On Modeling Local Search with Special-Purpose Combinatorial Optimization Hardware

Xiaoyuan Liu[†*] Hayato Ushijima-Mwesigwa[†] Avradip Mandal[†]
Sarvagya Upadhyay[†] Ilya Safro[*] Arnab Roy[†]

Abstract

Many combinatorial scientific computing problems are NP-hard which in practice requires using heuristics that either decompose a large-scale problem and solve many smaller local subproblems in parallel or iteratively improve the solution by local processing for sufficiently many steps to achieve a good quality/performance trade-off. These subproblems are NP-hard by themselves and the quality of their solution that depends on the subproblem size vitally affects the global solution of the entire problem. As we approach the physical limits predicted by Moore's law, a variety of specialized combinatorial optimization hardware (such as adiabatic quantum computers, CMOS annealers, and optical parametric oscillators) is emerging to tackle specialized tasks in different domains. Many researchers anticipate that in the near future, these devices will be hybridized with high-performance computing systems to tackle large-scale problems which will open a door for the next breakthrough in performance of the combinatorial scientific computing algorithms. Most of these devices solve the Ising combinatorial optimization model that can represent many fundamental combinatorial scientific computing models. In this paper, we tackle the main challenges of problem size and precision limitation that the Ising hardware model typically suffers from. Although, our demonstration uses one of these novel devices, the models we propose can be generally used in any local search and refinement solvers that are broadly employed in combinatorial scientific computing algorithms.

1 Introduction

Within the field of combinatorial optimization, driven by the physical limitations arising as a corollary of Moore's law [52], various research groups and institutions have started to develop novel hardware specifically designed for combinatorial optimization. Combinatorial scientific computing (CSC) as a field is expected to be one of the main beneficiaries of such hardware. The most realistic way to employ it would be through hybridization with traditional high-performance computing (HPC) systems. The main scientific computing task will be performed on HPC while its CSC accelerators and preprocessing algorithms will use that novel hardware. Examples of such hardware include adiabatic quantum computers [26], CMOS annealers [1, 64, 65] and Coherent Ising Machines [32, 38, 25]. Problems such as accelerating solutions of linear systems and least squares problems [9, 10, 5] in the spirit of CSC have been performed on quantum annealers where the solution is achieved by embedding the error function in the Ising Hamiltonian. The D-Wave hardware has also been used for traditional CSC goals to facilitate matrix inversion and factorization problems [44, 18, 47]. Although these emerging technologies exhibit novelty in terms of hardware, they are all unified by the mathematical framework of the Ising optimization model. In other words, they are all designed to solve optimization problems formulated in the Ising model or equivalently as a quadratic unconstrained binary optimization (QUBO) problem. According to [19, 34], these recent hardware advances enable the Ising or QUBO model to become a

[*]School of Computing, Clemson University.
[†]Fujitsu Laboratories of America, Inc.

unifying framework for combinatorial optimization. They have recently been termed as *Ising Processing Units* (IPU)'s [12]. Many fundamental CSC problems can be formulated as QUBO for IPU's. Examples include partitioning [61], clustering [58, 57], coloring [19], scheduling [60, 63], traveling salesman problem [45], maximum cliques [11], and vertex separator [19] as well as the combinatorial problems that directly model scientific computing tasks such as computing molecular similarity [23, 24], and protein structure identification [43].

The focus on development of specialized hardware to solve QUBO stems from the many advantages that QUBO offers which in fact go way beyond CSC tasks (e.g., finance [48], and machine learning [13, 22, 31, 40, 35]). The abstraction allows a cleaner and simpler mathematical framework to study a variety of problems arising in different disciplines. Many NP-hard CSC problems can be easily and efficiently reformulated in QUBO. While some of them admit integer programming formulation that can be converted to QUBO using standard techniques, others are necessarily formulated as QUBO. Secondly, unsupervised learning techniques such as spectral clustering, statistical neural models [53] social network problems like community detection can be naturally cast as QUBO. The model's ubiquity and ability to represent a wide range of scientific problems makes development of specialized hardware for solving it a fruitful academic as well as industrial endeavor.

Challenges with local search, QUBO, and IPU Many CSC problems are NP-hard which in practice requires using heuristics that either decompose a large-scale problem and solve many smaller problems in parallel or iteratively improve some feasible solution for sufficiently many steps. Algorithms such as Kernighan-Lin [30], Fiduccia-Mattheyses [17], 2-sum window minimization [49], and max-flow min-cut refinement [51] are among many other relevant examples applied in CSC solvers. All these can be formulated as versions of a local search (or improvement) strategy that gradually improves a solution. With the invent of IPUs, and anticipating a hybridization of IPU and HPC systems, we envision that such local search problems (which are themselves NP-hard) will be solved on IPU devices using QUBO formulation. Typically, the larger a local search subproblem is the better it can affect the global solution. However, increasing the size of subproblem comes with a price tag of its representation as a QUBO on an IPU.

Some of the well-known challenges of QUBO hardware include limited precision and the maximum number of variables the hardware can handle. For instance, D-Wave's 2000Q quantum annealer, even with up to 2000 qubits, can only handle arbitrary fully-dense QUBO of maximum 64 variables. Fujitsu's latest Digital Annealer can handle up to approximately 8000 variables. This brings into perspective that current optimization problems arising from real-world challenges can easily have millions of variables. Subsequently, this motivates the investigation of the algorithmic challenge of how to utilize these emerging technologies to efficiently solve large scale problems. Another challenge lies inherently within the QUBO model. Many problems in practice are coupled with multiple constraints; however, QUBO are unconstrained by definition. The natural strategy to turn a constrained problem into an unconstrained one is to introduce quadratic penalty terms (corresponding to each constraint) to the objective function. These penalties are introduced such that (i) if the constraint is satisfied, their contribution to objective function is exactly zero; and (ii) if the constraint is not satisfied, their contribution to objective function is negative/ positive for maximization/ minimization formulation. This leads to the need to tune the coefficients of the penalty terms. A small penalty term would easily lead to a violation of the constraint while a very large term can lead to difficulties in comparison of quality of feasible solutions, especially when bounded by the limited precision of the hardware. Another consequence of this is that not all feasible solutions to the QUBO formed would be a feasible solution to the original problem. Thus for some problems, finding a feasible solution via the QUBO model may become difficult, yet finding a feasible solution in their original formulation may be trivial. For example, problems arising from permutations of size n lead to the formulation of a QUBO of size $O(n^2)$. This QUBO would have 2^{n^2}

candidate solutions, however only $n!$ of them would be a candidate permutation. Since

$$\frac{n!}{2^{n^2}} \sim \sqrt{2\pi n}\left(\frac{n}{2^n e}\right)^n < \left(\frac{n}{2^n}\right)^n \longrightarrow 0$$

as $n \to \infty$, where $\sim$ means that the two values are asymptotic, we see that a candidate solution of the QUBO is not a permutation with high probability. Thus a straight-forward QUBO formulation of a permutation may not lead to the the ideal utilization of this emerging technology.

Our Contribution We develop different models and techniques for special-purpose hardware for large-scale combinatorial optimization often used in CSC. Similar to previous methods [56, 57, 58], for problems larger than current hardware sizes, we advocate the use of a local search framework to create sequence of sub-problems that can be solved with hardware of limited size. Our novel modeling technique also addresses the limitations of QUBO framework and demonstrates efficient use of the hardware.

