. OPEN SOURCE CODE FOR GENERATION OF PEGASUS FIGURES

All figures in this publication and its Supplemental Material can be generated (in vector graphic form) using our open source and customizable code which should be cited as Ref. [3]. The user can choose which type of Pegasus graph; the dimensions X and Y; the edge colors and widths, the vertex colors and widths; among other things (see the manual to Ref. [3] for details).

ACKNOWLEDGMENTS

We gratefully thank Kelly Boothby of D-Wave for her time spent in verifying the correctness of our algorithm. NC was funded by EPSRC (Project: EP/S00114X); SzSz by the NRDIF (NKFIH-K120569 and Quantum Technology National Excellence Program 2017-1.2.1-NKP-2017-00001) and the HAS (János Bolyai Research Scholarship and "Lendület" Program).

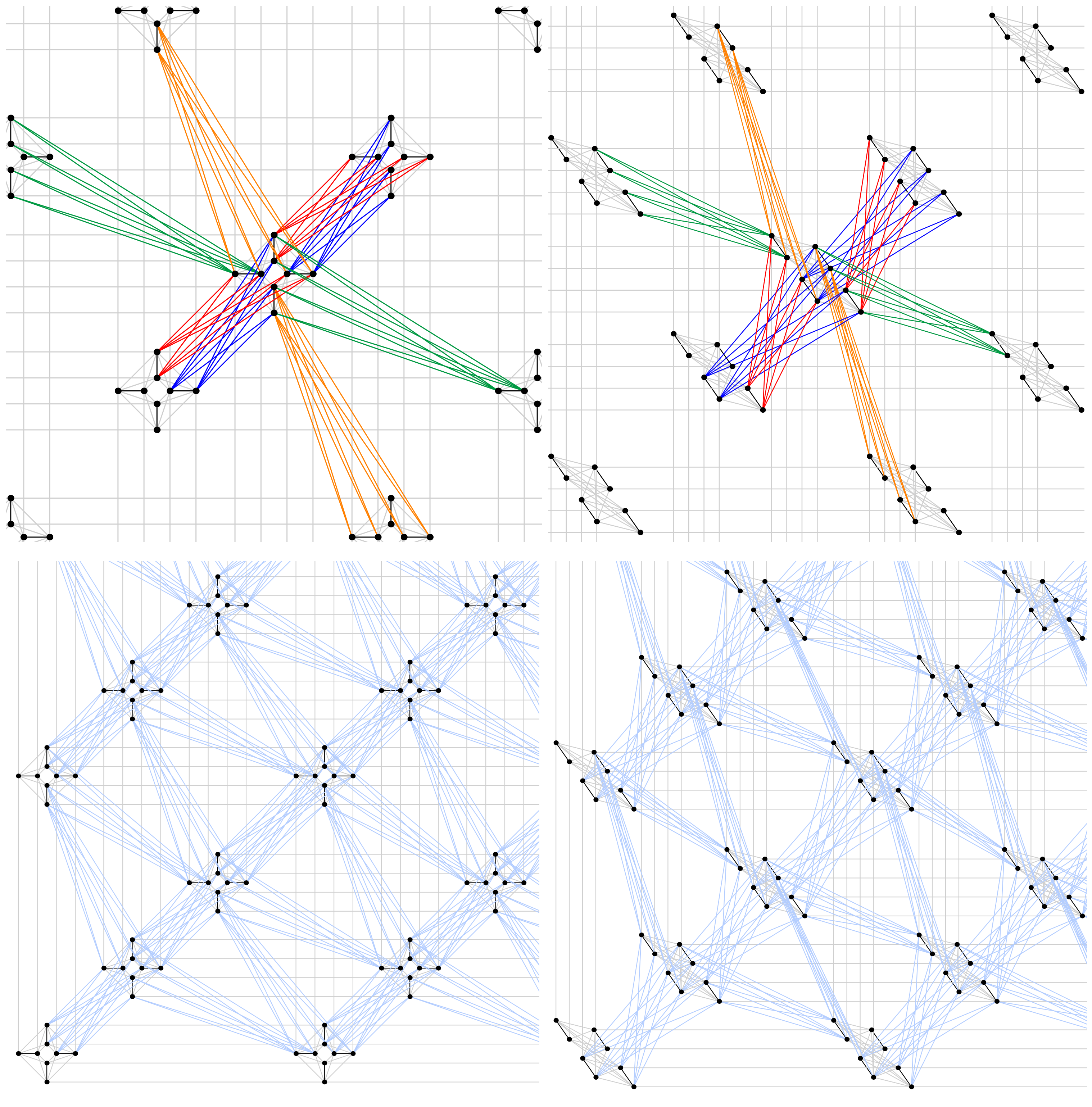
REFERENCES

[1] H. Neven, V. S. Denchev, M. Drew-Brook, J. Zhang, W. G. Macready, and G. Rose, *NIPS 2009 Demonstration: Binary Classification using Hardware Implementation of Quantum Annealing*, Tech. Rep. (2009).

[2] J. Edmonds, Canadian Journal of Mathematics **17**, 449 (1965); *Journal of Research of the National Bureau of Standards-B.*, Tech. Rep. 2 (1965).

[3] S. Szalay, N. Dattani, and N. Chancellor, (2018), https://github.com/HPQC-LABS/PegasusDraw, DOI: 10.5281/zenodo1953876.

Figure 3: Two more ways to visualize Pegasus, where the $K_{4,4}$ cells are simply drawn using the “diamond” and “tilted classic” depictions instead of the “upper-triangle” one.



Supplementary Material

Figure 1: $K_{4,4}$ cells (Chimera).

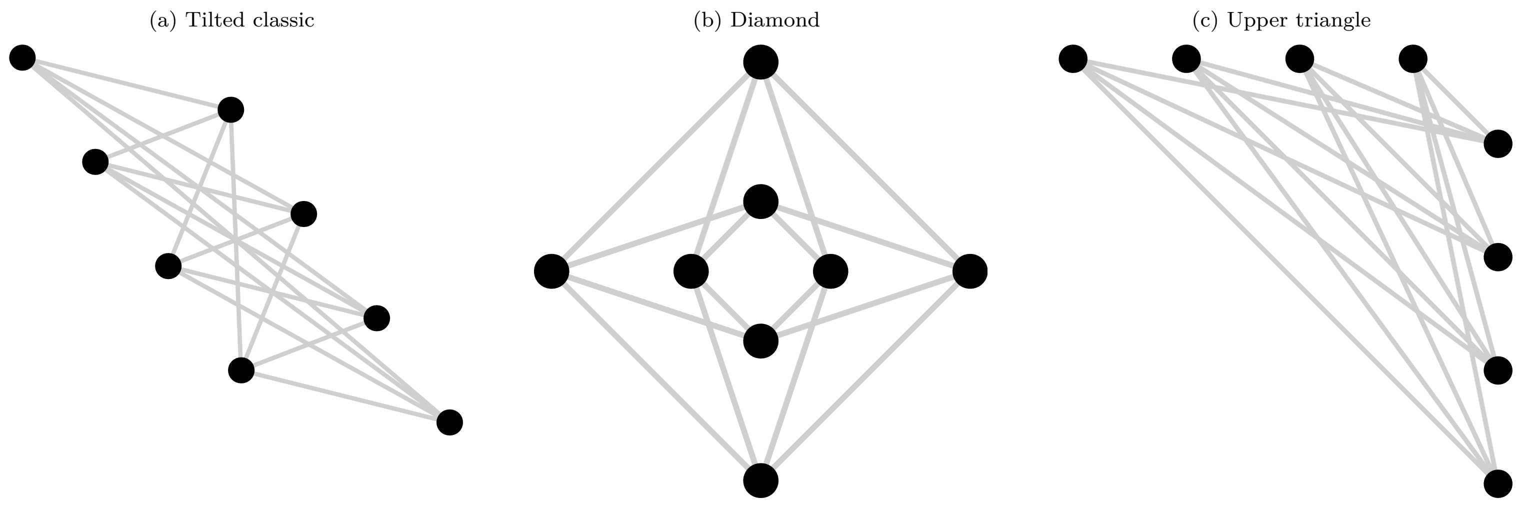


Figure 2: 2×2 arrays of $K_{4,4}$ cells in Chimera formation.

