Graph Partitioning using Quantum Annealing on the D-Wave System

Hayato Ushijima-Mwesigwa
Clemson University
Clemson, SC
Computer, Computational, & Statistical Sciences Division, Los Alamos National Laboratory
Los Alamos, NM
hushiji@g.clemson.edu

Christian F. A. Negre
Theoretical Division, Los Alamos National Laboratory
Los Alamos, NM
cnegre@lanl.gov

Susan M. Mniszewski*
Computer, Computational, & Statistical Sciences Division, Los Alamos National Laboratory
Los Alamos, NM
smm@lanl.gov

ABSTRACT

Graph partitioning (GP) applications are ubiquitous throughout mathematics, computer science, chemistry, physics, bioscience, machine learning, and complex systems. Post Moore's era supercomputing has provided us an opportunity to explore new approaches for traditional graph algorithms on quantum computing architectures. In this work, we explore graph partitioning using quantum annealing on the D-Wave 2X machine. Motivated by a recently proposed graph-based electronic structure theory applied to quantum molecular dynamics (QMD) simulations, graph partitioning is used for reducing the calculation of the density matrix into smaller subsystems rendering the calculation more computationally efficient. Unconstrained graph partitioning as community clustering based on the modularity metric can be naturally mapped into the Hamiltonian of the quantum annealer. On the other hand, when constraints are imposed for partitioning into equal parts and minimizing the number of cut edges between parts, a quadratic unconstrained binary optimization (QUBO) reformulation is required. This reformulation may employ the graph complement to fit the problem in the *Chimera graph* of the quantum annealer. Partitioning into 2 parts and k parts concurrently for arbitrary k are demonstrated with benchmark graphs, random graphs, and small material system density matrix based graphs. Results for graph partitioning using quantum and hybrid classical-quantum approaches are shown to be comparable to current "state of the art" methods and sometimes better.

CCS CONCEPTS

• **Computer systems organization** → **Quantum computing**; • **Mathematics of computing** → *Graph algorithms*;

*To whom correspondence should be addressed. E-mail: smm@lanl.gov

PMES'17, November 12–17, 2017, Denver, CO, USA

ACM ISBN 978-1-4503-5126-3/17/11…$15.00
https://doi.org/10.1145/3149526.3149531

KEYWORDS

Graph-partitioning, Community detection, Quantum Annealing

ACM Reference Format:
Hayato Ushijima-Mwesigwa, Christian F. A. Negre, and Susan M. Mniszewski. 2017. Graph Partitioning using Quantum Annealing on the D-Wave System. In *PMES'17: PMES'17: Second International Workshop on Post Moore s Era Supercomputing, November 12–17, 2017, Denver, CO, USA*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3149526.3149531

Quantum annealing (QA) is a combinatorial optimization technique meant to exploit quantum-mechanical effects such as tunneling and entanglement [31] to minimize and sample from energy-based models. Machines performing QA at the hardware level such as the D-Wave computer have recently become available [2, 29], and evidence for quantum mechanical effects playing a useful role in the processing have been seen [1, 3, 28]. The D-Wave system minimizes the following Ising objective function.

$$O(\mathbf{h},\mathbf{J},\mathbf{s}) = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j \qquad (1)$$

This is closely related to the Ising model energy function as a problem Hamiltonian, where spin variables $s_i \in \{-1, +1\}$ are subject to local fields h_i and pairwise interactions with coupling strengths J_{ij}.

Quantum computers use quantum bits (qubits) to hold information. Each qubit's behavior is governed by the laws of quantum mechanics, enabling qubits to be in a "superposition" state – that is, both a "-1" and a "+1" at the same time, until an outside event causes it to collapse into either a "-1" or a "+1" state. The output of an anneal is a low-energy ground state $\mathbf{s}$, which consists of an Ising spin for each qubit where $s_i \in \{-1, +1\}$. This is the basis upon which a quantum computer is constructed which gives the ability to quickly solve certain classes of NP-hard complex problems such as optimization, machine learning and sampling problems.

On the D-Wave device the connectivity between the binary variables s_i is described by a fixed sparse graph $G = (V, E)$ called the *Chimera graph*. Nodes in V as qubits represent problem variables with programmable weights, and edges as couplers in E have programmable connection strengths.

There are weights (h_i) associated with each qubit (s_i) and strengths (J_{ij}) associated with each coupler between qubits (s_i and s_j). A quantum machine instruction (QMI) solves the objective function given the weights, strengths, and qubits. The D-Wave 2X system has 1095 qubits and 3061 couplers with sparse bipartite connectivity. The *Chimera graph* of this particular machine consists of a 12 x 12 array of 4 x 4 bipartite unit cells.