Our experiments are carried out using Fujitsu's latest Digital Annealer. In order to show the differences in modeling methods, we compare our approach to previous methods on the quadratic assignment problem (QAP). In addition, we give models for local search technique for the traveling salesman problem (TSP) and the graph partitioning problem (GP) on IPU. Because of currently limited availability of this novel hardware, we cannot demonstrate the approach on really large CSC problems. However, we anticipate that in the near future, availability of these devices in combination with HPC will play an important role in breaking the barriers of CSC solvers and this type of modeling will be broadly applicable. Our multi-variable type of local search modeling can also replace typical local search algorithms using regular global or specialized solvers by increasing the number of variables in the subproblem, i.e., without employing IPU.

2 Background

2.1 Ising Model Ising model is a common mathematical abstraction which has been widely used in physics. In this class of graphical models, the nodes $\mathcal{N}$ represent discrete spin variables (i.e., $\sigma_i \in \{-1, 1\}, \forall i \in \mathcal{N}$), and the edges $\mathcal{E}$ represent the interactions of spin variables (i.e., $\sigma_i\sigma_j, \forall (i,j) \in \mathcal{E}$). For each node, a local field $h_i, i \in \mathcal{N}$ is specified, and for each edge, an interaction strength $J_{ij}, \forall (i,j) \in \mathcal{E}$ is specified. The energy of a configuration σ is given by the Hamiltonian function:

$$H(\sigma) = \sum_{(i,j)\in\mathcal{E}} J_{ij}\sigma_i\sigma_j + \sum_{i\in\mathcal{N}} h_i\sigma_i \tag{2.1}$$

The most common applications of the Ising model is to find the lowest possible energy of the model, namely, to find the configuration σ that minimizes the Hamiltonian function (2.1). Note that with a transformation of the variables, $\sigma_i = 2x_i - 1, i \in \mathcal{N}$, where $x_i \in \{0, 1\}$, an Ising optimization problem is equivalent to a Boolean optimization problem.

As summarized in [12], the five most primary solution methods of Ising optimization are simulated annealing [33], Large Neighborhood Search [20, 54], integer Programming [15, 37, 46], adiabatic quantum computation (AQC) [16, 27], and quantum Monte Carlo [42]. Novel hardware are developed based on these methods, for example, the D-Wave quantum annealer is based on AQC, and Fujitsu's Digital Annealer is based on Simulated Annealing with parallel tempering.

2.2 Digital Annealer Fujitsu's Digital Annealer (DA) is a hardware accelerator for solving fully dense QUBO problems [1]. Internally the hardware runs a modified version of Metropolis-Hastings algorithm [21, 39] for simulated annealing. The hardware utilizes massive parallelization and a clever sampling technique. The novel sampling technique speeds up the traditional Markov Chain Monte Carlo (MCMC) method by almost always moving to a new state instead of being stuck in a local minimum. Digital Annealer also supports Parallel Tempering (replica exchange MCMC sampling) [59] which improves dynamic properties of the Monte Carlo method. In our experiments we used this mode, as it requires less parameter tuning and better for consistent benchmarking. We used the second generation of the DA also known as the *Digital Annealing Unit* (DAU), which supports eight thousand binary decision variables with up to 64 bits of precision for the individual entries of the weight matrix.

2.3 Related Work With respect to solving problems larger than the current hardware can accommodate, a large number of work has focused on formulating the original problem as a large QUBO and then using some decomposing technique to create sub-problems or sub-QUBOs that can be individually solved directly on the hardware. The tool qbsolv [4] developed by D-Wave systems is one such example. For example, with the limitation that the D-Wave 2X and D-Wave 2000Q quantum annealers can solve fully dense QUBO of up to approximately 45 and 64 variables respectively, researchers at Volkswagen [41] solved a traffic flow optimization problem that modeled traffic from 418 cars that required 1254 boolean variables. In [61, 40], problems in Graph Partitioning and Community Detection were solved for graphs larger than hardware size.

An alternative approach to solve problems larger than the hardware size is to first identify a subproblem, and then model this subproblem as a QUBO and solve with a given hardware. The authors in [56, 57, 58, 62] generally took this approach for solving the graph partitioning and community detection problem on available quantum computing hardware. In their work, they solved problems on the D-Wave quantum annealer and the gate-model IBM quantum computer by creating and solving smaller QUBOs. Their approach is referred to as *Quantum Local Search* (QLS). A straight forward extension of QLS applied to general problem does not take the limitations of the QUBO model into account. In this work, we consider modeling of sub-problems while taking limitations of the QUBO model into account.

3 Modeling Local Search

Local Search is a class of metaheuristic methods for solving large scale combinatorial optimization problems. A local search algorithm moves from one feasible solution to another by applying some local changes to the current solution. The use of local search in combinatorial optimization dates back to the 1950s when the first edge-exchange algorithms were introduced for the Travelling Salesman Problem [2, 14, 36]. Since then it has been broadened with various levels of success in different problems. The scaling of Moore's law together with the use of sophisticated data structures has made local search algorithms the state-of-the-art for many problems. With the introduction of the special-purpose hardware for combinatorial optimization, in a post-Moore's law era, hybrid techniques in combination with hardware that leverage the existing sophisticated data structures must be developed to fully utilize these technologies. However, since sub-problems within local search are not traditionally described with respect to Ising model, it remains unclear how well established algorithms can take leverage these technologies. In this section, we take steps to demonstrate the use of this technology. We do this by giving different ways in which a sub-problem in existing algorithms can be efficiently modeled as a QUBO. In particular, we give models for sub-problems from TSP, Graph Partitioning and QAP that are unconstrained by definition.

3.1 Modeling Constraints Many well known binary optimization problems are formulated with constraints. While modeling a given problem as a QUBO, one of the most common constraints are the constraints that are usually referred to as 1-Hot constraints or 1-Hot encodings. For n boolean variables, a 1-Hot encoding is simply the constraint that

$$\sum_{i=1}^{n} x_i = 1. \tag{3.2}$$

For n^2 boolean variables a 2-Way 1-Hot encoding are the constraints

$$\begin{aligned} \sum_{i=1}^{n} x_{i,j} &= 1, \quad j = 1, \ldots, n \\ \sum_{j=1}^{n} x_{i,j} &= 1, \quad i = 1, \ldots, n. \end{aligned} \tag{3.3}$$

The 2-Way 1-Hot encoding are particularly common because they model permutations which have common occurrence in combinatorial optimization problems.

If $\mathbf{x} = \{x_{i,j}\}_{1 \leq i,j \leq n}$ and $Q(\mathbf{x})$ is a quadratic function we want to minimize, subject to 2-Way 1-Hot constraints, it is well known, that for appropriate choice of positive constants λ_i and λ'_j 's, the above set of constraints can be encoded as a QUBO problem where the goal is to minimize

$$\begin{aligned} Q(\mathbf{x}) &+ \sum_{i=1}^{n} \lambda_i \Big(\sum_{j=1}^{n} x_{i,j} - 1 \Big)^2 \\ &+ \sum_{j=1}^{n} \lambda'_j \Big(\sum_{i=1}^{n} x_{i,j} - 1 \Big)^2. \end{aligned} \tag{3.4}$$

If the above problem forms a sub-problem in a local search framework, the 2-Way 1-Hot encoding would significantly reduce the feasible solution search space compared to if there were no such constraints. The formulations of the TSP and Graph Partitioning contain such constraints that reduce the feasible solution search space significantly. However, in the following sub-sections we use these two problems as examples where sub-problems can easily be developed that do not require any constraints and thus increases the search space the QUBO solver can search per iteration. We then give a general modeling technique where one can control the solution space searched per iteration of the QUBO solver. We present this general modeling technique with respect to the Quadratic Assignment Problem (QAP) mainly because a large class of problems form special cases of QAP.