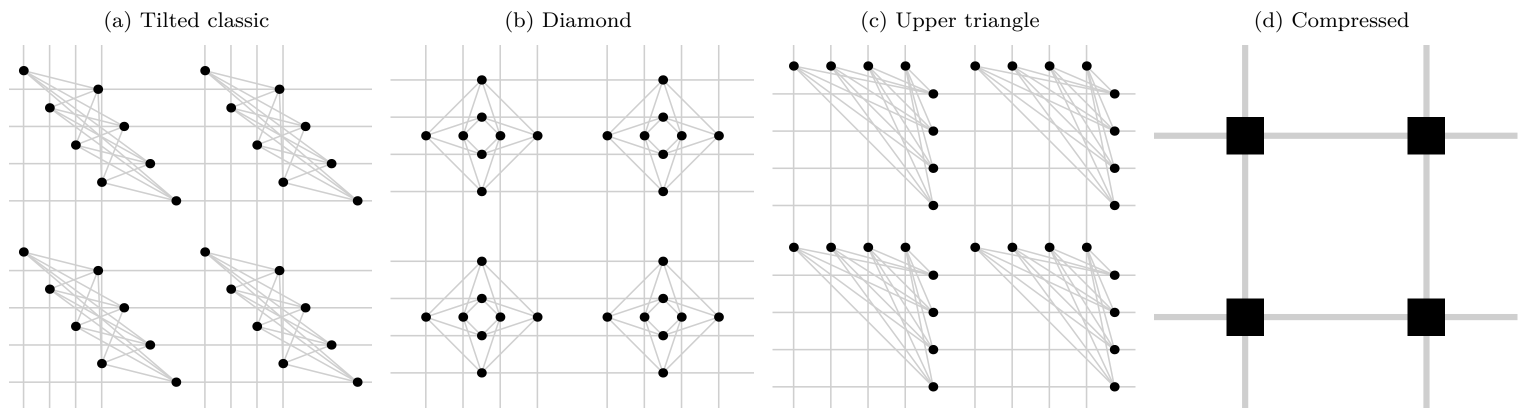


Figure 3: 5×5 arrays of $K_{4,4}$ cells in Chimera formation.

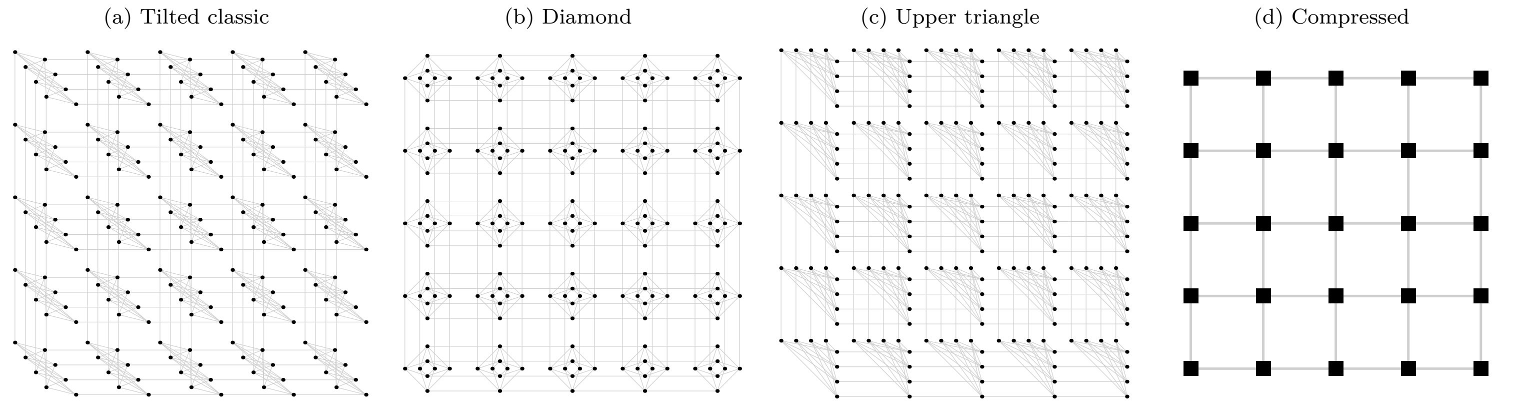


Figure 4: Pegasus cells.

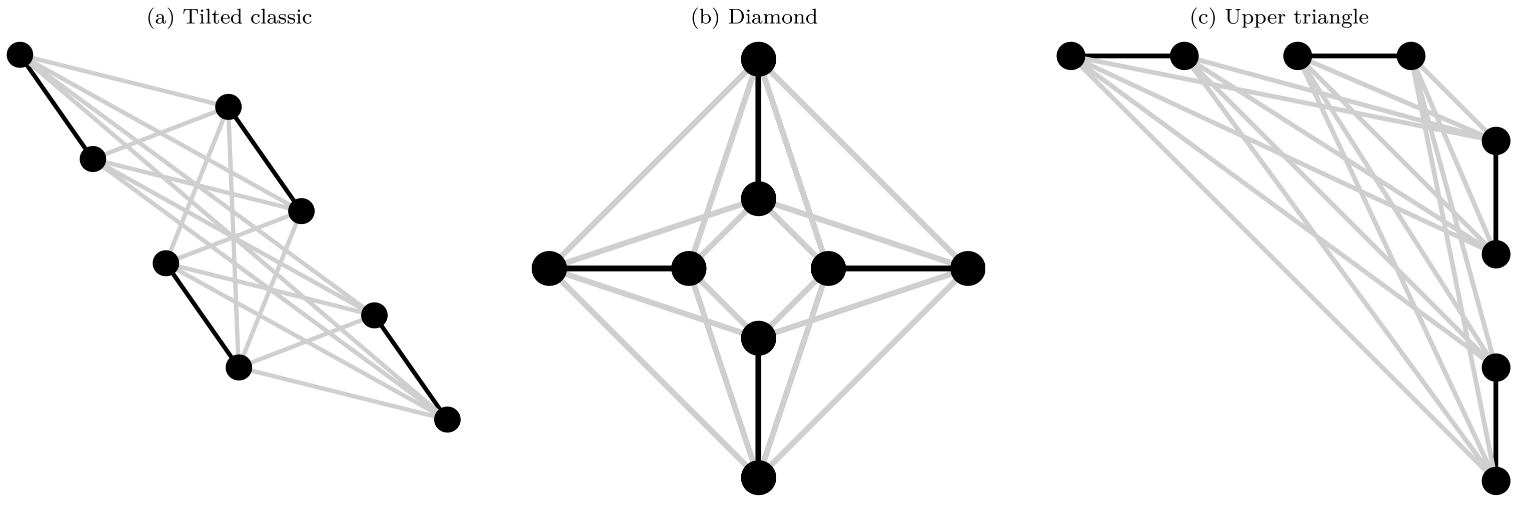


Figure 5: 64 inter-layer edges (Pegasus).