Often the quantum unconstrained binary optimization (QUBO) representation with it's 0/1-valued variables is more natural than the Ising -1/+1-valued variables. The QUBO objective function is shown in Eq. 2, where Q is an $n \times n$ upper-triangular matrix of coupler strengths and x is a vector of binary variables (0/1). Q_{ii} is an analog to the Ising h_i, as are Q_{ij} and J_{ij}. The Ising and QUBO models are related through the transformation $s = 2x - 1$. The D-Wave machine allows for either form.

$$O(\mathbf{Q}, \mathbf{x}) = \sum_i Q_{ii} x_i + \sum_{i<j} Q_{ij} x_i x_j \tag{2}$$

Physical constraints on current D-Wave platforms such as limited precision/control error and range on weights and strengths, sparse connectivity, and number of available qubits have an impact on the problem size and performance. Embedding algorithms are required to map or fit a problem graph onto the hardware. Strictly quantum approaches are limited by the number of graph nodes that can be represented on the hardware. Larger graphs require hybrid classical-quantum approaches.

The field of mathematics devoted to methods for efficiently partitioning a graph dates back to the 1970's [8, 25, 38]. Graph partitioning (GP) methods emerged to reduce the complexity of graphs for many different purposes such as applying divide and conquer techniques for efficient computation [26]. Other applications of GP include physical network design, VLSI design, telephone network design, load balancing of high performance computing (HPC) codes to minimize total communication between processors [22], distributed sparse matrix-vector multiplication (partitioning the rows of a matrix to minimize communication), physics lattices [14, 20], chemical elements related through bonds [36], metabolic networks [17] and social networks [7, 19, 32, 40].

Graph theory algorithms are used to determine the embedding of a problem graph on the D-Wave system [5, 9, 18]. Other graph theory algorithms that have been implemented on the D-Wave system include graph coloring [11, 12], a graph isomorphism solver [43], and spanning tree calculations [35].

This work is motivated by a recently proposed graph-based electronic structure theory applied to quantum molecular dynamics (QMD) simulations [34]. In this approach, GP is used for reducing the calculation of the density matrix into smaller subsystems rendering the calculation computationally more efficient. This procedure is done at each timestep of a QMD simulation taking the previous density matrix as an adjacency matrix for the new graph.

When GP is unconstrained (with no limitation on partition size) and communication volume is minimized, the resulting natural parts are called communities. Community detection (CD) gives a high level "skeleton" view of the general structure of a graph based on a modularity metric [17, 32]. Detection and characterization of community structure in networks has been used to identify secondary structures in proteins [34] and to better understand the complex processes occurring in amino acid networks such as the allosteric mechanism in proteins [37].

Uniform or constrained GP partitions a graph into similar-sized parts while minimizing the number of *cut edges* between parts. Classical approaches to GP rely on heuristics and approximation algorithms. In this work, we initially address partitioning into 2 parts. Considering the D-Wave architecture, a quantum GP approach should be able to partition a graph into k parts concurrently, without recursion or stages. In our work, we were able to accomplish this by using ideas from the graph coloring problem [11, 12, 27]. A super-node is used to represent k subnodes where the graph is split into k parts. This is a natural formulation for the D-Wave computer, but quickly runs out of real estate for large graphs and large partitionings, requiring the use of a hybrid classical-quantum approach.

The D-Wave 2X computer is able to solve graphs of limited size (∼45 nodes). Graphs addressing interesting scientific questions quickly exceed this size. In order to extend this limit we explore techniques such as representing the graph by its complement for GP and reducing the coupler count by thresholding the modularity matrix for CD. Additionally, hybrid classical-quantum approaches can be employed, such as *qbsolv* [4], where processing on the CPU creates subgraphs to be run on the quantum processing unit (QPU) and assembled for the final result.

Following, we describe our methods for traditional algorithms for CD and GP implemented on the D-Wave quantum annealer, as well as methods formulated to take advantage of the hardware. Results are compared with existing benchmarks and current "state of the art" tools.

1 METHODS

1.1 Graph Clustering/Community Detection

A graph can be divided in sets of nodes belonging to different communities (also called clusters). Nodes within any of these communities have a high probability of being connected (high intraconnectivity); whereas nodes in different communities have a lower probability of being connected (low interconnectivity). This natural division of a graph into communities differs from the usual GP problem in that the size of the communities cannot be predefined a priori.

One metric that can quantify the quality of a community structure is the modularity [32, 33]. This metric performs a comparison of the connectivity of edges within communities with the connectivity of an equivalent network where edges are placed randomly. Let $G = (V, E)$ be a weighted graph with an adjacency matrix A. The modularity matrix B is

taken as the difference between A and a matrix constructed as an outer product of the vector degree $\mathbf{g}$. The node degree g_i is defined as $g_i = \sum_j A_{ij}$. Newman's expression for the modularity matrix B can be rewritten as follows:

$$B = A - \frac{\mathbf{g}\mathbf{g}^T}{2m} \tag{3}$$

If we now have a vector of labels $\mathbf{s}$, with $s_i \in \{-1, +1\}$ with "-1" or "+1" classifying nodes corresponding to different communities, the problem of finding the optimal modularity requires solving $\max_{\mathbf{s}}(Q(\mathbf{s}))$, or $\min_{\mathbf{s}}(-Q(\mathbf{s}))$ where:

$$Q(\mathbf{s}) = \mathbf{s}^T B \mathbf{s} \tag{4}$$

By explicitly writing all the terms of Eq. 4, we can identify that the coupler strength J_{ij} and onsite energy h_i will have to be set to $-2B_{ij}$ and $-B_{ii}$ respectively to embed the problem on the D-Wave system. We can see that matrix B does not impose any restrictions and requires no reformulation for the embedding. The problem of Eq. 4 is formulated as an Ising problem and as a consequence it is straightforward to apply quantum annealing techniques using the D-Wave system. In order to divide the graph into 2^N communities, a recursive subdivision can be performed where each of the communities is subdivided into two new communities.

