

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 binary optimization problems into five 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. In cases where two or more graphs have a minor embedding with the same overhead in terms of auxiliary variables, we make recommendations for which gadgets are best to use for certain problems.

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

When quadratizing a high-degree term, the resulting quadratic function (called the ‘gadget’) 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’).


Gadget graphs for all known single cubic terms and for all known single quartic terms are given in Figure 1.

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 [5], and in their very recently presented *Pegasus* graph, both shown in Fig. 2.


Any graph, can be mapped onto the Chimera or Pegasus graphs by minor-embedding [6], 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 qubits instead of one, making the number

of qubits needed to solve the original problem much larger than before, and sometimes completely impossible. For example a quartic function with 1000 binary variables has $\binom{1000}{3} > 166$ million possible cubic terms and $\binom{1000}{4} > 40$ billion possible quartic terms which have to be quadrated, and then minor-embedded. If our minimization method can only be applied for up to 50 billion qubits, we cannot afford for each quartic-to-quadratic gadget to require its own auxiliary qubit for minor-embedding.


We have provided minor-embeddings for all gadget graphs in Figure 1, for both Chimera and Pegasus. We note that **all** cubic to quadratic gadgets involvng 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.

Gadgets with adjacency graph corresponding to `all_to_aux.tikz`():


- NTR-KZFD
- NTR-ABCG

Gadgets with adjacency graph corresponding to `logical_fork.tikz`():


- NTR-ABCB

Gadgets with adjacency graph corresponding to `k4_missing_2edge.tikz`():

- Asymmetric reduction

Gadgets with adjacency graph corresponding to `k4_missing_edge.tikz`():

- Asymmetric cubic reduction

Gadgets with adjacency graph corresponding to `k4.tikz`():

- PTR-Ishikawa

* n.dattani@cfa.harvard.edu

† nicholas.chancellor@durham.ac.uk

Figure 1

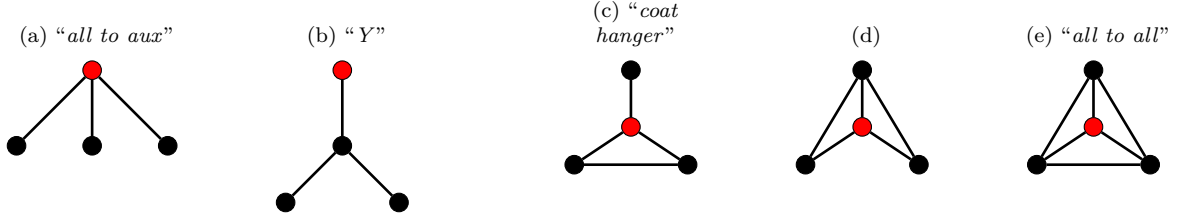


Figure 2

Table I

Gadget Graph	Example Gadgets	N_{aux} Quadratization	N_{aux} Chimera	N_{aux} Pegasus
Cubic \rightarrow Quadratic				
All to Aux	NTR-KZFD	1	0	0
	NTR-ABCG			
Y	NTR-ABCB	1	0	0
Coat Hanger	AR	1	1	0
(d)	ACR	1		0
	PTR-Ishikawa			
All to All	PTR-BCR(1-4)	1		0
	PTR-KZ			
Quartic \rightarrow Quadratic				
All2Aux	NTR-KZFD	1	0	0
Tripod	NTR-ABCB	1	0	0
All2Aux+1	PTR	2	1	0
All2All - 1	PTR-Ishikawa	2	?	2
	PTR-BCR-2			
All2Aux+K_4	PTR-BCR-4	1	3	2
	PTR-BCR-3	2	?	2
Two K'_4s	PTR-KZ	3	?	1

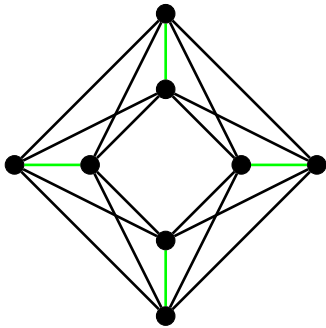


Figure 3: pegasus_v_chimera_uc.tikz: shows extra edges added to chimera unit cell to make Pegasus unit cell

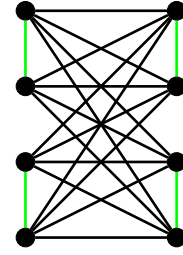


Figure 4: pegasus_v_chimera_uc_k44.tikz: shows extra edges added to chimera unit cell to make Pegasus unit cell

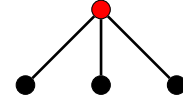




Figure 5: all_to_aux.tikz

- PTR-BCR (1-4)
- PTR-KZ
- Z version of PTR-KZ

Gadgets with adjacency graph corresponding to all_to_aux4.tikz():