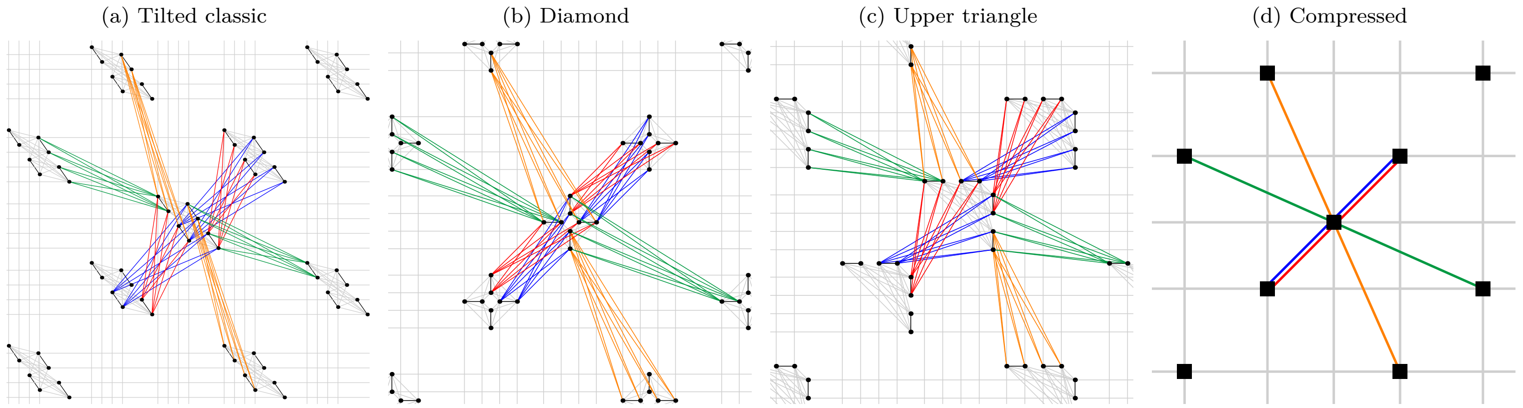


Figure 6: 64 inter-layer edges within an $(X, Y, Z) = (5, 5, 3)$ lattice (Pegasus).

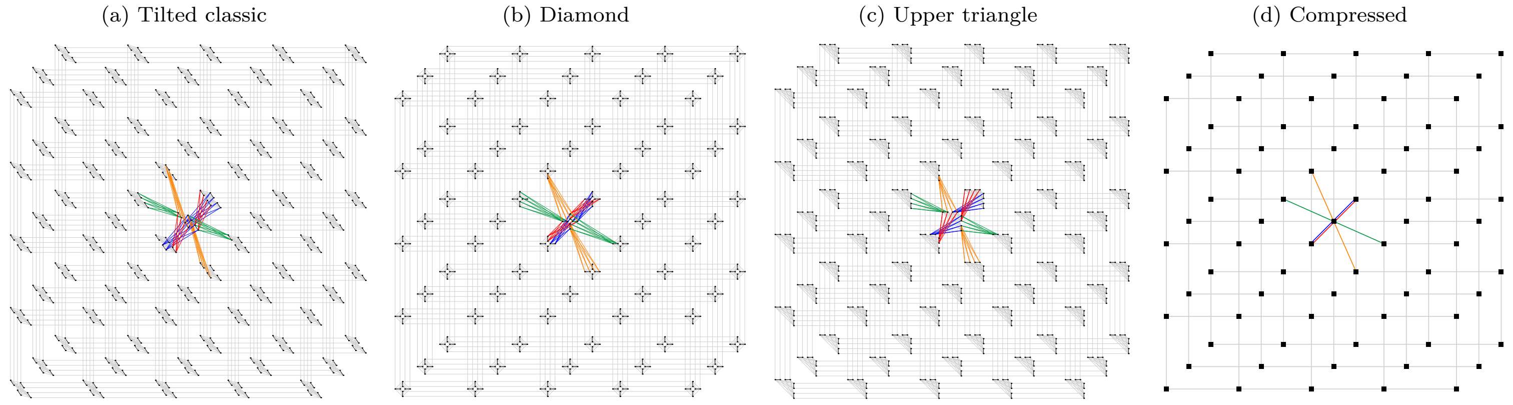
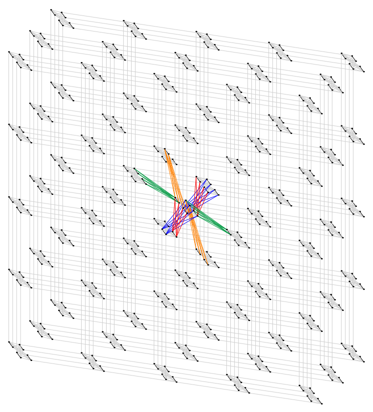
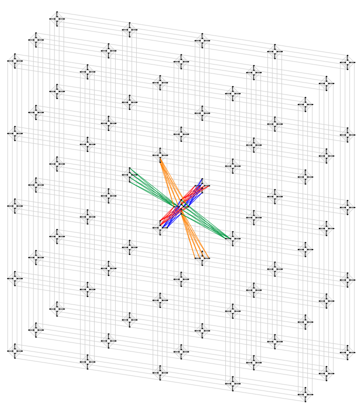


Figure 7: 64 inter-layer edges within a $(X, Y, Z) = (5, 5, 3)$ *tilted* lattice (Pegasus).

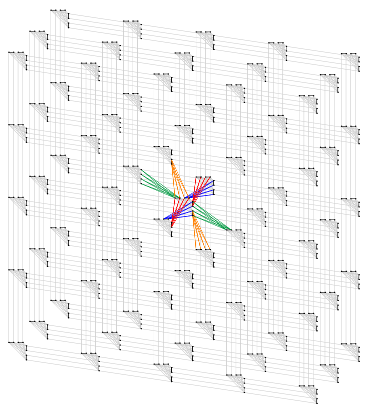
(a) Tilted classic



(b) Diamond



(c) Upper triangle



(d) Compressed

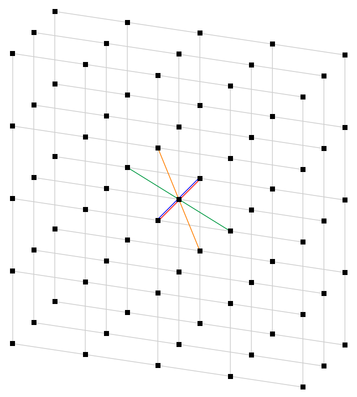
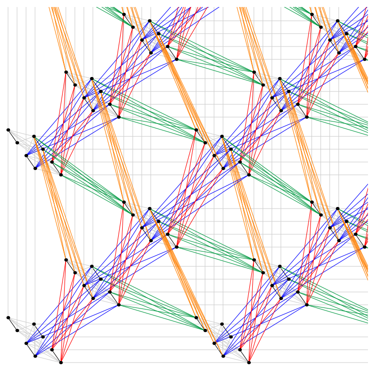
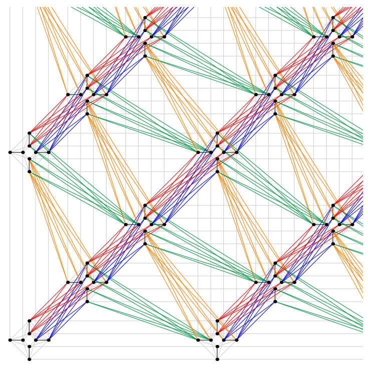


Figure 8: $(X, Y, Z) = (2, 2, 3)$ patch cropped out of Pegasus.