1.2 The Graph Partitioning Problem

Across multiple disciplines, graphs are frequently used to model different application problems. The sizes of these graphs can be arbitrarily large compared to the computational resources at hand. In order to reduce the complexity or enable parallelization, regardless of the application, a common technique used is to partition the graph into smaller subproblems. It has been shown that GP is an NP-hard problem thus achieving an efficient exact solution seems unlikely unless $P = NP$ [16, 23].

Popular GP algorithms such as Kernighan-lin [25], Fiduccia-Mattheyses [15] algorithms, or spectral methods incorporated into multi-level methods are often used. Since GP is often studied as a combinatorial optimization problem, combinatorial optimization methods, in particular, metaheuristics such as simulated annealing are also commonly used. In this work, we demonstrate how quantum annealing, a metaheuristic, can be used for partitioning a graph. We first formally define the GP problem and then relate it to quantum annealing and show how it can be mapped onto the D-Wave hardware.

Let $G = (V, E)$ be a graph with vertex set V and edge set E such that $n = |V|$ (number of vertices) and $m = |E|$ (number of edges). For a fixed integer k, the GP problem is to find a partition, $\Pi = (\Pi_1, \ldots, \Pi_k)$, of the vertex set V into k parts, i.e.,$\Pi_1 \cup \Pi_2 \cup \cdots \cup \Pi_k = V$ and $\Pi_i \cap \Pi_j = \emptyset$ for $i \neq j$ subject to the balancing constraint $|\Pi_i| \leq (1+\epsilon)\lceil \frac{n}{k} \rceil \quad i = 1, \ldots, k$. In this work, we formulate the GP problem for the case when $\epsilon = 0$. This balancing constraint is sometimes known as *perfectly balanced*. The objective is to minimize the number of *cut edges*. A cut edge is defined as an edge whose end points are in different parts. In order to map the GP problem on the D-Wave hardware, we need to formulate it as a QMI. We first formulate it for $k = 2$ and later generalize it for arbitrary k. For $k = 2$, the objective of minimizing the number of *cut edges* can be formulated into a quadratic programming problem as follows: Label each vertex i with $s_i = \pm 1$, depending on the part they belong to, then it can easily be shown that the number of *cut edges* is given by $\frac{1}{4}\mathbf{s}^T L \mathbf{s}$ where L is the *Laplacian matrix* of G. Thus, the graph partitioning problem is equivalent to

$$\begin{aligned} & \min_{\mathbf{s}}(\tfrac{1}{4}\mathbf{s}^T L \mathbf{s}) \\ \text{subject to} \quad & \sum_i s_i = 0 \\ \text{with} \quad & s_i \in \{-1, 1\}, \;\; i = 1, ..., n \end{aligned} \tag{5}$$

Another equivalent formulation using the adjacency matrix A is

$$\begin{aligned} & \max_{\mathbf{s}}(\mathbf{s}^T A \mathbf{s}) \\ \text{subject to} \quad & \sum_i s_i = 0 \\ \text{with} \quad & s_i \in \{-1, 1\}, \;\; i = 1, ..., n \end{aligned} \tag{6}$$

whose objective function is the difference between the number of edges and twice the number of *cut edges*.

In order to partition a graph using a quantum annealer, the graph partitioning problem must be reformulated as a QUBO or Ising problem in analogy with Eq. 1 or 2. In order to do this, the constraints in the formulation (6) need to be removed, leading to a relaxation of the original problem as follows:

$$\begin{aligned} & \max_{\mathbf{s}}(\beta \mathbf{s}^T A \mathbf{s} - \alpha(\textstyle\sum s_i)^2) \\ \text{with} \quad & s_i \in \{-1, 1\} \; i = 1, ..., n \end{aligned} \tag{7}$$

where α and β are weight parameters such that an increase in α indicates an importance of the balancing criterion while an increase in β indicates an increase in importance for a smaller cut over the balancing criterion. α and β are chosen according to [30]. We refer to formulation (7) as the Ising formulation where variable $s \in \{-1, +1\}$.