- NTR-KZFD

Gadgets with adjacency graph corresponding to logical_fork3.tikz():

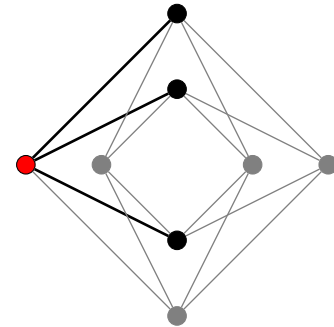


Figure 6: all_to_aux_chimera.tikz

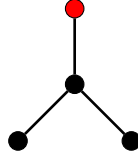


Figure 7: logical_fork.tikz

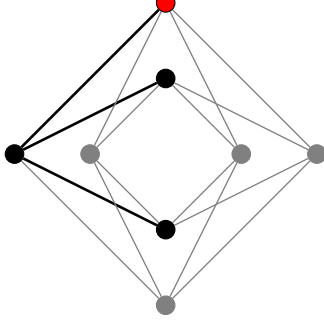
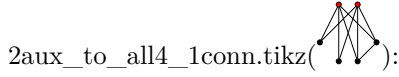


Figure 8: logical_fork_chimera.tikz

- NTR-ABCB

Gadgets with adjacency graph corresponding to



- positive term reduction

Gadgets with adjacency graph corresponding to



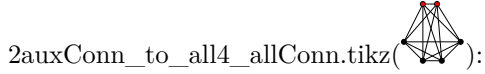
- PTR-Ishikawa

Gadgets with adjacency graph corresponding to



- PTR-BCR-2
- PTR-BCR-4

Gadgets with adjacency graph corresponding to



- PTR-BCR-3

Gadgets with adjacency graph corresponding to

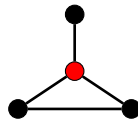
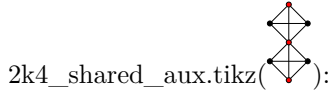


Figure 9: k4_missing_2edge.tikz

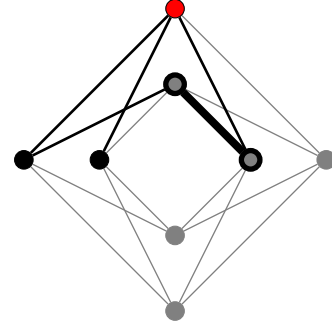


Figure 10: k4_missing_2edge_chimera.tikz

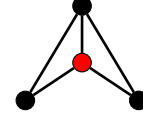


Figure 11: k4_missing_edge.tikz

- Two copies of PTR-KZ sharing an auxilla
- Two copies of z-version of PTR-KZ sharing an auxilla