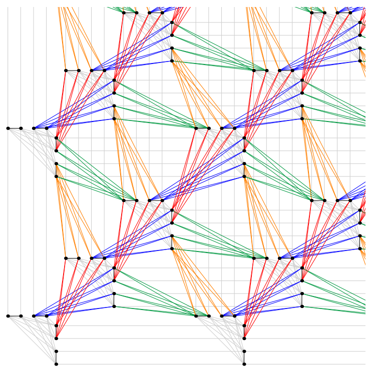
(a) Tilted classic



(b) Diamond



(c) Upper triangle



(d) Compressed

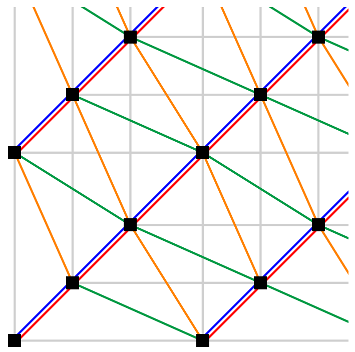
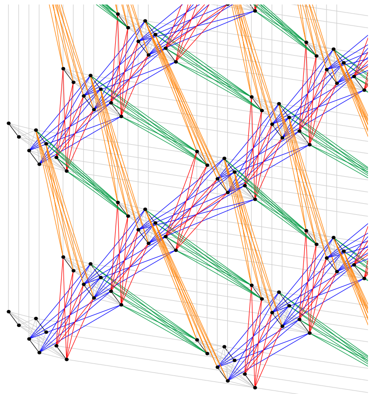
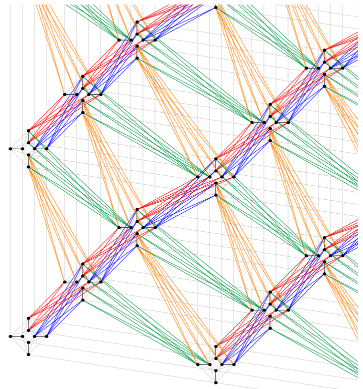


Figure 9: $(X, Y, Z) = (2, 2, 3)$ patch cropped out of a *tilted* Pegasus

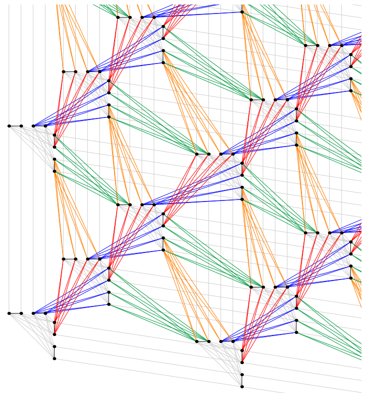
(a) Tilted classic



(b) Diamond



(c) Upper triangle



(d) Compressed

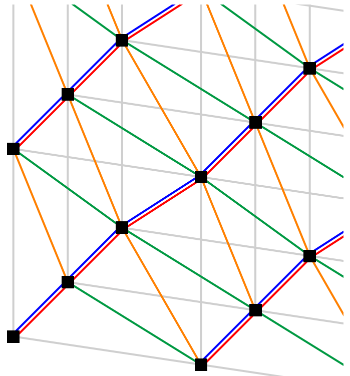


Figure 10: $(X, Y, Z) = (2, 2, 3)$ patch cropped out of Pegasus (with all Pegasus-only edges either black or light blue).

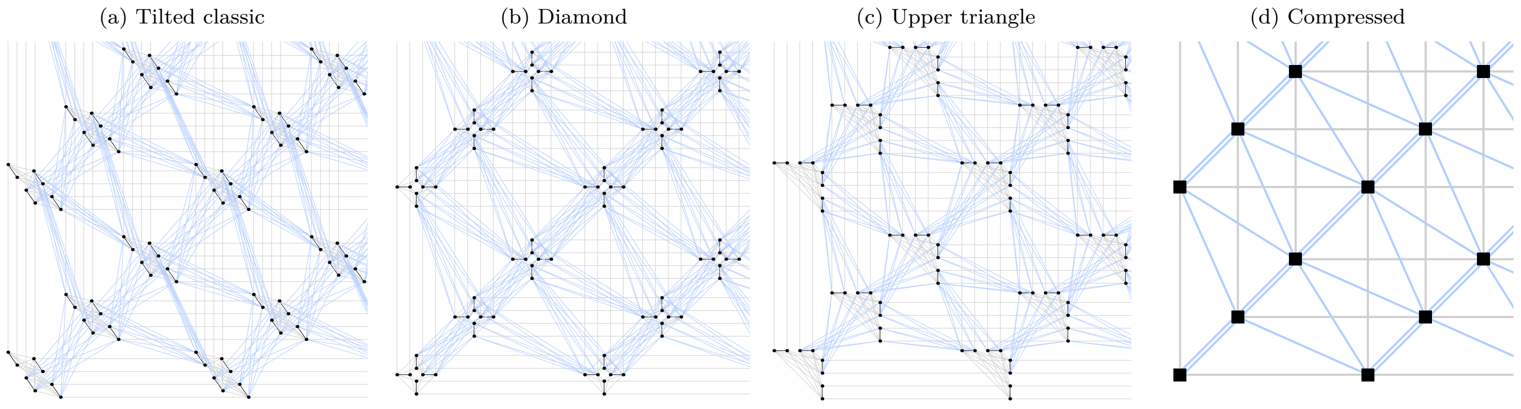


Figure 11: $(X, Y, Z) = (2, 2, 3)$ patch cropped out of *a tilted* Pegasus (with all Pegasus-only edges either black or light blue).

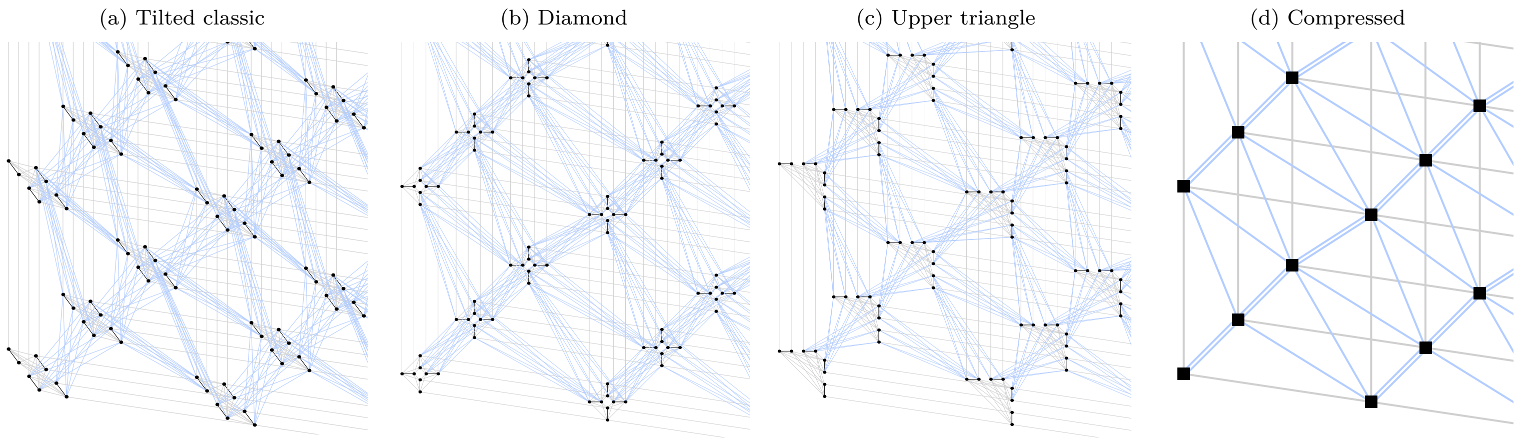


Figure 12: $(X, Y, Z) = (5, 5, 3)$ lattice of Pegasus.