Since $(\sum s_i)^2 = \sum s_i^2 + 2\sum s_i s_j = \mathbf{s}^T \mathbb{1}_{n\times n} \mathbf{s}$, where $\mathbb{1}_{n\times n}$ is the $n \times n$ matrix with all entries equal to 1. We can rewrite formulation (7) as

$$\begin{aligned} & \max_{\mathbf{s}}(\mathbf{s}^T(\beta A - \alpha \mathbb{1}_{n\times n})\mathbf{s}) \\ \text{with} \quad & s_i \in \{-1, 1\} \; i = 1, ..., n \end{aligned} \tag{8}$$

Or equivalently,

$$\begin{aligned} & \min_{\mathbf{s}}(\mathbf{s}^T(\alpha \mathbb{1}_{n\times n} - \beta A)\mathbf{s}) \\ \text{with} \quad & s_i \in \{-1, 1\} \; i = 1, ..., n \end{aligned} \tag{9}$$

In order to transform the Ising model into a QUBO, we use the transformation $\mathbf{s} = 2\mathbf{x} - \mathbb{1}_n$ where $\mathbb{1}_n$ is the vector of all ones and $\mathbf{x} \in \{0, 1\}^n$. In general, if we apply the transformation to any symmetric matrix M, a straightforward substitution and simplification gives the transformation $\mathbf{s}^T M \mathbf{s} = 4\mathbf{x}^T M \mathbf{x} - 4\mathbf{x}^T M \mathbb{1}_n + \mathbb{1}_n^T M \mathbb{1}_n$. Thus, taking $M = \alpha \mathbb{1}_{n\times n} - \beta A$, we have

$$\begin{aligned} & \min_{\mathbf{s}}(\mathbf{s}^T(\alpha \mathbb{1}_{n\times n} - \beta A)\mathbf{s}) \\ = & \min_{\mathbf{x}}(\; \mathbf{x}^T(\alpha \mathbb{1}_{n\times n} - \beta A)\mathbf{x} - \mathbf{x}^T(\alpha \mathbb{1}_{n\times n} - \beta A)\mathbb{1}_n). \end{aligned} \tag{10}$$

These QUBO variables are then mapped onto the *Chimera graph* of the D-Wave machine to perform the calculation. Because of physical limitations of the hardware, embedding algorithms must be used. Note that when $\alpha = \beta$, the graph complement is mapped onto the *Chimera graph*. This is particularly helpful for dense graphs in reducing the number of qubits and couplers required for the embedding.

1.3 k-Concurrent Approach for Graph Partitioning

The k-Concurrent approach allows us to partition a graph into k parts without recursion. This is a general approach that can be applied to clustering or partitioning. Similar to the graph coloring problem [11, 12, 27] each graph vertex is represented by a super-node consisting of k subnodes, where k is the number of partitions desired (see Fig. 1). Each of the k subnodes has a unary encoding (either "0" or "1"). After GP, only one of the subnodes is set to "1", while the rest are "0" for each vertex denoting which part it belongs to.

Adjacent vertices are given different colors (or parts) in the graph coloring problem. In contrast, partitioning influences adjacent vertices to be in the same part. Super-nodes are connected by super-edges, pairing corresponding subnodes. Within each super-node, subnodes are constrained such that only one will be turned on as in the graph coloring problem [11, 12]. Using k-Concurrent GP results in k copies of the graph in parallel as a $kN \times kN$ matrix to represent the partitioning problem.

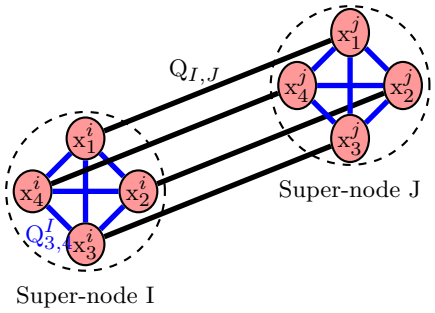

Figure 1: An example of the super-node concept used in k-Concurrent GP is shown for partitioning into 4 parts. Two super-nodes I and J consisting of four subnodes each are connected by a super-edge $\mathbf{Q}_{I,J}$. Internal edges $\mathbf{Q}^{I/J}_{l,m}$ where $l, m \in \{1-4\}$ are set to enforce the selection of only one subnode to be equal to "1" after GP. The super-edge $\mathbf{Q}_{I,J}$ is shown with connections between corresponding subnodes.

We now give a precise QUBO formulation for partitioning into k parts concurrently. In order to partition a graph concurrently into an arbitrary number of parts, k, we need to formulate the problem as a QUBO, which takes on binary variables. The authors in [6] provide different mathematical formulations for graph partitioning. In this section, we make a slight modification to the standard formulation and reformulate it as a quadratic program, which generalizes the above formulation of partitioning into 2 parts. We then relax the mathematical formulation as a QUBO written in matrix form.