3.2 Travelling Salesman Problem The travelling salesman problem is by far one of the most well known combinatorial optimization problems. Many algorithms, both heuristics and exact methods have been proposed. As such it's instructive to describe any new general approach with respect to TSP for ease of exposition. In QUBO representation, it has a formulation with n^2 variables which makes solving current real size TSP instances directly on near term hardware unlikely. If we cannot directly solve a large TSP instance with hardware, the next best thing is to accelerate current TSP algorithms and heuristics. Current state-of-the-art methods consist of using local search with sophisticated data structures. From the hardware perspective, improving the speed or quality of local search moves seems like the best option to enhance current methods. The k-opt heuristics is one of the most popular heuristics for solving TSP. However, a straight-forward implementation of k-opt would require at least k^2 variables due to the 2-way 1-hot encoding of the sub-problem. Within a QUBO model, adding constraints reduces the number of feasible solutions searched per iteration. In this sub-section we give an alternative model whose formulation does not require constraints thus can search at most up to 2^H feasible solutions per each iteration, where H is the hardware size.

***k*-opt Algorithm** k-opt is a well-known local search heuristic for the TSP. A k-opt move consists of removing k edges from a given tour and then reconnecting the k segments to possibly get a shorter tour. In this sub-section, we use the TSP as an easy example to give a QUBO formulation of a sub-problem that does not contain 1-Hot encoding although the main problem does. For ease of exposition, for a binary variable $y \in \{0,1\}$, we use the notation $\overline{y}$ such that

$$\overline{y} = 1 - y. \tag{3.5}$$

In the case of the 2-opt algorithm. A move consists of whether 2 edges in the tour should be replaced with 2 other edges. In an effort to motivate our modeling technique, we first model a move in 2-opt, and 3-opt as a QUBO. An equivalent and alternative way of stating a 2-opt move is as follows: remove two edges from the current tour, thus creating two disjoint paths which we shall refer to as segments. Then a 2-opt move is to decide whether or not to reverse one of these segments. A tour segment $(v_1, v_2, \dots, v_m)$ is *reversed* if it appears in the new tour in the reversed order $(v_m, v_{m-1}, \dots, v_1)$. Therefore a 2-opt move represented as a QUBO with one variable as

$$\begin{aligned} &\min\ (w_{u_1v_2} + w_{u_2v_1})y + (w_{u_1u_2} + w_{v_1v_2})\overline{y} \\ &\text{s.t. } y \in \{0,1\}. \end{aligned} \tag{3.6}$$

Figure 1 depicts a 2-opt and 3-opt move with respect to decision variables that constitute of reversing a tour segment or not.

We can then extend this approach for larger k and define a move as a k-reversal as shown in Figure 2. At each iteration, the hardware would reverse up to k segments of the tour. In general, since y_i is 1 if segment i is reversed and $\overline{y_i}$ is 1 if it is *not* reversed, then product $y_i\overline{y_j}$ is 1 if and only if segment i is reversed and segment j is not. Thus, the quadratic terms can only be in the form of either $\overline{y_i}\,\overline{y_j}, \overline{y_i}y_j, y_i\overline{y_j}$ or y_iy_j. Therefore, if

$$\begin{aligned} q(y_i, y_j) = {}& w_{v_iu_j}\overline{y_i}\,\overline{y_j} + w_{u_iu_j}y_i\overline{y_j} \\ &+ w_{v_iv_j}\overline{y_i}y_j + w_{u_iv_j}y_iy_j \end{aligned} \tag{3.7}$$

then $q(y_i, y_{i+1})$ and $q(y_{i-1}, y_i)$, contain all the quadratic terms in a k-reversal move with respect to y_i, for $i = 2, \dots, k-1$.

$$\begin{aligned} &\min\ q(y_k, y_1) + \sum_{i=1}^{k-1} q(y_i, y_{i+1}) \\ &\text{s.t.} \quad y_i \in \{0,1\} \quad i = 1, \dots, k \end{aligned} \tag{3.8}$$

In particular, all 3-opt moves are also 3-reversal moves. The main advantage of this model is that it does not model any constraints thus any feasible solution of the QUBO is a feasible solution of the TSP. In other words, at most 2^k feasible solutions are considered at each iteration of the local search. The major drawback of this approach is that it is not equivalent to a k-opt move. In particular, there are $(k-1)!2^{k-1}$ possible ways to reconnect k segments. However, out of all these moves, k-reversal only considers 2^k of them. With this in mind, we believe that this model gives the reader a more intuitive understanding on how to model sub-problems of a given problem. In the next subsection, we discuss about the Kernighan-Lin algorithm which is based on swapping nodes between parts. These moves can efficiently be modeled as a QUBO without modeling any constraints.

3.3 Graph Partitioning Problem The graph partitioning problem (GP) is another well known combinatorial optimization problem with many applications in CSC [6]. Formally, let $G = (V, E)$ be an

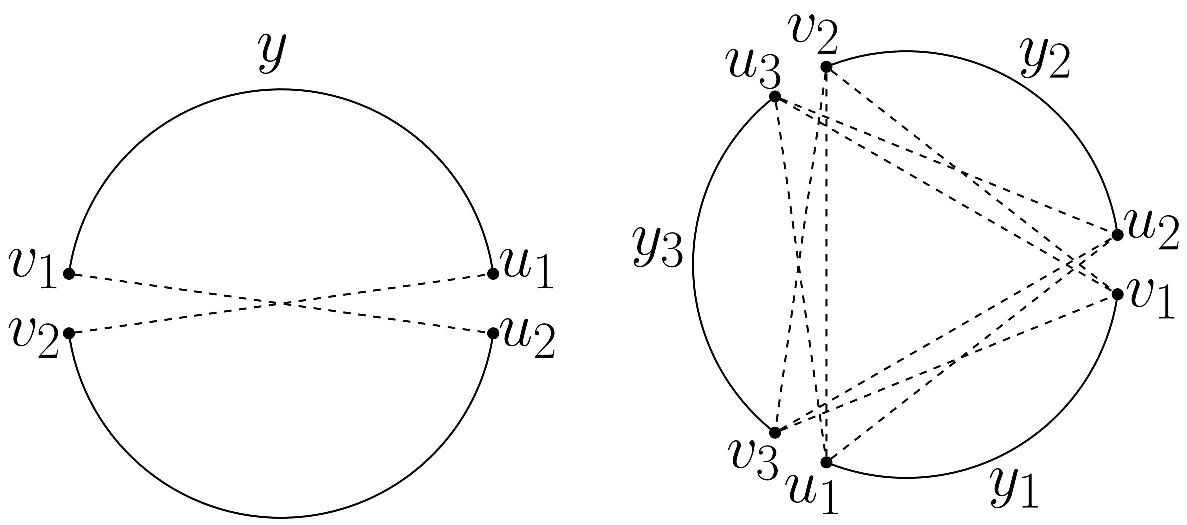

Figure 1: 2-Opt and 3-Opt sub-problems

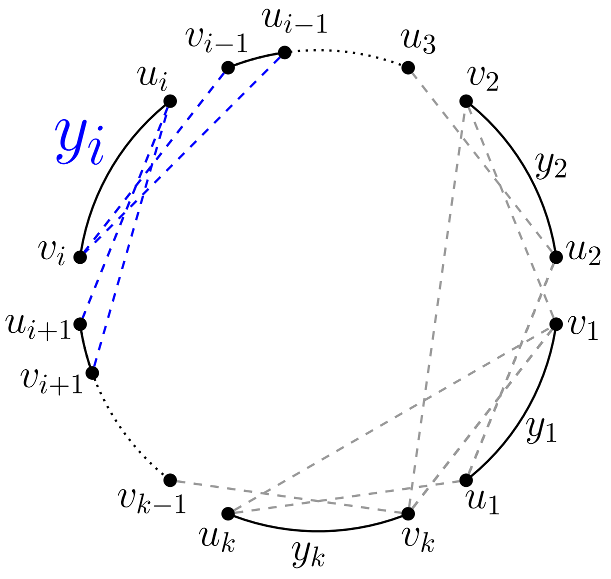

Figure 2: Sub-problem for TSP that reverses at most k segments of the tour. This sub-problem is purely an unconstrained problem thus does not require modeling any constraints as a QUBO. As such, all 2^k feasible solutions of the QUBO are also feasible solutions of the TSP