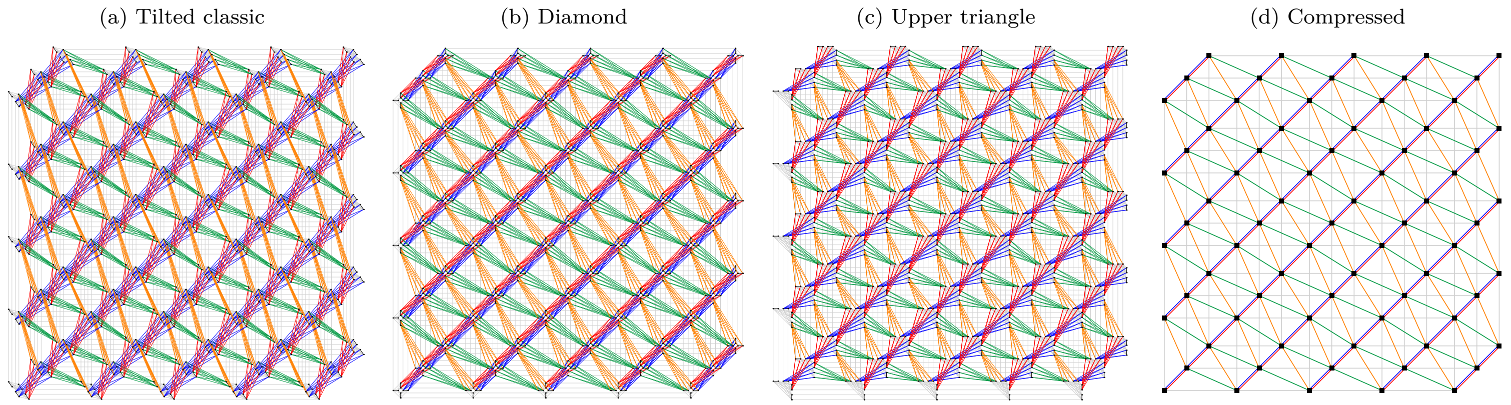
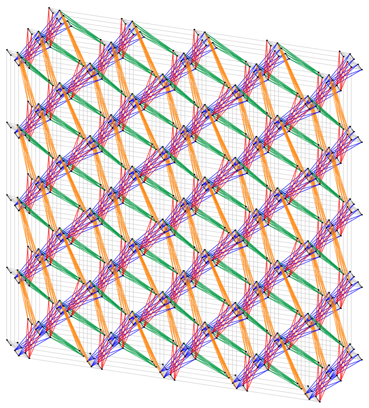
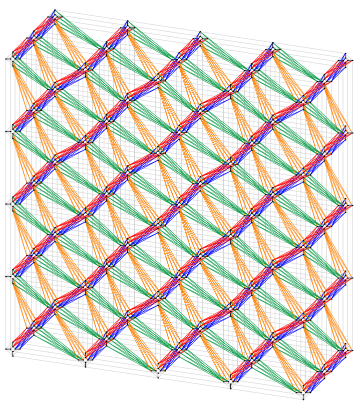
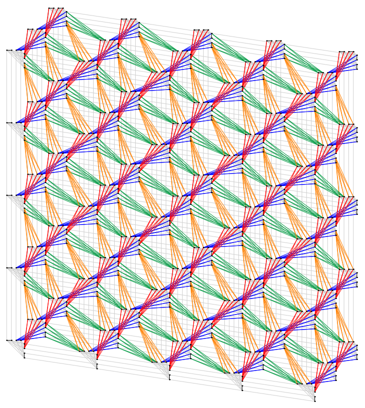


Figure 13: $(X, Y, Z) = (5, 5, 3)$ *tilted* lattice of Pegasus.


(a) Tilted classic


(b) Diamond


(c) Upper triangle

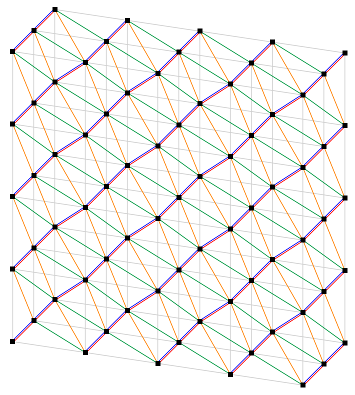
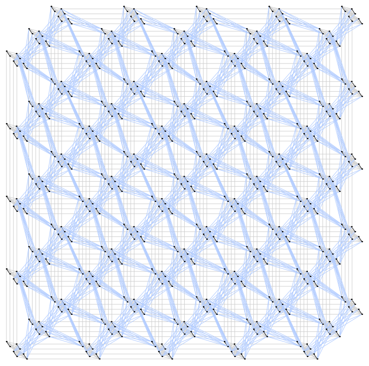
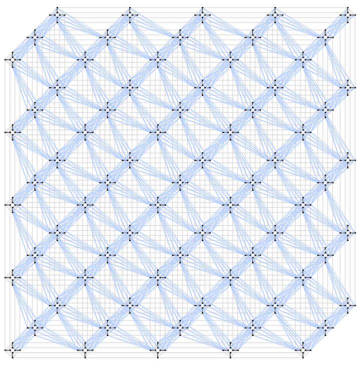
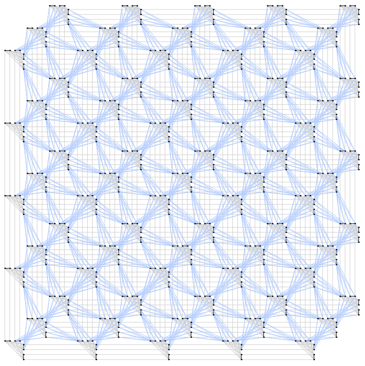

(d) Compressed

Figure 14: $(X, Y, Z) = (5, 5, 3)$ lattice of Pegasus (with all Pegasus-only edges either black or light blue).


