Decomposition Algorithms for Scalable Quantum Annealing

Elijah Pelofske
Los Alamos National Laboratory, Los Alamos, NM 87545, USA
epelofske@lanl.gov

Georg Hahn
Lancaster University, Lancaster LA1 4YW, U.K.
ghahn@cantab.net

Hristo Djidjev
Los Alamos National Laboratory, Los Alamos, NM 87545, USA
djidjev@lanl.gov

CCS CONCEPTS

• **Theory of computation** → *Quantum computation theory*; *Graph algorithms analysis.*

ACM Reference Format:
Elijah Pelofske, Georg Hahn, and Hristo Djidjev. 2019. Decomposition Algorithms for Scalable Quantum Annealing. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Commercial adiabatic quantum annealers such as D-Wave 2000Q, which is available at LANL, have the potential to solve important NP-hard optimization problems efficiently. However, one of the primary constraints of such devices is the limited number and connectivity of their qubits, which limits the size of the problems directly solvable on the machine to about 65 variables. This research presents two exact decomposition methods (for the Maximum Clique and the Minimum Vertex Cover problem, two important and well known NP-hard problems) that allow solving problems of arbitrarily large sizes by splitting them up recursively into a series of suitably small subproblems. Those subproblems are then solved exactly or approximately using any method of choice, which includes a quantum annealer. Whereas some previous approaches are based on heuristics that do not guarantee optimality of their solutions, our decomposition algorithms have the property that the optimal solution of the input problem can be reconstructed in polynomial time given all generated subproblems are solved optimally. We present a study of various heuristic and exact bounds as well as reduction methods that help increase the scalability of our decomposition algorithms.

2 MAXIMUM CLIQUE PROBLEM

Given a graph G, the *maximum clique* (MC) problem asks to find a subset S of the vertices of G of maximum size such that there is an edge in G joining any two vertices of S. To solve the MC problem in case G is too large to fit in the annealer hardware, we use the CH-partitioning introduced in [5], [4], in order to split G into two smaller subgraphs G_1 and G_2 on which a MC should be found. We send to D-Wave any of the subgraphs G_1 and G_2 that is small enough to fit the annealer hardware and record the found solution. In case any of G_1 or G_2 is still too large to be solved directly on the annealer hardware, we apply the decomposition algorithm recursively. Finally, we use the solutions for G_1 and G_2 to construct a solution for G.

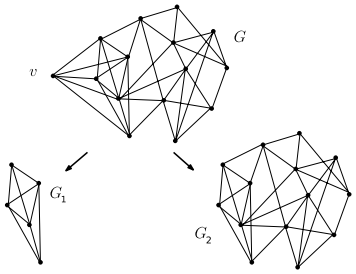

Figure 1: MC decomposition

As shown in Figure 1, in order to compute suitable G_1 and G_2, we choose an arbitrary vertex of G and define G_1 as the subgraph of G induced by all neighbors of v (but without v itself), and define G_2 as the graph resulting when v and all edges incident to v are removed from G. Intuitively, G_1 contains all cliques of G that contain v, and G_2 contains all cliques of G that do not contain v. Hence, a maximum clique of G will be either in G_1 or G_2.

Since each of the subgraphs G_1 and G_2 contains at least one less vertex than G, the decomposition algorithm is guaranteed to complete in finite time, but the number of subgraphs generated might be exponential in the worst case. To reduce the number of subgraphs, we use upper and lower bounds on the solution contained in any generated subgraph (with the aim of discarding them) as well as reduction techniques. For instance, if for a currently considered subgraph G_i we compute an upper bound b_i for G_i, meaning that the size of the MC for G_i is not greater than b_i, and the largest clique found so far has size larger than b_i, then G_i can be ignored. This technique, widely used in branch-and-bound algorithms in combinatorial optimization, can lead to a dramatic decrease in the number of subgraphs that are generated.

We use and compare the effectiveness of several upper bound techniques: (i) We use a greedy search heuristic to find an upper bound on the *chromatic number* of the graph, which is the minimum number of colors needed to color each vertex of G such that no edge connects two vertices of the same color. Since each vertex in a clique must have a distinct color, the chromatic number is an upper bound on the clique number [7]. Although computing the chromatic number is NP-hard, there are much better heuristics for its approximation compared with the ones for the clique number. Therefore, a greedy search heuristic for the chromatic number provides an easily computable bound. To compute a graph coloring we use the heuristic function *greedy_color* of the NetworkX package [6], which is applied to the complement $\overline{G}$ of G. (ii) The *Lovász*

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

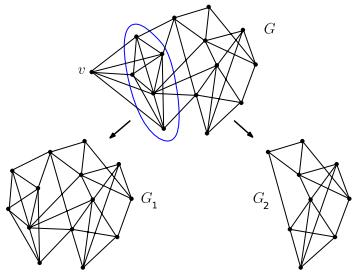

Figure 2: MVC decomposition

number of the complement of a graph G is defined as an upper bound on the Shannon capacity of $\overline{G}$ [8], and is an upper bound of the clique number, typically better than the chromatic number [7]. A drawback is that existing algorithms for computing the Lovász number are not very scalable because they involve solving a semi-definite programming problem. (iii) Any maximum clique heuristic can be used to provide a lower bound on the clique number, provided the result returned by the heuristic is a clique. We use the *Fast Maximum Clique Finder* of [9] in heuristic mode. We call this the *FMC heuristic*. (iv) At any point in the decomposition tree, a lower bound on one of the current subbranches can be obtained simply by solving the other subbranch. This gives a lower bound for the maximum clique size in all nodes (and leaves) of the yet unsolved branch. We call this the *decomposition bound*.