The decision variables are given by $x_{i,j} = 1$ if node i is in part j and 0 otherwise. The constraint $\sum_{j=1}^{k} x_{i,j} = 1$ for each node i ensures that each node is in exactly one part. While $\sum_{i=1}^{n} x_{i,j} = \frac{n}{k}$ for $j = 1, \ldots, k$ are the balancing constraints for the part sizes. If we define $\mathbf{x}_j$ as $\mathbf{x}_j^T = (x_{1,j} x_{2,j} \ldots x_{n,j})$ The number of cut edges across the k parts is given by $\frac{1}{2}\left(\sum_{j=1}^{k} \mathbf{x}_j^T L \mathbf{x}_j\right)$ where L is the Laplacian matrix. Thus, giving us a generalization

$$\begin{aligned} & \min_{\mathbf{x}} \left(\textstyle\sum_{j=1}^{k} \mathbf{x}_j^T L \mathbf{x}_j\right) \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{i,j} = \frac{n}{k} \qquad j = 1, \ldots, k \\ & \sum_{j=1}^{k} x_{i,j} = 1 \qquad i = 1, \ldots, n \\ \text{with} \quad & x_{i,j} \in \{0, 1\}, \qquad i = 1, ..., n.\ j = 1, \ldots, k \end{aligned} \tag{11}$$

A relaxation of the above formulation is given by

$$\begin{aligned} & \min \quad \beta\left(\textstyle\sum_{j=1}^{k} \mathbf{x}_j^T L \mathbf{x}_j\right) + \sum_{j=1}^{k} \alpha_j \left(\sum_{i=1}^{n} x_{i,j} - \frac{n}{k}\right)^2 \\ & \quad + \sum_{i=1}^{n} \gamma_i \left(\sum_{j=1}^{k} x_{i,j} - 1\right)^2 \\ & \text{with} \quad x_{i,j} \in \{0, 1\},\ i = 1, ..., n.\ j = 1, \ldots, k \end{aligned} \tag{12}$$

where β, α_i and γ_i are positive penalty constants. The relaxation in (12) is a QUBO formulation and can easily be written in matrix notation.

2 RESULTS AND DISCUSSION

Our GP experiments on the D-Wave machine used software tools such as *sapi* Python [10] for graphs that fit onto the architecture (up to $\sim$ 70 verices) and the hybrid classical-quantum *qbsolv* [4] for larger graphs (up to $\sim$9000 vertices). NetworkX [21] was used for generating and processing graphs as part of this work. Random graph models (e.g. Erdos-Renyi, PowerLaw), large graphs from the Walshaw Archive [41, 42], and QMD molecule electronic structure graphs [13] were used to evaluate our methods. The quality of the GP was evaluated by a comparison metric as the number of *cut edges* between partitions (smaller is better). The results were compared to existing multi-level GP frameworks, METIS [24] and KaHIP [39] (winner of the 10th DIMACS challenge), or in some cases the best known solution from the Walshaw Archive [42]. KaHIP is a family of graph partitioning programs. In this work, we used KaFFPa with its preconfiguration variant set to `strong` and KaFFPaE with the setting `-mh_enable_kabapE` designed for *perfectly balanced* partitioning. We set the running time in these experiments to 600 seconds.

When running experiments using *sapi* directly on the D-Wave computer, multiple solutions are returned as a histogram, starting with the lowest energy. GP experiments using *qbsolv* return the lowest energy solution.

2.1 Community Detection with Thresholding

In Table 1 we show the results of modularity values for clustering into two communities using the D-Wave system. In this case we applied community detection to the karate club graph (34 vertices) and used *qbsolv* to solve the problem on the *chimera graph*. We explored thresholding of the modularity matrix. In this case we can see that with a threshold value of 0.12 meaning that we set $B_{ij} = 0$ if $B_{ij} < 0.12$, modularity is reduced by ~ 30 % but the required number of edges is significantly less (~ 65 % less) which has direct consequence on the number of qubits/couplers required for embedding on the D-Wave hardware. The reduction in the number of edges in both cases will result in a reduction of the number of qubits/couplers that are needed to embed the matrix into the QPU of the D-Wave machine. Preprocessing of the modularity matrix to produce a sparse version that is still representative of the original allows for larger problems to be solved efficiently by community clustering using quantum annealing.

Table 1: Graph 2-Clustering with Thresholding

Threshold	# Edges	Modularity
0	561	0.37179487
0.02	544	0.37179487
0.05	411	0.37146614
0.07	300	0.37146614
0.08	244	0.27714497
0.10	227	0.27714497
0.11	212	0.25509533
0.12	194	0.25509533
0.13	169	0

2.2 Graph Partitioning

For partitioning a graph into 2 parts of equal or similar size, we studied two sets of graphs chosen in order to determine the limits of the different partitioning methods used. The first set contained graphs of a relatively small size, having at most 70 vertices. The second set consisted of graphs with at most 9000 vertices chosen from the graph partitioning archive by Chris Walshaw [41, 42], an online archive dedicated to documenting the best quality solutions from a set of benchmark graphs.