(a) Tilted classic


(b) Diamond


(c) Upper triangle

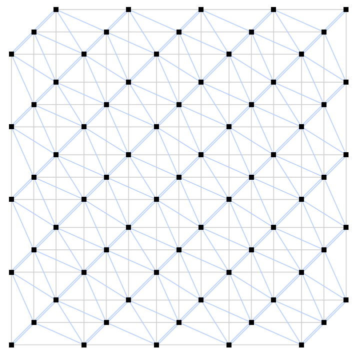
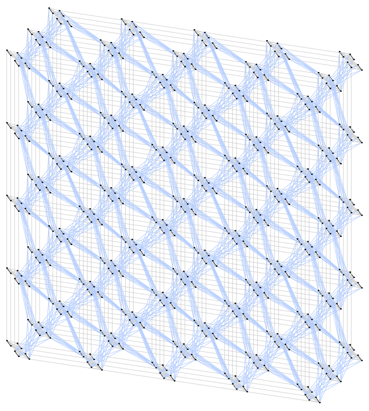
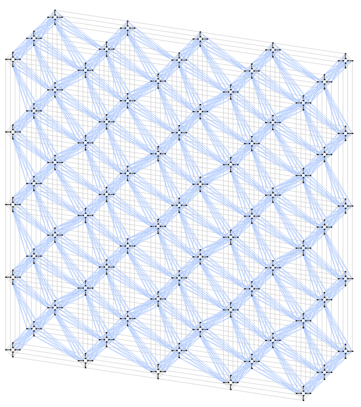
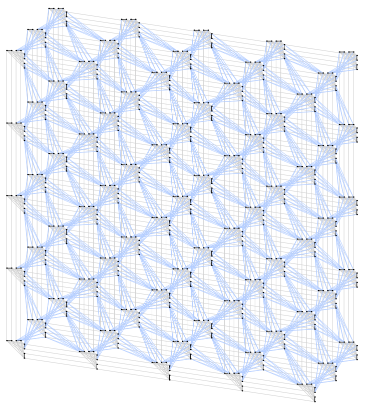
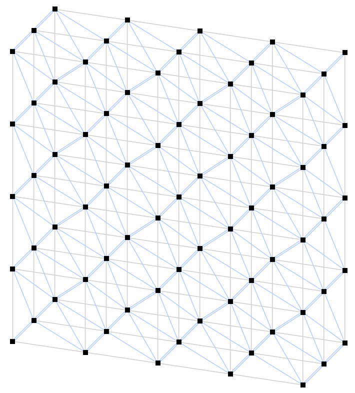

(d) Compressed

Figure 15: $(X, Y, Z) = (5, 5, 3)$ *tilted* lattice of Pegasus (with all Pegasus-only edges either black or light blue).


(a) Tilted classic


(b) Diamond


(c) Upper triangle


(d) Compressed

Embedding quadratization gadgets on Chimera and Pegasus graphs

Nike Dattani[*]
Harvard-Smithsonian Center for Astrophysics

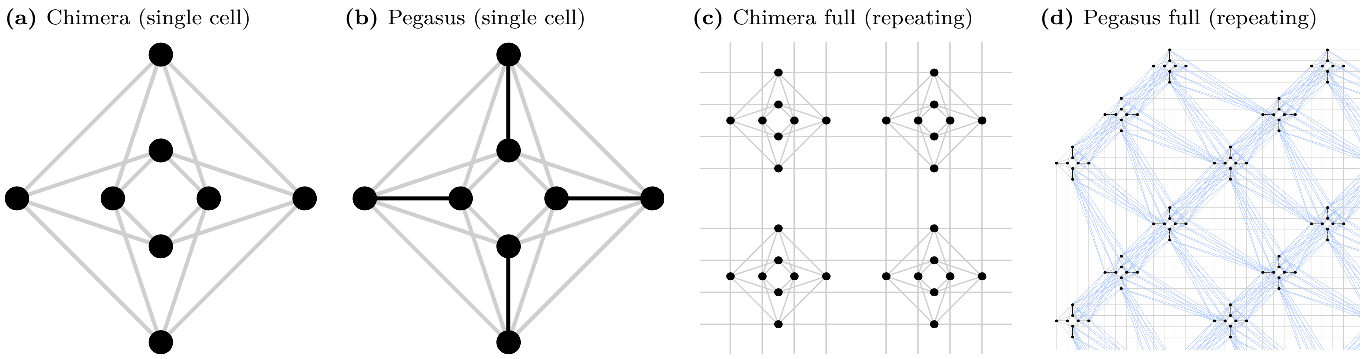
Nicholas Chancellor[†]
Durham University

We group all known quadratizations of cubic and quartic terms in binary optimization problems into six and seven unique graphs respectively. We then perform a minor embedding of these graphs onto the well-known Chimera graph, and the brand new *Pegasus* graph. We conclude with recommendations for which gadgets are best to use when aiming to reduce the total number of qubits required to embed a problem.

Discrete optimization problems are often naturally formulated in terms of minimizing some polynomial of degree > 2 [1], which is then 'quadratized' into a quadratic functionwhich can be solved using standard algorithms for universal classical computers [2], using special-purpose classical annealers [3], or using quantum annealers [4]. With dozens of quadratization methods available , one should choose the best quadratization for a given problem, and for a given method for solving the quadratized problem.

There are ways to quadratize functions of discrete variables without adding any auxiliary variables [5–8], but when those methods cannot be applied we introduce auxiliary variables. The resulting quadratic functions (called 'gadgets') that with good accuracy (or sometimes exact accuracy) simulate the original high-degree functions, will have some connectivity between the binary variables (or bits, or qubits, herein referred to for convenience only, as qubits) which can be represented by a graph in which vertices represent qubits and edges indicate when two different qubits appear together in a quadratic term. Since this graph incorporates no information about the linear terms, constant term, or the coefficients of the quadratic terms, many different gadgets have the same graph, therefore in this paper we will classify all known quadratization gadgets into categories according to their corresponding graph (herein called their 'gadget graph').

Figure 1: Graph connectivities for D-Wave's Chimera and Pegasus graphs.

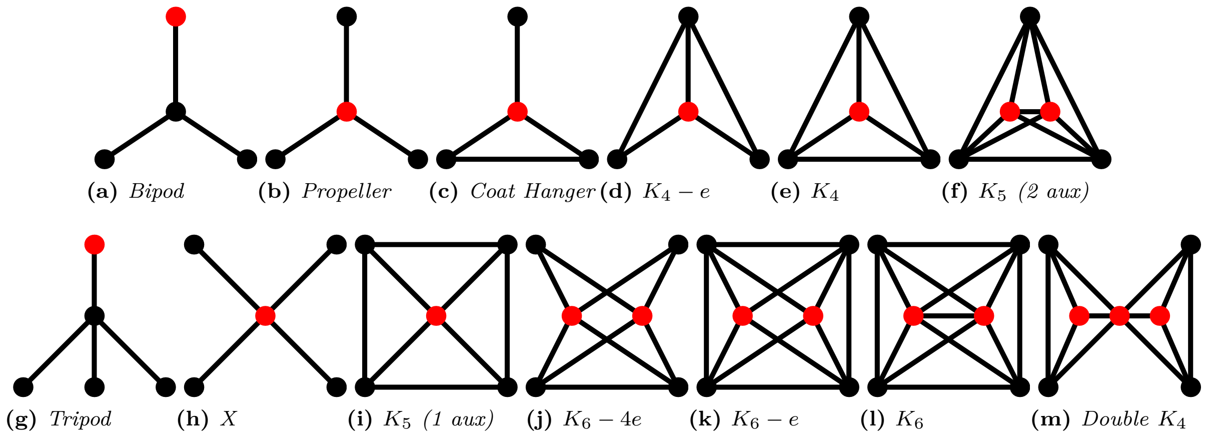

(a) Chimera (single cell)
(b) Pegasus (single cell)
(c) Chimera full (repeating)
(d) Pegasus full (repeating)

Gadget graphs for all known single cubic terms and for all known single quartic terms are given in Fig. 2. Gadget graphs tell us a lot about how costly the quadratic optimization problem will be, and those with larger connectivity tend to yield more difficult functions to optimize. Furthermore, some optimization methods only work if their corresponding graph has a certain connectivity, two examples of such connectivities being the ones in D-Wave's well-known Chimera graph [9], and in their very recent *Pegasus* graph, both shown in Fig. 1