undirected graph of vertices V and edges E. Let $|V|$ denote the number of vertices of the graph, and w_{ij} be the weight of the edge with endpoints i and j. For a fixed integer K, the K-way GP is to find a partition $\Pi = (\Pi_1, \cdots, \Pi_K)$ of the vertices V into K equal parts such that the number of cut-edges is minimized, where a cut-edge is defined as an edge whose endpoints are in different partitions. The QUBO formulation for GP [61] is given as follows:

$$\begin{aligned} \min_{\Pi} \qquad & A\sum_{i=1}^{|V|}(\sum_{\ell=1}^{K} x_{i\ell} - 1)^2 + B\sum_{\ell=1}^{K}\left(\sum_{i=1}^{|V|} x_{i\ell} - \frac{|V|}{K}\right)^2 \\ + \quad & \sum_{(i,j)\in E}\sum_{\ell=1}^{K} w_{ij}(x_{i\ell} - x_{j\ell})^2 \end{aligned}$$

where A, B are large positive constants to penalize the violation of constraints. For a feasible solution, the binary variables $x_{i\ell}$ are interpreted as follows:

$$x_{i\ell} = \begin{cases} 1, & \text{if vertex } i \text{ is in partition } \ell \\ 0, & \text{otherwise.} \end{cases}$$

Kernighan-Lin Algorithm The Kernighan-Lin (KL) algorithm is a very popular algorithm for graph partitioning dating back to the seminal paper [30] and used in a variety of multilevel solvers [28, 50, 3, 55]. The KL algorithm is an iterative algorithm whose goal is to reduce the number of cut edges between two parts. The main concepts used in the algorithm can be described as follows. Let $|V| = 2n$, $V_1, V_2, \subset V$ such that $|V_1| = n = |V_2|$ and $V_1 \cap V_2 = \emptyset$. For $u \in V_1$ define

$$E_u = \sum_{v\in V_2} w_{uv}; \quad I_u = \sum_{v\in V_1} w_{uv} \tag{3.9}$$

as the *External* and *Internal* degree of node $u \in V_1$ respectively. Let

$$D_u = E_u - I_u \tag{3.10}$$

be the cost reduction of moving node $u \in V_1$ to V_2. We refer to this as the D-value of u. Then the cost reduction from swapping u and v is given by

$$g_{uv} = D_u + D_v - 2w_{uv}. \tag{3.11}$$

This is usually referred to as the *gain* of swapping. We can express a similar process in a QUBO formulation. Let $s : V_1 \to V_2$ be the one-to-one function that identifies a node in V_2 that will potentially be swapped given a node in V_1. Then let

$$M = \big\{\{u, s(u)\} | u \in V_1\big\} \tag{3.12}$$

For every $\{u, v\} \in M$, define the binary variable y_{uv} such that

$$y_{uv} = \begin{cases} 1, & \text{if } u \text{ swaps with } v \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

Since the function s is a one-to-one function, we simplify the notation and refer to the variable y_{uv} simply as y_u or y_v. In other words,

$$y_{uv} = y_{vu} = y_u = y_v \tag{3.14}$$

Thus y_v simply represents the variable associated to moving node v for any $v \in V$. Then we can write the external degree of a node $u \in V_1$ as

$$\begin{aligned} E_u = &\sum_{v \in V_2} w_{uv}(\overline{y_u}\,\overline{y_v} + y_u y_v) \\ &+ \sum_{v \in V_1} w_{uv}(y_u \overline{y_v} + \overline{y_u}\, y_v) \end{aligned} \tag{3.15}$$

Since the sum of the external degree for every node in V is in fact equal twice the cut, we thus have an optimal move as

$$\begin{aligned} \min \ &\sum_{u \in V} E_u \\ &y_u \in \{0,1\} \quad u \in V \end{aligned} \tag{3.16}$$

We now generalize this to give an optimal move in a k-way partitioning.

Let $P(u) \in \{1, \ldots, k\}$ be the part number of node $u \in V$, and M be the set of non-intersecting pairs of nodes being considered for the decision to be swapped, such that if $\{u_1, u_2\}, \{u_3, u_4\} \in M$, then $u_i \neq u_j$ for $i \neq j$ and if $\{u, v\} \in M$ then $P(u) \neq P(v)$, i.e, they are distinct and nodes in each pair belong to different parts. Similar to the bisection case, define $y_{uv} = y_{vu} = y_u = y_v$ as the binary variable that is 1 if and only if node u swaps parts with v. Define the *Neighborhood* of $u \in V$ as

$$N(u) := \{v \in V | P(u) = P(v)\} \tag{3.17}$$

and

$$\overline{N}(u) := \{v \in V | \{w, v\} \in M, w \in N(u)\} \tag{3.18}$$

then since each node is restricted to moving to only one other part, we can describe this move as a QUBO similar to the one in the graph bisection problem. In what follows, we follow the notation and treatment of KL algorithm as in the from the developers of METIS [29].

The external degree of a node $u \in V$ can be defined in terms of the disjoint sets $N(u)$ and $\overline{N}(u)$.

$$\begin{aligned} E_u = &\sum_{j \in N(u)} w_{uj}(y_u \overline{y_j} + \overline{y_u}\, y_j) \\ &+ \sum_{j \in \overline{N}(u)} w_{uj}(y_u y_j + \overline{y_u}\,\overline{y_j}) \\ &+ \sum_{j \in V \setminus N(u) \cup \overline{N}(u)} w_{uj} \end{aligned} \tag{3.19}$$

Then for a partition given by $V_1, \ldots, V_k$, the cut is given by

$$\frac{1}{2} \sum_{j=1}^{k} \sum_{u \in V_j} E_u \tag{3.20}$$

where the $\frac{1}{2}$ is added to include the double counting of each edge in the cut which can be ignored for optimization purposes. Therefore an optimal move of swaps would be

$$\begin{aligned} \min \ &\sum_{j=1}^{k} \sum_{u \in V_j} E_u \\ &y_u \in \{0,1\} \quad u \in V \end{aligned} \tag{3.21}$$

or simply as

$$\begin{aligned} \min \quad & \sum_{\{u,v\}\in M} E_u + E_v \\ & y_{uv} \in \{0,1\} \quad \{u,v\} \in M \end{aligned} \tag{3.22}$$

where the formulation given in (3.22) is independent from k. Note that if there exists a node $u \in V$ such that $u \notin m$ for any $m \in M$, then by definition, $y_u = 0$, thus not every node need to be considered to be move at each iteration of the algorithm.