For the first set, we partitioned the graphs using *sapi* as our main interface to the D-Wave machine. Thus, 100 percent of the partitioning was carried out by the quantum annealer. We generated random graphs (Erdos-Renyi) with a variable probability parameter p (fraction of total possible edges). The largest complete graph that is fully embeddable on the D-Wave 2X has approximately 45 vertices. However, our results demonstrate that for large values of p, (i.e., dense graphs) we can successfully partition such graphs even if they have more than 45 vertices. This was possible due to our use of the graph complement in the QUBO formulation. In our experiments, we partitioned graphs with up to 70 vertices. All our results gave solutions with a comparable quality to the solvers METIS and KaHIP. Table 2 shows the results for this first set of graphs. For the second set, we partitioned the graphs using the hybrid classical-quantum *qbsolv* as our main interface to the D-Wave machine. This enabled us to partition graphs with over 100 vertices. Our solutions for the Walshaw benchmark graphs, shown in Fig. 3, gave high quality partitions, in all cases having a cut size smaller than METIS and comparable to KaHIP. Our solutions matched the best known for the last 2 graphs. For partitioning a graph into $k = 2^i$ parts for $i > 1$, we performed our experiments on molecule electronic structure graphs as density matrices generated from QMD simulations [13]. As an example, Fig. 2 shows the phenyl dendrimer molecule composed of 22 covalently bonded phenyl groups with C and H atoms only with a total of 262 atoms. The electronic structure of this molecule consists of 730 orbitals, resulting in a graph of 730 vertices. This molecule is a special case, where the associated graph has a fractal-like structure and represents a difficult case for GP.

Table 2: Graph 2-Partitioning using *sapi*

N	METIS	KaFFPa	*sapi*
simulator			
20	82	82	82
30	183	182	182
40	326	324	330
D-Wave 2X			
40	334	334	334
60	766	765	768
70	1039	1042	1045

Table 3: Graph 2-Partitioning for Walshaw benchmarks using *qbsolv*

Graph	N	Best	METIS	KaFFPaE	*qbsolv*
add20	2395	596	723	613	602
data	2851	189	225	189	191
3elt	4720	90	91	90	90
bcsstk33	8738	10171	10244	10171	10171

2.3 k-Concurrent Graph Partitioning

k-Concurrent GP requires that a graph of size N (number of nodes) $\times$ k (number of partitions) be embedded in the *Chimera graph*. This quickly uses up the available qubits. Results of running concurrent GP on small random graphs on the D-Wave QPU only using *sapi* are shown in Table 4. The number of *cut edges* between partitions are shown.

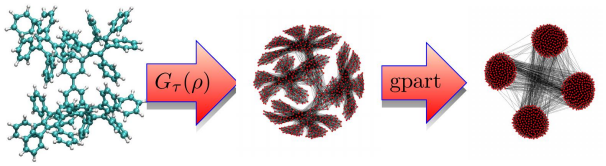

Figure 2: From left to right: Phenyl dendrimer molecular structure composed of 22 covalently bonded phenyl groups with C and H atoms only shown in cyan and white colors respectively. This molecule consists of 262 atoms with 730 orbitals. We also show the resulting electronic structure graph $G_\tau(\rho)$ constructed from the QMD density matrix ρ using a threshold τ that leads to 730 vertices (see reference [13]); and the resulting 4-Concurrent GP using the D-Wave machine where nodes are grouped by which part they belong to.

Concurrent GP has been formulated to produce equal sized partitions. Partition sizes never differ by more than one. In this case, METIS comparisons are contrained to equal sized partitions or very close (using the -ufactor=1 option). Results using *sapi* are comparable to METIS and *qbsolv*. Running directly on the D-Wave machine using *sapi* limits the the embeddable graph size to ∼45 nodes. A 15 node graph partitioned into 4 parts used almost all available qubits. Similarly for 20 nodes split into 3 parts. We can see that running k-Concurrent GP on the QPU produces quality results. k-Concurrent GP on large graphs requires the use of the hybrid classical-quantum *qbsolv*. Results for random dense graphs of 250, 500, and 1000 nodes split into 2, 4, 8, and 16 parts are shown in Table 5. The quality of the partitionings is comparable to METIS, while the number of *cut edges* is consistently reduced by tens to hundreds. In Table 6 the results for k-Concurrent GP of the molecule electronic structure graphs are shown. Fig. 2 shows the molecular structure for the Phenyl dendrimer, followed by the electronic structure graph from the QMD density matrix, and the resulting 4-Concurrent GP. Due to the fractal structure, this is a difficult case for GP. The *qbsolv* results are comparable or better than METIS for the Phenyl dendrimer and the Peptide 1aft. We see a very large number of *cut edges* for METIS when 4-partitioning the Phenyl Dendrimer due to the constraint on equal partitions.

Table 4: k-Concurrent Graph Partitioning using *sapi*. We show the number of *cut edges* between partitions.

N	k	*sapi*	METIS	*qbsolv*
10	2	19	19	19
10	3	29	29	29
10	4	32	33	32
11	2	25	26	25
11	3	36	36	36
11	4	38	39	38
15	2	45	47	45
15	3	62	62	62
15	4	70	73	70
20	2	83	83	83
20	3	120	122	120
23	2	109	114	109
27	2	156	164	156
30	2	182	183	182

Table 5: k-Concurrent Graph Partitioning using *qbsolv*

N	k	METIS	*qbsolv*
250	2	13691	13600
	4	20884	20587
	8	24384	24459
	16	26224	26176
500	2	55333	54999
	4	83175	83055
	8	98073	97695
	16	105061	105057
1000	2	221826	221420
	4	334631	334301
	8	392018	392258
	16	421327	420970