Any graph, can be mapped onto the Chimera or Pegasus graphs by minor-embedding [10, 11], where the Chimera graph or the Pegasus graph is a graph minor of the graph representing the problem that needs to be optimized. This often means that one binary variable in the quadratic optimization problem needs to be represented by two (or more) qubits instead of one (called a "chain"), making the number of physical qubits needed to solve the original problem larger than before, and sometimes impossibly large to solve. For example if we wanted to factor the smallest RSA number which has not yet been factored on a classical computer (RSA-230) by minimization of a corresponding binary optimization problem, we would have to minimize a quartic function of 6594 variables which can be turned into a quadratic problem involving 148 776 variables [12]. Say we have a quantum annealer with 149 000 qubits: This means

[*] n.dattani@cfa.harvard.edu
[†] nicholas.chancellor@durham.ac.uk

Figure 2: *Gadget graphs.* Graphs showing the connectivity between qubits in quadratization gadgets for cubic to quadratic gadgets (top row), and quartic to quadratic gadgets (bottom row). Red vertices represent auxiliary qubits and black vertices represent logical qubits. Black edges denote the existence of a quadratic term in the gadget, involving the two corresponding qubits represented by vertices connected by the edge. *Linear and constant terms in the gadgets are completely ignored here.*



that only 224 out of the 142 182 gadget graphs can afford an auxiliary qubit for minor-embedding into the hardware's native connectivity graph. We therefore want *efficient gadgets* which can be minor-embedded into our hardware with *zero* or *very few* extra qubits.

In this paper we have provided minor-embeddings for all gadget graphs in Fig. 2, for both Chimera and Pegasus. We note that ***all*** cubic to quadratic gadgets involving one auxiliary qubit can be embedded onto Pegasus without any further auxiliary qubits for the embedding, because Pegasus contains the K_4 graph, which means any possible connections between the three logical qubits and the one auxiliary qubit are already contained in Pegasus. Since Chimera does not contain K_4, only negative cubic terms are so far known to be quadratizable with gadgets that embed directly onto Chimera without any extra qubits for the embedding. Furthermore, *all* cubic and quartic gadgets are embeddable with a single 'cell' of Pegasus and with chains of length at most two, whereas with Chimera cells, three of the quartic gadgets require multiple cells and two require chains of length three.

Table I: Grouping of all known quadratization gadgets for singe cubic terms, into categories corresponding to their gadget graphs. For each unique gadget graph, the number of auxiliary qubits required for construction of the gadget is listed, followed by the number of auxiliary qubits required to minor-embed the gadget graph onto Chimera and Pegasus.

Gadget Graph	Example Gadgets	N_{aux} **Quadratization**	N_{aux} **Embedding**	N_{aux} **Total**	N_{aux} **Embedding**	N_{aux} **Total**
			Chimera		**Pegasus**	
	Cubic → Quadratic					
Bipod	NTR-ABCB	1	0	**1**	0	**1**
Propeller	NTR-KZFD NTR-ABCG	1	0	**1**	0	**1**
Coat hanger	PTR-A	1	1	**2**	0	**1**
$\boldsymbol{K_4 - e}$	NTR-AC	1	1	**2**	0	**1**
$\boldsymbol{K_4}$	PTR-Ishikawa NTR-RBL-(3→ 2) PTR-BCR-1,2,3,4 PTR-KZ	1	2	**3**	0	**1**
$\boldsymbol{K_5}$ **(2 aux)**	PTR-RBL-(3→ 2)	2	3	**5**	1	**2**

I. MINOR EMBEDDINGS FOR CUBIC TO QUADRATIC GADGETS

A. Chimera graph

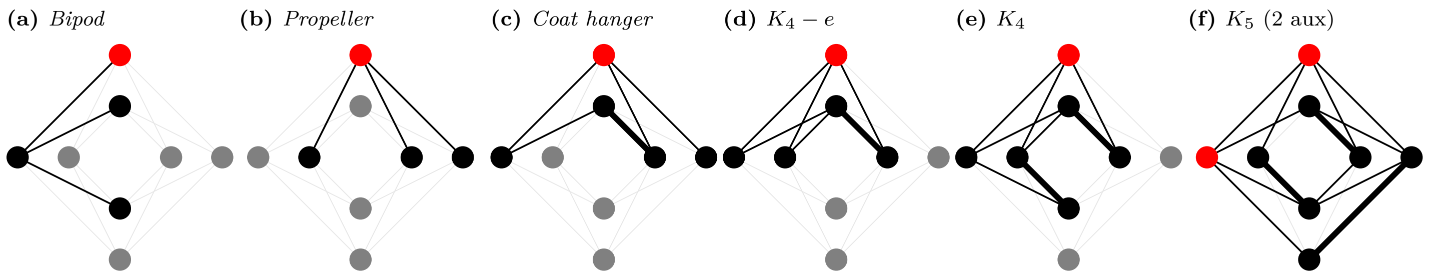


Figure 3: Minor embeddings of all ***cubic*** to quadratic gadgets onto a 'unit cell' of a Chimera graph. Grey vertices and edges are not used. Thick edges denote chains for minor embedding, in which two or more qubits (vertices) represent one logical qubit (this is done when logical qubits need to be connected to more qubits than the Chimera unit cell otherwise allows).

B. Pegasus graph

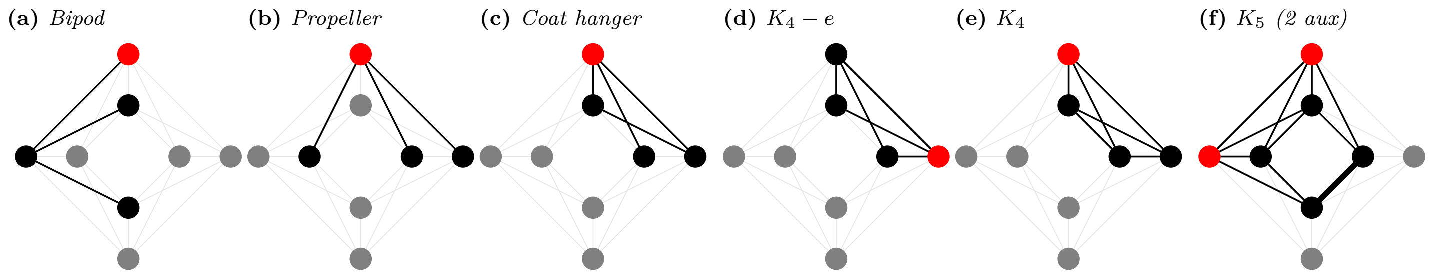


Figure 4: Minor embeddings of all ***cubic*** to quadratic gadgets onto a Pegasus 'cell'. Grey vertices and edges are not used.