3 MINIMUM VERTEX COVER PROBLEM

A vertex cover of graph G is a set S of vertices of G such that each edge of G has at least one endpoint in S. The minimum vertex cover (MVC) problem is to find a vertex cover of minimum size, and it is an NP-hard problem.

To find a MVC of a graph too large for the hardware, we use a similar decomposition strategy as for the MC problem. As shown in Figure 2, we again select an arbitrary vertex v of G, we define G_1 as the graph resulting from the removal of v and all edges incident to v from G, and we define G_2 as the graph resulting from the removal of vertex v as well as all neighbors of v in G. As in the MC case, G_1 contains all MVCs of G that contain v, and G_2 contains all MVCs of G that do not contain v. Because of the close relationship between the MV and MVC problems, the upper bounds described for the MC problem can be used as lower bounds for the MVC problem.

4 EXPERIMENTAL ANALYSIS

We examine the role of vertex choice, lower and upper bounds, and reduction algorithms in order to improve the computational time for our algorithms.

4.1 Vertex Choice

While any vertex v can be used when computing the decomposition, the choice of the vertex can significantly impact the number of subproblems generated and, hence, the running time. For both the MC and the MVC problems, we test four different vertex choices: we choose v as a vertex of highest, lowest, or median degree; or choosing v randomly.

4.2 Bounds

We experiment with the upper and lower bounds described above. For the MC problem the bounds are the following: (i) Chromatic number heuristic upper bound; (ii) Lovász number upper bound; (iii) FMC heuristic lower bound; (iv) Deterministic bounds lower bound. For the MVC problem, the techniques are the same, but the bounds are inverted.

4.3 Reduction techniques

These techniques help reduce the sizes of the graphs without affecting the MC or the MVC, respectively. For the MC problem, we test two different reduction techniques: (i) Persistency Analysis [3]; and (ii) k-core reduction based on the current lower bound and the edge k-core algorithm [2, 4]. In the case of MVC, we use three reduction techniques: (i) Persistency Analysis [3] ; (ii) Neighbor-based vertex removal [10]; and (iii) B&R Reduction [1].

4.4 Conclusion

Based on the shortest running times of each of these three techniques, we find the following best combination for both the MC and MVC problems.

The *DBR algorithm* uses high degree vertex selection, decomposition based upper bound, a chromatic number heuristic lower bound, and neighbor based vertex removal for graph reduction for solving the MVC problem.

The *DBK algorithm* uses low degree vertex selection, decomposition based lower bound, a chromatic number heuristic upper bound, and k-core reduction for graph reduction for solving the MC problem.

Using DBR and DBK respectively, we can solve the MVC and MC problems on graphs with up to about 300 vertices and roughly 22450 edges.

REFERENCES

[1] Takuya Akiba and Yoichi Iwata. 2015. Branch-and-Reduce Exponential/FPT Algorithms in Practice: A Case Study of Vertex Cover. *2015 Proceedings of the Seventeenth Workshop on Algorithm Engineering and Experiments (ALENEX)* (2015), 1–12.
[2] V. Batagelj and M. Zaversnik. 2011. An O(m) Algorithm for Cores Decomposition of Networks. *Adv Dat An Class* 5, 2 (2011).
[3] E. Boros and P.L. Hammer. 2002. Pseudo-Boolean optimization. *Discrete Appl Math* 123, 1–3 (2002), 155–225.
[4] G. Chapuis, H. Djidjev, G. Hahn, and G. Rizk. 2017. Finding Maximum Cliques on the D-Wave Quantum Annealer. *Proceedings of the 2017 ACM International Conference on Computing Frontiers (CF'17)* (2017), 1–8.
[5] H.N. Djidjev, G. Hahn, A. Niklasson, and V. Sardeshmukh. 2015. Graph Partitioning Methods for Fast Parallel Quantum Molecular Dynamics. *SIAM Workshop on Combinatorial Scientific Computing CSC16* (2015).
[6] Aric Hagberg, Daniel Schult, and Pieter Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. *Proceedings of SciPy2008* (2008), 11–15.
[7] Donald E Knuth. 1993. The sandwich theorem. *Electron J Comb* 1, A1 (1993), 1–49.
[8] L. Lovász. 1979. On the Shannon Capacity of a Graph. *IEEE Trans Inf Theory* IT-25, 1 (1979), 1–7.
[9] Bharath Pattabiraman, Md. Mostofa Ali Patwary, Assefaw H. Gebremedhin, Wei-keng Liao, and Alok Choudhary. 2013. Fast Algorithms for the Maximum Clique Problem on Massive Sparse Graphs. In *Algorithms and Models for the Web Graph*, Anthony Bonato, Michael Mitzenmacher, and Paweł Prałat (Eds.). Springer International Publishing, Cham, 156–169.
[10] E. Pelofske, G. Hahn, and H. Djidjev. 2019. Solving large Minimum Vertex Cover problems on a quantum annealer. *Proceedings of the Computing Frontiers Conference CF'19* (2019), 76–84.