Table 6: k-Concurrent Graph Partitioning of molecule electronic structure graphs using *qbsolv*

k	METIS	*qbsolv*
Phenyl dendrimer N=730		
2	706	706
4	20876	2648
8	22371	15922
16	28666	26003
Peptide 1aft N=384		
2	12	12
4	29	20
8	121	66
16	209	180

3 CONCLUSION

We have shown that GP framed as a QUBO problem is a natural fit for the D-Wave quantum annealer. Our results using quantum and hybrid classical-quantum approaches are comparable to traditional GP tools. We showed that solving CD using a thresholded modularity matrix did not change the community results and could be run in a reduced qubit/coupler footprint. 2-partitioning of the Walshaw archive graphs and random graphs equaled or outperformed existing GP tools and in some cases equaled the best known results. A k-Concurrent GP approach using the super-node concept based on the map coloring problem was used to partition random graphs and molecule electronic structure graphs into 2, 4, 8, and 16 parts, improving the quality of the partitioning over existing tools by reducing the number of *cut edges* between

parts by tens to hundreds. k-Concurrent GP was shown to run on the QPU directly for small graphs and using hybrid classical-quantum *qbsolv* for large graphs.

Quantum annealing GP approaches were shown to produce quality partitions for example graphs as well as electronic structure graphs from QMD simulations. Future plans include applying k-Concurrent GP to other domains and extending the k-Concurrent approach to CD for discovering communities in graph structure problems.

ACKNOWLEDGMENTS

The authors would like to acknowledge D-Wave Systems for their useful tutorials and use of the Burnaby D-Wave machine. The authors would also like to acknowledge the NNSA's Advanced Simulation and Computing (ASC) program at Los Alamos National Laboratory (LANL) for use of their Ising D-Wave 2X quantum computing resource. The authors would also like to thank the developers of KaHIP, in particular, Christian Schulz for his useful suggestions in using their programs. This research has been funded by the Los Alamos National Laboratory (LANL) Information Science and Technology Institute (ISTI) and Laboratory Directed Research and Development (LDRD). Assigned: Los Alamos Unclassified Report 17-23649. LANL is operated by Los Alamos National Security, LLC, for the National Nuclear Security Administration of the U.S. DOE under Contract DE-AC52-06NA25396.

REFERENCES

[1] Tameem Albash, Troels F Rønnow, Matthias Troyer, and Daniel A Lidar. 2015. Reexamining classical and quantum models for the D-Wave One processor. *The European Physical Journal Special Topics* 224, 1 (2015), 111–129.

[2] Sergio Boixo, Troels F Rønnow, Sergei V Isakov, Zhihui Wang, David Wecker, Daniel A Lidar, John M Martinis, and Matthias Troyer. 2014. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics* 10 (2014), 218–224. Issue 3.

[3] Sergio Boixo, Vadim N Smelyanskiy, Alireza Shabani, Sergei V Isakov, Mark Dykman, Vasil S Denchev, Mohammad H Amin, Anatoly Yu Smirnov, Masoud Mohseni, and Hartmut Neven. 2016. Computational multiqubit tunnelling in programmable quantum annealers. *Nature communications* 7 (2016).

[4] Mike Booth, Steven P. Reinhardt, and Aidan Roy. 2017. Partitioning Optimization Problems for Hybrid Classical/Quantum Execution. *D-Wave Technical Report Series* 14, 1006A-A (2017), 1–9. http://github.com/dwavesystems/qbsolv

[5] Tomas Boothby, Andrew D King, and Aidan Roy. 2016. Fast clique minor generation in Chimera qubit connectivity graphs. *Quantum Information Processing* 15, 1 (2016), 495–508.

[6] Marc Boulle. 2004. Compact mathematical formulation for graph partitioning. *Optimization and Engineering* 5, 3 (2004), 315–333.

[7] Sergey Brin and Lawrence Page. 2012. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks* 56, 18 (2012), 3825–3833.

[8] Aydin Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2013. Recent Advances in Graph Partitioning. *CoRR* abs/1311.3144 (2013).

[9] Jun Cai, William G Macready, and Aidan Roy. 2014. A practical heuristic for finding graph minors. *arXiv preprint arXiv:1406.2741* (2014).

[10] D-Wave-Systems. 2016. Developer Guide for Python, Release 2.4. *D-Wave Technical Report Series* 09, 1024A-B (2016), 1–71.

[11] Edward D. Dahl. 2013. Programming with D-Wave: Map Coloring Problem. *D-Wave Systems Whitepaper* (2013), 1–12.

[12] Edward D. Dahl. 2016. D-Wave Quantum Programming Tutorial. *2016 LANL D-Wave Tutorials* (2016).

[13] Hristo N. Djidjev, Georg Hahn, Susan M. Mniszewski, Christian F.A. Negre, Anders M.N. Niklasson, and Vivek B. Sardeshmukh. [n. d.]. Graph Partitioning Methods for Fast Parallel Quantum Molecular Dynamics. In *2016 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing*. 42–51. https://doi.org/10.1137/1.9781611974690.ch5

[14] Irving Fatt et al. 1956. The network model of porous media. (1956).