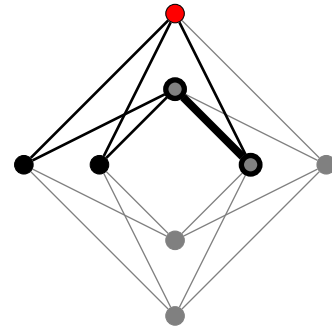


Figure 12: k4_missing_edge_chimera.tikz

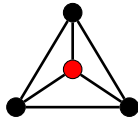


Figure 13: k4.tikz

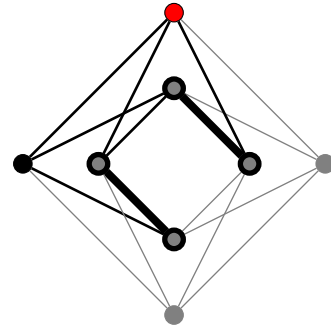


Figure 14: k4_chimera.tikz

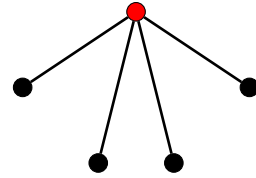


Figure 15: all_to_aux4.tikz

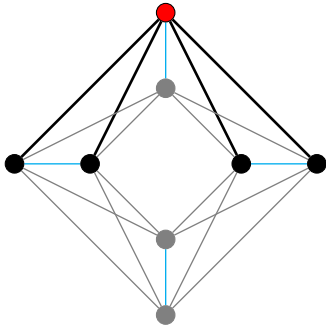


Figure 16: all_to_aux4_inPegasus.tikz

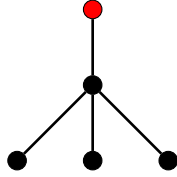


Figure 17: logical_fork3.tikz

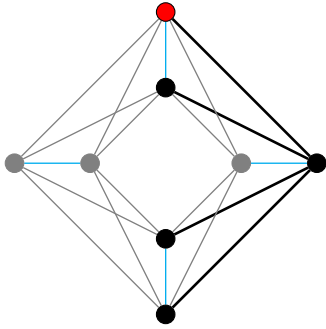


Figure 18: logical_fork3_inPegasus.tikz

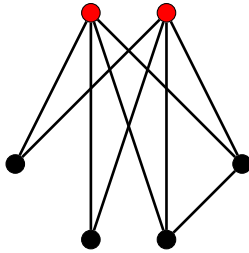


Figure 19: 2aux_to_all4_1conn.tikz

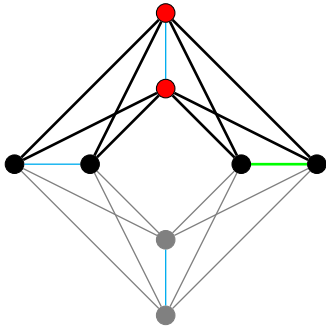


Figure 20: 2aux_to_all4_1conn_inPegasus.tikz

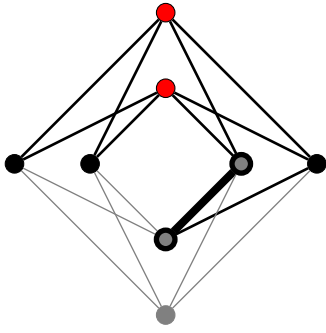


Figure 21: 2aux_to_all4_1conn_inChimera.tikz

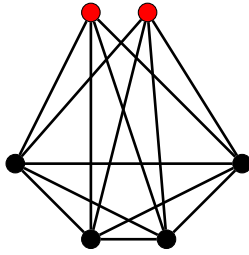


Figure 22: 2aux_to_all4_allConn.tikz

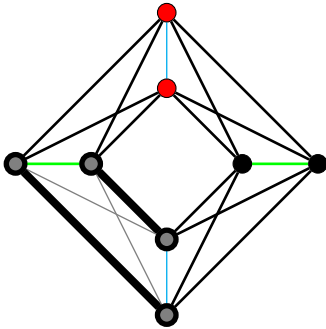


Figure 23: 2aux_to_all4_allConn_inPegasus.tikz

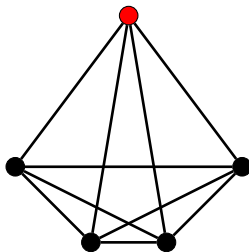


Figure 24: aux_to_all4_allConn.tikz

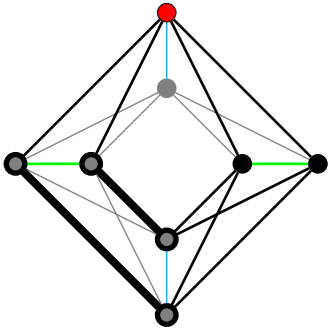


Figure 25: aux_to_all4_allConn_inPegasus.tikz

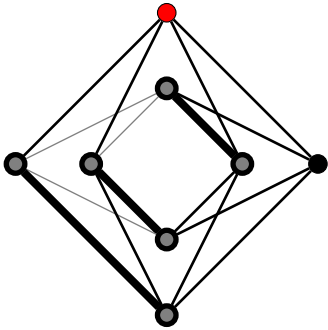


Figure 26: aux_to_all4_allConn_inChimera.tikz

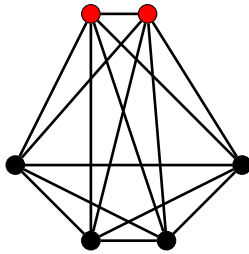


Figure 27: 2auxConn_to_all4_allConn.tikz

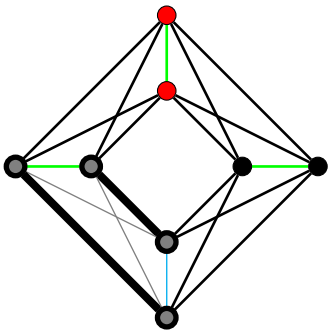


Figure 28: 2auxConn_to_all4_allConn_inPegasus.tikz

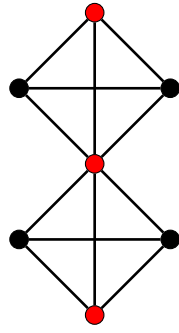


Figure 29: 2k4_shared_aux.tikz

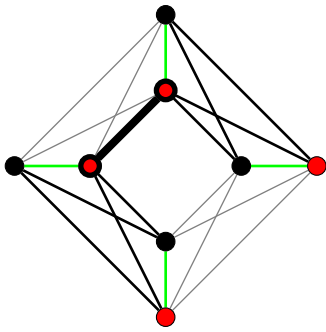


Figure 30: 2k4_shared_aux_inPegasus.tikz