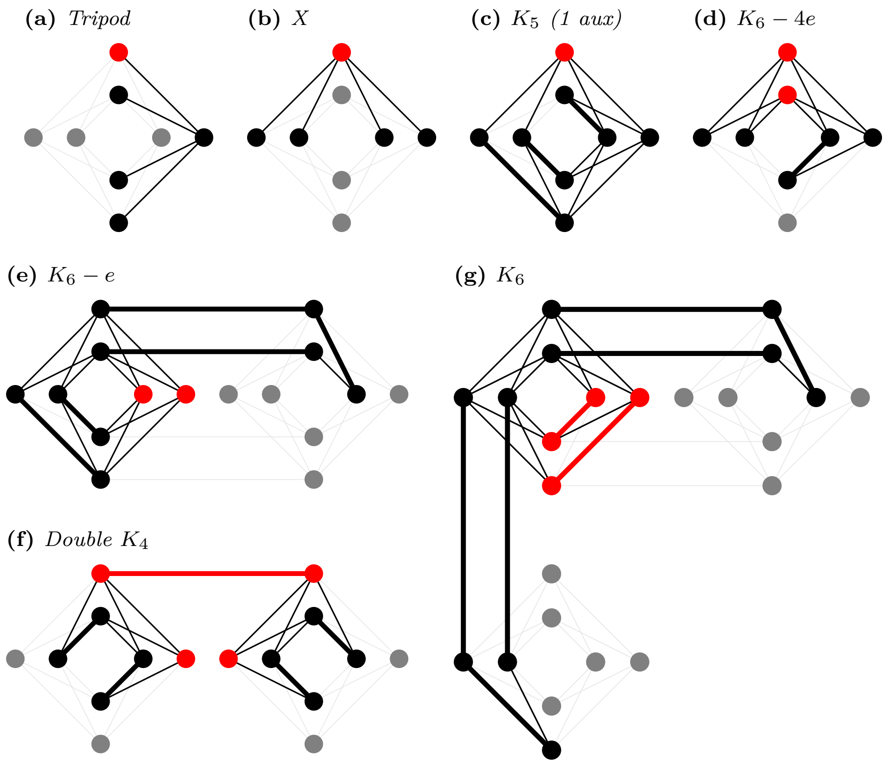
Table II: Grouping of all known quadratization gadgets for single ***quartic*** terms, into categories corresponding to their gadget graphs. For each unique gadget graph, the number of auxiliary qubits required for construction of the gadget is listed, followed by the number of auxiliary qubits required to minor-embed the gadget graph onto Chimera and Pegasus.

Gadget Graph	Example Gadgets	N_{aux} **Quadratization**	N_{aux} **Embedding**	N_{aux} **Total**	N_{aux} **Embedding**	N_{aux} **Total**
			Chimera		**Pegasus**	
Bipod	NTR-ABCB	1	0	**1**	0	**1**
X	NTR-KZFD	1	0	**1**	0	**1**
$\boldsymbol{K_5}$ **(1 aux)**	PTR-BCR-2 PTR-BCR-4 NTR-RBL-(4→ 2)	1	3	**4**	1	**2**
$\boldsymbol{K_6 - 4e}$	PTR	2	1	**3**	0	**2**
$\boldsymbol{K_6 - e}$	PTR-Ishikawa	2	5	**7**	2	**4**
$\boldsymbol{K_6}$	PTR-BCR-3	2	8	**10**	2	**4**
Double $\boldsymbol{K_4}$	PTR-CZW	3	5	**8**	1	**4**

II. MINOR EMBEDDINGS FOR QUARTIC TO QUADRATIC GADGETS

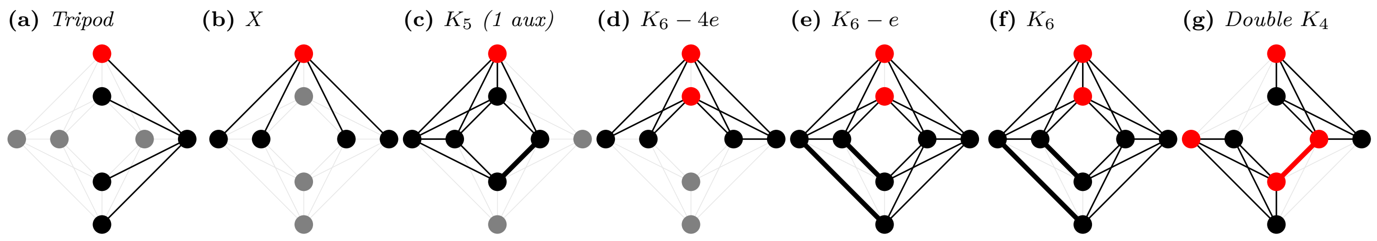
A. Chimera graph

Figure 5: Minor embeddings of all ***quartic*** to quadratic gadgets onto a ‘unit cell’ of a Chimera graph. Grey vertices and edges are not used. Thick edges denote chains for minor embedding, in which two or more qubits (vertices) represent one logical qubit (this is done when logical qubits need to be connected to more qubits than the Chimera unit cell otherwise allows).



B. Pegasus graph

Figure 6: Minor embeddings of ***quartic*** to quadratic gadgets onto a Pegasus ‘cell’. Grey vertices and edges are not used.



III. RECOMMENDED GADGETS

When reducing the number of total qubits is the most important factor in compiling a discrete optimization problem, negative terms (whether cubic or quartic) can be quadratized with only 1 auxiliary qubit, and in such a way not to require any additional qubits for embedding onto either Chimera and Pegasus (so only 1 auxiliary qubit is required in total).

For positive terms the situation is a bit more complicated: Still only 1 auxiliary qubit is required for quadratization of cubic terms, and for Pegasus the quadratization can be done in such a way as to require no further auxiliary qubits for the minor-embedding, however embedding with Chimera will require at minimum 1 further auxiliary qubit (so 1 auxiliary qubit is required in total for Pegasus, but 2 auxiliary qubits are required in total for Chimera).

For quartic positive terms, we need at least 2 total auxiliary qubits on Pegasus and at least 3 auxiliary qubits for Chimera. It is interesting to note that for positive quartic terms, while the gadgets leading to the K_5 graph require 1 fewer auxiliary qubit than the gadgets leading to the $K_6 - 4e$ graph for quadratization, the graph needs 2 more auxiliary qubits than $K_6 - 4e$ for embedding onto Chimera, and 1 more auxiliary qubit for embedding onto Pegasus (which requires 0 in the former case). Therefore, while gadgets leading to K_5 may appear at first sight to be more efficient than gadgets leading to $K_6 - 4e$, due to requiring fewer qubits for quadratization, they require more *total* auxiliary qubits for embedding onto Chimera and an equivalent number of total qubits for embedding onto Pegasus.

[1] N. S. Dattani and N. Bryans, http://arxiv.org/abs/1411.6758 (2014), arXiv:1411.6758.

[2] V. Kolmogorov, “QPBO,”.

[3] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, IEEE Journal of Solid-State Circuits **51**, 303 (2016).

[4] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, Nature **473**, 194 (2011).

[5] H. Ishikawa, in *2014 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2014) pp. 1362–1369.

[6] R. Tanburn, E. Okada, and N. Dattani, Physical Review A (2015), arXiv:1508.04816.

[7] E. Okada, R. Tanburn, and N. S. Dattani, , 5 (2015), arXiv:1508.07190.

[8] R. Dridi and H. Alghassi, Scientific Reports **7**, 43048 (2017).

[9] H. Neven, V. S. Denchev, M. Drew-Brook, J. Zhang, W. G. Macready, and G. Rose, *NIPS 2009 Demonstration: Binary Classification using Hardware Implementation of Quantum Annealing*, Tech. Rep. (2009).

[10] V. Choi, Quantum Information Processing **7**, 193 (2008).

[11] V. Choi, Quantum Information Processing **10**, 343 (2011).

[12] Z. Li, N. S. Dattani, X. Chen, X. Liu, H. Wang, R. Tanburn, H. Chen, X. Peng, and J. Du, http://arxiv.org/abs/1706.08061 (2017), arXiv:1706.08061.