[15] Charles M Fiduccia and Robert M Mattheyses. 1988. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*. ACM, 241–247.

[16] Michael R Garey, David S. Johnson, and Larry Stockmeyer. 1976. Some simplified NP-complete graph problems. *Theoretical computer science* 1, 3 (1976), 237–267.

[17] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[18] Timothy D Goodrich, Blair D Sullivan, and Travis S Humble. 2017. Optimizing Adiabatic Quantum Program Compilation using a Graph-Theoretic Framework. *arXiv preprint arXiv:1704.01996* (2017).

[19] Andreas Grönlund. 2006. *Complex patterns: from physical to social interactions*. Ph.D. Dissertation. Umeå University.

[20] Martin Haenggi. 2010. Interference in lattice networks. *arXiv preprint arXiv:1004.0027* (2010).

[21] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Laboratory (LANL). 11–16 pages. http://networkx.github.io/

[22] Bruce Hendrickson and Tamara G Kolda. 2000. Graph partitioning models for parallel computing. *Parallel Comput.* 26, 12 (2000), 1519 – 1534. Graph Partitioning and Parallel Computing.

[23] Laurent Hyafil and Ronald L Rivest. 1973. *Graph partitioning and constructing optimal decision trees are polynomial complete problems*. IRIA. Laboratoire de Recherche en Informatique et Automatique.

[24] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20, 1 (1998), 359–392.

[25] Brian W Kernighan and Shen Lin. 1970. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal* 49, 2 (1970), 291–307.

[26] Masato Kobayashi and Hiromi Nakai. 2011. *Linear scaling techniques in computational chemistry*. Vol. 13. Springer.

[27] Gary A Kochenberger, Fred Glover, Bahram Alidaee, and Cesar Rego. 2005. An unconstrained quadratic binary programming approach to the vertex coloring problem. *Annals of Operations Research* 139, 1 (2005), 229–241.

[28] Trevor Lanting, AJ Przybysz, A Yu Smirnov, Federico M Spedalieri, Mohammad H Amin, Andrew J Berkley, R Harris, Fabio Altomare, Sergio Boixo, Paul Bunyk, et al. 2014. Entanglement in a Quantum Annealing Processor. *Phys. Rev. X* 4 (May 2014), 021041. Issue 2.

[29] Los Alamos National Laboratory. 2016. Not Magic... Quantum. *1663 Magazine* (2016), 14–19.

[30] Andrew Lucas. 2013. Ising formulations of many NP problems. *arXiv preprint arXiv:1302.5843* (2013).

[31] Catherine C. McGeoch. 2014. Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice. *Synthesis Lectures on Quantum Computing* 5 (July 2014), 1–93. Issue 2. https://doi.org/10.2200/S00585ED1V01Y201407QMC008

[32] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[33] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 2 (2004), 026113.

[34] Anders M. N. Niklasson, Susan M. Mniszewski, Christian F. A. Negre, Marc J. Cawkwell, Pieter J. Swart, Jamal Mohd-Yusof, Timothy C. Germann, Michael E. Wall, Nicolas Bock, Emanuel H. Rubensson, and Hristo Djidjev. 2016. Graph-based linear scaling electronic structure theory. *Journal of Chemical Physics* 144, 23, Article 234101 (2016).

[35] MA Novotny, Q L Hobl, JS Hall, and K Michielsen. 2016. Spanning Tree Calculations on D-Wave 2 Machines. In *Journal of Physics: Conference Series*, Vol. 681. IOP Publishing, 012005.

[36] Milan Randić and Dejan Plavsić. 2003. Characterization of molecular complexity. *International Journal of Quantum Chemistry* 91, 1 (2003), 20–31.
[37] Ivan Rivalta, Mohammad M Sultan, Ning-Shiuan Lee, Gregory A Manley, J Patrick Loria, and Victor S Batista. 2012. Allosteric pathways in imidazole glycerol phosphate synthase. *Proceedings of the National Academy of Sciences* 109, 22 (2012), E1428–E1436.
[38] Peter Sanders and Christian Schulz. 2012. High quality graph partitioning. In *Graph Partitioning and Graph Clustering*. 1–18.
[39] Peter Sanders and Christian Schulz. 2013. Think locally, act globally: Highly balanced graph partitioning. In *Proceedings of the 12th International Symposium on Experimental Algorithms*. 164–175.
[40] Ricard V Solé and Sergi Valverde. 2004. Information theory of complex networks: on evolution and architectural constraints. In *Complex networks*. Springer, 189–207.
[41] Alan J Soper, Chris Walshaw, and Mark Cross. 2004. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. *Journal of Global Optimization* 29, 2 (2004), 225–241.
[42] Chris Walshaw. 2016. The Graph Partitioning Archive. (2016). http://chriswalshaw.co.uk/partition/
[43] Kenneth M Zick, Omar Shehab, and Matthew French. 2015. Experimental quantum annealing: case study involving the graph isomorphism problem. *Sci. Rep.* 5 (2015).