3.4 Quadratic Assignment Problem Quadratic Assignment Problem (QAP) is one of the fundamental combinatorial optimization problems. In fact, many other famous problems including TSP, GP can all be formulated into QAP.

Consider a set of facilities $\mathcal{N} = \{1, \dots, n\}$ and a set of $\mathcal{N} = \{1, \dots, n\}$ locations. For each pair of facilities, a weight or flow is specified and for each pair of locations, a distance is specified. The quadratic assignment problem (QAP) is to assign all facilities to different locations with the goal of minimizing the allocation cost, taking the costs as the sum of all possible distance-flow products.

Consider f_{ij} the flow between facility i and facility j, and d_{kl} the distance between location k and location ℓ. The QAP can be formulated as the following QUBO:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{\ell=1}^{n} f_{ij} d_{k\ell} x_{ik} x_{j\ell} \\ & +A\sum_{k=1}^{n}(\sum_{i=1}^{n} x_{ik} - 1)^2 \\ & +A\sum_{i=1}^{n}(\sum_{k=1}^{n} x_{ik} - 1)^2 \\ \text{s.t.} \quad & x_{ik} \in \{0,1\}, i,k = 1,2,\cdots,n \end{aligned} \tag{3.23}$$

where A is a positive constant to penalize the violation of constraints.

For a feasible solution, the binary variables x_{ik} can be interpreted as follows:

$$x_{ik} = \begin{cases} 1, & \text{if facility } i \text{ is assigned to location } k \\ 0, & \text{otherwise.} \end{cases}$$

Note that this is a classical example of problem with 2-Way 1-Hot constraints, and each feasible solution is an encoding of a permutation.

It is not practical to directly solve the original problem exactly when the size of the problem gets larger, and this leads to the development of a large collection of local search algorithms.

Local Search Algorithm As summarized in [7], much research has been devoted to the development of local search heuristics to provide good quality solutions of QAP in a reasonable time. All such algorithms start with an initial feasible solution and iteratively refines the current solution. The most frequently used neighborhoods for QAPs are the pair-exchange neighborhood and the cyclic triple-exchange neighborhood.

A straight forward implementation of QLS in [56, 57, 58] would be: Given an initial feasible solution (permutation) of QAP, in each iteration, select a subset of the current permutation, formulate a sub-QUBO that search through possible permutations inside the subset. The sub-QUBO formulated in each iteration will have k^2 binary variables, where k is the size of the subset selected.

We first extend QLS by choosing multiple subsets that are pairwise disjoint. Given an initial feasible solution (permutation) of QAP, in each iteration, we first select m pairwise disjoint subsets of size k from the permutation (For the sake of simplicity, we assume that the subsets are equally sized, but in general they can be different in sizes). Next, we try to refine the current solution by finding the optimal allocation of each subset we select simultaneously, and we achieve this by formulating it into a QUBO and solve the QUBO. To make it more clear, We use figure 3.4 to further explain. In figure 3.4, there are 4 disjoint subsets, each with size 3, thus $m = 4, k = 3$, we allow facility f_1, f_2, f_3 to relocate to location l_1, l_2, l_3, facility f_4, f_5, f_6 to relocate to location l_4, l_5, l_6, etc.

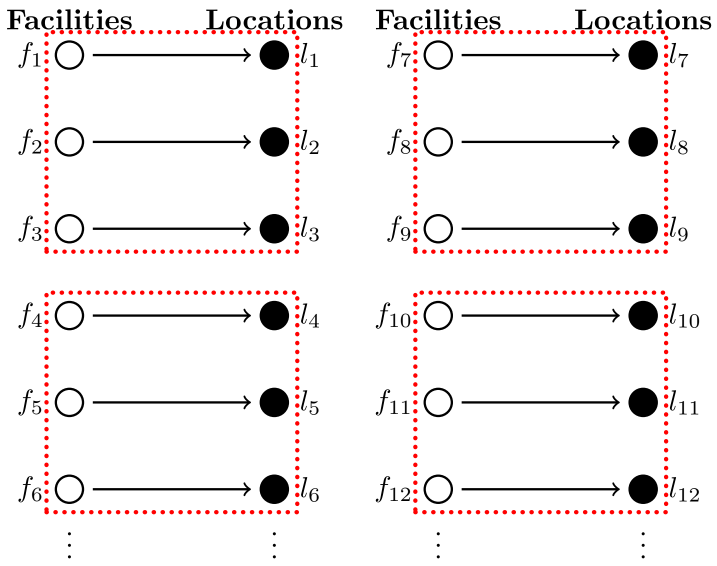

Figure 3: Multiple Local Search Simultaneously with $k = 3$

Notice that here, k determines the *degree of freedom* of each facility, with larger k, the search space for each facility gets larger, while m determines the breadth of the search, with larger m, we can update the allocation of more facilities simultaneously. In each iteration, we search for a better solution from $(k!)^m$ possible permutations.

Suppose in an iteration, the initial solution is a permutation $\pi : \mathcal{N} \to \mathcal{N}$, and we select refinement subsets $\mathcal{C} = \{C_j\}_{j\in[1,m]}$, where $C_j : \mathcal{N}_j \to \pi(\mathcal{N}_j), j \in [1, m]$ are pairwise disjoint. Here, $\mathcal{N}_j = \{n_{j\ell}\}_{\ell\in[1,k]}$ are subsets of the permutation, and we define $\pi(\mathcal{N}_j) = \{\pi(n_{j\ell})\}_{\ell\in[1,k]}, j \in [1, m]$. Since we only allow permutations inside each subset, the QUBO we formulate is equivalent to the formulation (3.23) with multiple variables fixed as zero. Namely, $x_{ik} = 0$ for all $i \in \mathcal{N}_j, k \notin \pi(\mathcal{N}_j), j = 1, 2, \cdots, m$. Therefore, the number of binary variables in (3.23) will be reduced to mk^2. However, in this formulation, we still have the 2-Way 1-Hot encoding.

Next, we consider the special case when $k = 2$, namely, each subset only contains 2 elements, as shown in Figure 3.4. There are only two possibilities for each subset, swap the current allocation or keep the current allocation. If we formulate this local search with binary variables interpreted as follows:

$$x_i = \begin{cases} 1, & \text{if apply pairwise exchange on } C_i \\ 0, & \text{otherwise.} \end{cases}$$

We can further reduce the number of variables to m, what's more, since the permutation constraints

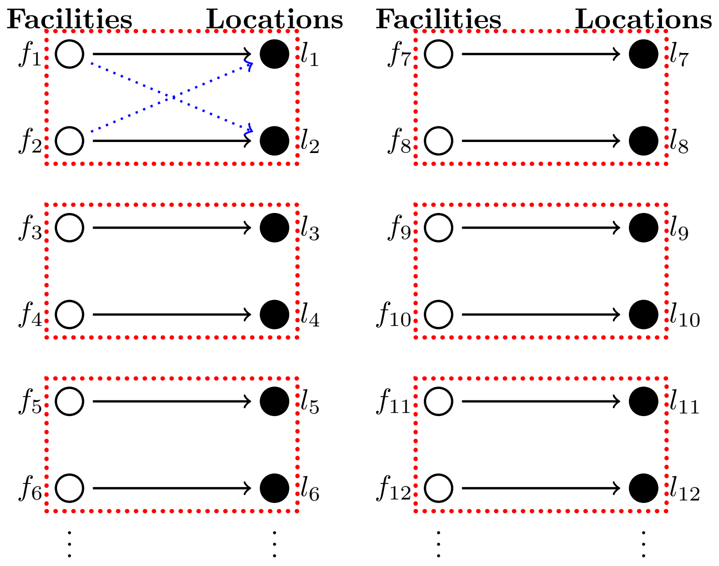

Figure 4: Multiple Local Search Simultaneously with $k = 2$

are automatically satisfied after the local change, the 2-Way 1-Hot encoding in (3.23) will be dropped. The QUBO formulation is given as follows:

$$
\begin{aligned}
\min \quad & \sum_{C_i, C_j \in \mathcal{C}} \Bigg(x_i \overline{x_j} \sum_{a,b \in N_i \cup N_j} f_{ab} d_{\pi_i(a)\pi_i(b)} \\
+ \quad & \overline{x_i}\, x_j \sum_{a,b \in N_i \cup N_j} f_{ab} d_{\pi_j(a)\pi_j(b)} \\
+ \quad & x_i x_j \sum_{a,b \in N_i \cup N_j} f_{ab} d_{\pi_{ij}(a)\pi_{ij}(b)} \\
+ \quad & \overline{x_i}\,\overline{x_j} \sum_{a,b \in N_i \cup N_j} f_{ab} d_{\pi(a)\pi(b)} \Bigg) \\
+ \quad & \sum_{C_i \in \mathcal{C}, n_\ell \notin \cup_j C_j} \Bigg(x_i \sum_{a,b \in N_i \cup \{n_\ell\}} f_{ab} d_{\pi_i(a)\pi_i(b)} \\
+ \quad & \overline{x_i} \sum_{a,b \in N_i \cup \{n_\ell\}} f_{ab} d_{\pi(a)\pi(b)} \Bigg).
\end{aligned}
$$

Here, π is the initial solution (permutation), π_i is the permutation that apply pairwise exchange on C_i only, π_j is the permutation that apply pairwise exchange on C_j only, and π_{ij} is the permutation that apply pairwise exchange on both C_i and C_j.

The special case when $k = 2$ we present here restrict each local change as pairwise exchange, this can be generalized further, we can define various kinds of local changes, and interpret the binary variables as decision variables: apply this local change or not. With this modeling of local search, the 2-Way 1-Hot encoding will always be automatically satisfied, meaning no penalizing parameters to scale in the QUBO. Moreover, we use less variables in the formulation.

Another important factor of the algorithm is the order in which the neighborhood is scanned, or in other words, the rules of selecting the subsets. This order can be deterministic or chosen at random. In our experiments, we explore a greedy selection rule. Namely, we scan the pair exchange neighborhood, that is, all permutations which can be obtained from the given one by applying a transposition to it. We then rank the pairs by the improvement of the solution, and finally select the subsets based on this ranking.

3.5 General Algorithm In this sub-section, we summarize the general algorithm for modeling local search with special-purpose hardware while comparing our approach to previous methods.

For a general combinatorial optimization problem, QLS algorithm in [56, 57, 58] can be summarized as follows:

Algorithm 3.1.

```
1: Starting from a feasible solution of the original problem of size $n$
2: for $i = 1, 2, \cdots$ do
3:     Choose a subset of the $n$ variables
4:     Formulate the sub-QUBO and use a QUBO solver to find a new feasible solution
5:     if new solution is better then
6:         accept the new solution
7:     end if
8: end for
```

We extend the above algorithm in such a way that instead of only choosing one subset from the n possible variables, we choose m pairwise disjoint subsets from the n variables. Within each subset, we search through all possible feasible configurations by formulating it into a sub-QUBO. The subsets need not to be equally sized, but for the sake of simplicity, we assume that all subsets are of size k. We note that the unlike similar methods for local search in parallel, the m subsets do not need to be mutually independent, which makes this a powerful approach. The general algorithm can be summarized as follows:

Algorithm 3.2.

```
1: Starting from a feasible solution of the original problem of size $n$
2: for $i = 1, 2, \cdots$ do
3:     Choose $m$ pairwise disjoint subsets of size $k$ from the $n$ variables
4:     Formulate the sub-QUBO and use QUBO solver to find a new feasible solution
5:     if new solution is better then
6:         accept the new solution
7:     end if
8: end for
```

Algorithm 3.2 for $k > 2$ creates an optimization problem that has constraints, therefore there is the need to introduce penalty terms to the QUBO. The case when $k = 2$ can be reduced to a sub-problem that does not have any constraints therefore we give a different algorithm for this case as it does not require any penalty terms. In summary, for this case, a binary variable represents the decision of whether or not a specified local change should be applied.

The algorithm can be described as follows:

Algorithm 3.3.

```
1: Starting from a feasible solution of the original problem of size $n$
2: for $i = 1, 2, \cdots$ do
3:     Choose $m$ pairwise disjoint subsets from the $n$ variables
4:     Define a possible local change and assign a binary variable for each subset to represent whether
       or not to apply that local change
5:     Formulate the QUBO and use QUBO solver to find a solution
6:     if new solution is better then
7:         Accept the new solution
8:     end if
9: end for
```

Further comparison of the approaches can be found in the supplementary material. With our approach, for problems with 2-Way 1-Hot constraints like TSP and QAP, in each iteration, we solve a QUBO to explore the search space of size $(k!)^m$. Moreover, in the case when $k = 2$, since there are no constraints, all solutions searched by the QUBO solver are feasible solutions, while for other values of k, a large number of the solutions searched are infeasible since they will violate the permutation constraints.

Finally, in general, parallel local search requires that if more than one local search is performed at the same iteration, these sub-problems must be mutually independent, whereas our approach does not have this requirement making it a more powerful approach.

4 Experiments on QAP

We implemented the algorithm using Python 2.7. As a QUBO solver, we compared the modeling approaches using the Fujitsu Digital Annealing Unit (DA) with the parallel tempering mode. We test our algorithm on problem instances from the QAP benchmark library, `QAPLIB` [8]. There are mainly two parameters in DA we tuned in our experiments. The first is the number of iterations. This refers to the number of MC steps in the DA algorithm [1]. As pointed out in [1], each MC step takes the same amount of time, therefore can be considered as a time limit on DA. The second is the choice of initial state for the DA. We carried out experiments in both setting, (i) initialing from a random state and (ii) initializing from a given state.

The main goal of the experiments is to *compare the different modeling strategies* for local search on special-purpose hardware of fixed size. The modeling techniques presented in this work can be summarized by the parameters k and m, therefore, in our experiments we vary these values and demonstrate the advantages of one over the other. (Additional plots can be found in the supplemental material.)

The parameters k and m: We first compare the performance of the algorithm with different values of k and m. We have tested the algorithm on multiple problem instances from `QAPLIB`, and they all have similar results. Figure 5 gives the value of the objective function with respect to the number of iterations of the algorithm. We start with a randomly generated initial solution, and apply greedy selection rule to select the refinement clusters. We can see that with $k = 2$, we can reach close to optimal within 5 iterations, and the quality of the solution is the best among all choices of k and m.

Since at each iteration, we are solving a QUBO in the DA, the configuration in DA will also effect quality of the solution. Figure 5 shows the result when we did not give any starting initial solution in the annealing process, and Figure 6 shows the result when we assign an initial solution to DA. The advantage of assigning initial solution is that we will never get a solution that is worse than the initial feasible solution, and the drawback is, we will be more likely to stay in a local optimal. From the plots, we can see that $k = 2$ in both cases achieve the best result among all choices of k and m.

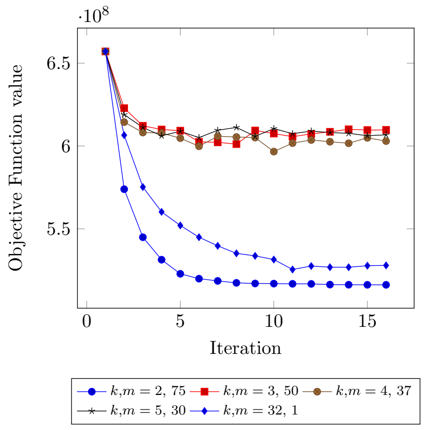

Figure 5: Comparison of different k and m, greedy selection, random start in DA. Dataset: ‘tai150b’

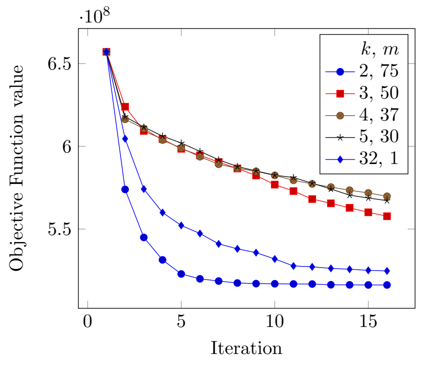

Figure 6: Comparison of different k and m, greedy selection, assign initial solution to DA. Dataset: ‘tai150b’

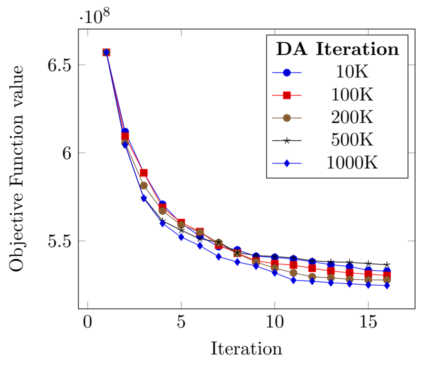

Figure 7: Dependence of objective value on the number of iterations using the greedy selection, assigning initial solution to DA with each refinement subset size is $k = 32$. Dataset: ’tai150b’

Time limit per iteration Next, we compare the performance of the modeling strategies with different time limit for solving the QUBO. We give the DA different number of iterations to control the annealing time. We found that when $k = 2$, with a small number of iterations (10,000), we can find a solution that has good quality, as shown in Figure 8. While for larger k, in most cases, as what shows in Figure 7, we need more iterations in DA to guarantee the quality of the solution.

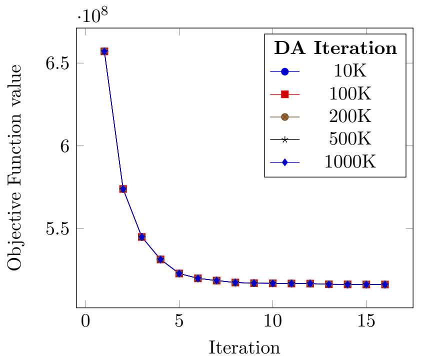

Figure 8: Dependence of objective value on the number of iterations using the greedy selection, assigning initial solution to DA with each refinement subset size is $k = 2$. Dataset: 'tai150b'

5 Conclusions and Discussions

Post Moore systems such as special-purpose hardware are being developed for different scientific domains. In combinatorial optimization, different novel hardware is emerging that are unified with respect to the Ising model, that is, they are designed to solve problems formulated in the Ising model or as a QUBO. As the hardware emerges, there is a challenge for existing well-established algorithms to take advantage of these hardware especially for large-scale problems. We have tackled this challenge by proposing different models, modeling techniques and algorithms that utilize the hardware efficiently. In particular, for large problems, we have demonstrated how to model multiple sub-problems that are not necessarily mutually independent as a step in a local search framework. Given that the QUBO is unconstrained by definition, we have further showed how to model sub-problems as a QUBO that implicitly satisfy the constraints, thus searching an exponentially larger search space per iteration compared to previous methods and also utilizing the given hardware more efficiently. This then provides new possibilities to escape from local optima. Our novel modeling techniques and algorithms can easily be adopted to a large class of local search algorithms thus demonstrating one step in which well-established algorithms can utilize this emerging technologies.

References

[1] Maliheh Aramon, Gili Rosenberg, Elisabetta Valiante, Toshiyuki Miyazawa, Hirotaka Tamura, and Helmut Katzgrabeer. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Frontiers in Physics*, 7:48, 2019.

[2] Frederick Bock. An algorithm for solving travelling-salesman and related network optimization problems. In *Operations Research*, volume 6, pages 897–897. INST OPERATIONS RESEARCH MANAGEMENT SCIENCES 901 ELKRIDGE LANDING RD, STE , 1958.

[3] Erik G Boman, Ümit V Çatalyürek, Cédric Chevalier, and Karen D Devine. The zoltan and isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering and coloring. *Scientific Programming*, 20(2):129–150, 2012.

[4] Michael Booth, Steven P Reinhardt, and Aidan Roy. Partitioning optimization problems for hybrid classical. *quantum execution. Technical Report*, pages 01–09, 2017.

[5] Ajinkya Borle and Samuel J Lomonaco. Analyzing the quantum annealing approach for solving linear least squares problems. In *International Workshop on Algorithms and Computation*, pages 289–301. Springer, 2019.

[6] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. In *Algorithm Engineering*, pages 117–158. Springer, 2016.

[7] Rainer E Burkard, Eranda Cela, Panos M Pardalos, and Leonidas S Pitsoulis. The quadratic assignment problem. In *Handbook of combinatorial optimization*, pages 1713–1809. Springer, 1998.

[8] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. Qaplib–a quadratic assignment problem library. *Journal of Global optimization*, 10(4):391–403, 1997.

[9] Chia Cheng Chang, Arjun Gambhir, Travis S Humble, and Shigetoshi Sota. Quantum annealing for systems of polynomial equations. *Scientific reports*, 9, 2019.

[10] Tyler H Chang, Thomas CH Lux, and Sai Sindhura Tipirneni. Least-squares solutions to polynomial systems of equations with quantum annealing. *Quantum Information Processing*, 18(12):374, 2019.

[11] Guillaume Chapuis, Hristo Djidjev, Georg Hahn, and Guillaume Rizk. Finding maximum cliques on a quantum annealer. In *Proceedings of the Computing Frontiers Conference*, pages 63–70. ACM, 2017.

[12] Carleton Coffrin, Harsha Nagarajan, and Russell Bent. Evaluating ising processing units with integer programming. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 163–181. Springer, 2019.

[13] Daniel Crawford, Anna Levit, Navid Ghadermarzy, Jaspreet S Oberoi, and Pooya Ronagh. Reinforcement learning using quantum boltzmann machines. *arXiv preprint arXiv:1612.05695*, 2016.

[14] Georges A Croes. A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812, 1958.

[15] Sanjeeb Dash. A note on qubo instances defined on chimera graphs. *arXiv preprint arXiv:1306.1202*, 2013.

[16] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[17] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*, pages 175–181. IEEE, 1982.

[18] Robert C Foster, Brian Weaver, and James Gattiker. Applications of quantum annealing in statistics. *arXiv preprint arXiv:1904.06819*, 2019.

[19] Fred Glover and Gary Kochenberger. A tutorial on formulating qubo models. *arXiv preprint arXiv:1811.11538*, 2018.

[20] Firas Hamze and Nando de Freitas. From fields to trees. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 243–250. AUAI Press, 2004.

[21] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[22] Maxwell Henderson, John Novak, and Tristan Cook. Leveraging adiabatic quantum computation for election forecasting. *arXiv preprint arXiv:1802.00069*, 2018.

[23] Maritza Hernandez and Maliheh Aramon. Enhancing quantum annealing performance for the molecular similarity problem. *Quantum Information Processing*, 16(5):133, 2017.

[24] Maritza Hernandez, Arman Zaribafiyan, Maliheh Aramon, and Mohammad Naghibi. A novel graph-based approach for determining molecular similarity. *arXiv preprint arXiv:1601.06693*, 2016.

[25] Takahiro Inagaki, Yoshitaka Haribara, Koji Igarashi, Tomohiro Sonobe, Shuhei Tamate, Toshimori Honjo, Alireza Marandi, Peter L McMahon, Takeshi Umeki, Koji Enbutsu, et al. A coherent ising machine for 2000-node optimization problems. *Science*, 354(6312):603–606, 2016.

[26] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, R Harris, Andrew J Berkley, Jan Johansson, Paul Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346):194, 2011.

[27] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

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

[29] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

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

[31] Amir Khoshaman, Walter Vinci, Brandon Denis, Evgeny Andriyash, and Mohammad H Amin. Quantum variational autoencoder. *Quantum Science and Technology*, 4(1):014001, 2018.

[32] David Kielpinski, Ranojoy Bose, Jason Pelc, Thomas Van Vaerenbergh, Gabriel Mendoza, Nikolas Tezak, and Raymond G Beausoleil. Information processing with large-scale optical integrated circuits. In *2016*

IEEE International Conference on Rebooting Computing (ICRC), pages 1–4. IEEE, 2016.

[33] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[34] Gary A Kochenberger and Fred Glover. A unified framework for modeling and solving combinatorial optimization problems: A tutorial. In *Multiscale Optimization Methods and Applications*, pages 101–124. Springer, 2006.

[35] Anna Levit, Daniel Crawford, Navid Ghadermarzy, Jaspreet S Oberoi, Ehsan Zahedinejad, and Pooya Ronagh. Free energy-based reinforcement learning using a quantum processor. *arXiv preprint arXiv:1706.00074*, 2017.

[36] Shen Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

[37] Catherine C McGeoch and Cong Wang. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Frontiers*, page 23. ACM, 2013.

[38] Peter L McMahon, Alireza Marandi, Yoshitaka Haribara, Ryan Hamerly, Carsten Langrock, Shuhei Tamate, Takahiro Inagaki, Hiroki Takesue, Shoko Utsunomiya, Kazuyuki Aihara, et al. A fully programmable 100-spin coherent ising machine with all-to-all connections. *Science*, 354(6312):614–617, 2016.

[39] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[40] Christian FA Negre, Hayato Ushijima-Mwesigwa, and Susan M Mniszewski. Detecting multiple communities using quantum annealing on the d-wave system. *arXiv preprint arXiv:1901.09756*, 2019.

[41] Florian Neukart, Gabriele Compostella, Christian Seidel, David Von Dollen, Sheir Yarkoni, and Bob Parney. Traffic flow optimization using a quantum annealer. *Frontiers in ICT*, 4:29, 2017.

[42] M Peter Nightingale and Cyrus J Umrigar. *Quantum Monte Carlo methods in physics and chemistry.* Number 525. Springer Science & Business Media, 1998.

[43] Nicolas Melo De Oliveira, Ricardo Martins De Abreu Silva, and Wilson Rosa De Oliveira. Qubo formulation for the contact map overlap problem. *International Journal of Quantum Information*, 16(08):1840007, 2018.

[44] Daniel OMalley, Velimir V Vesselinov, Boian S Alexandrov, and Ludmil B Alexandrov. Nonnegative/binary matrix factorization with a d-wave quantum annealer. *PloS one*, 13(12):e0206653, 2018.

[45] Christos Papalitsas, Theodore Andronikos, Konstantinos Giannakis, Georgia Theocharopoulou, and Sofia Fanarioti. A qubo model for the traveling salesman problem with time windows. *Algorithms*, 12(11):224, 2019.

[46] JF Puget. D-wave vs cplex comparison. part 2: Qubo, 2013, 2018.

[47] Michael L Rogers and Robert L Singleton Jr. Floating-point calculations on a quantum annealer: Division and matrix inversion. *arXiv preprint arXiv:1901.06526*, 2019.

[48] Gili Rosenberg, Poya Haghnegahdar, Phil Goddard, Peter Carr, Kesheng Wu, and Marcos López De Prado. Solving the optimal trading trajectory problem using a quantum annealer. *IEEE Journal of Selected Topics in Signal Processing*, 10(6):1053–1060, 2016.

[49] Ilya Safro, Dorit Ron, and Achi Brandt. A multilevel algorithm for the minimum 2-sum problem. *J. Graph Algorithms Appl.*, 10(2):237–258, 2006.

[50] Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics (JEA)*, 19:2–2, 2015.

[51] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In *European Symposium on Algorithms*, pages 469–480. Springer, 2011.

[52] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[53] Elad Schneidman, Michael Berry II, Ronen Segev, and William Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440:1007–12, 05 2006.

[54] Alex Selby. Efficient subgraph-based sampling of ising-type models with frustration. *arXiv preprint arXiv:1409.3934*, 2014.

[55] Ruslan Shaydulin, Jie Chen, and Ilya Safro. Relaxation-based coarsening for multilevel hypergraph partitioning. *Multiscale Modeling & Simulation*, 17(1):482–506, 2019.

[56] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Christian FA Negre, Ilya Safro, Susan M Mniszewski, and

Yuri Alexeev. A hybrid approach for solving optimization problems on small quantum computers. *Computer*, 52(6):18–26, 2019.

[57] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. Community detection across emerging quantum architectures. *arXiv preprint arXiv:1810.07765*, 2018.

[58] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. Network community detection on small quantum computers. *Advanced Quantum Technologies*, page 1900029, 2019.

[59] Robert H Swendsen and Jian-Sheng Wang. Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57(21):2607, 1986.

[60] Tony T Tran, Minh Do, Eleanor G Rieffel, Jeremy Frank, Zhihui Wang, Bryan O'Gorman, Davide Venturelli, and J Christopher Beck. A hybrid quantum-classical approach to solving scheduling problems. In *Ninth annual symposium on combinatorial search*, 2016.

[61] Hayato Ushijima-Mwesigwa, Christian FA Negre, and Susan M Mniszewski. Graph partitioning using quantum annealing on the d-wave system. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, pages 22–29. ACM, 2017.

[62] Hayato Ushijima-Mwesigwa, Ruslan Shaydulin, Christian FA Negre, Susan M Mniszewski, Yuri Alexeev, and Ilya Safro. Multilevel combinatorial optimization across quantum architectures. *arXiv preprint arXiv:1910.09985*, 2019.

[63] Davide Venturelli, D Marchand, and Galo Rojo. Job shop scheduling solver based on quantum annealing. In *Proc. of ICAPS-16 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling (COPLAS)*, pages 25–34, 2016.

[64] Masanao Yamaoka, Chihiro Yoshimura, Masato Hayashi, Takuya Okuyama, Hidetaka Aoki, and Hiroyuki Mizuno. 24.3 20k-spin ising chip for combinational optimization problem with cmos annealing. In *2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, pages 1–3. IEEE, 2015.

[65] Chihiro Yoshimura, Masanao Yamaoka, Hidetaka Aoki, and Hiroyuki Mizuno. Spatial computing architecture using randomness of memory cell stability under voltage control. In *2013 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4. IEEE, 2013.