Techniques to Enhance a QUBO Solver For Permutation-Based Combinatorial Optimization

Siong Thye Goh
stgoh@smu.edu.sg
Singapore Management University
Singapore

Jianyuan Bo
jybo.2020@phdcs.smu.edu.sg
Singapore Management University
Singapore

Sabrish Gopalakrishnan
sabrishg@smu.edu.sg
Singapore Management University
Singapore

Hoong Chuin Lau*
hclau@smu.edu.sg
Singapore Management University
Singapore

ABSTRACT

Many combinatorial optimization problems can be formulated as a problem to determine the order of sequence or to find a corresponding mapping of the objects. We call such problems permutation-based optimization problems. Many such problems can be formulated as a quadratic unconstrained binary optimization (QUBO) or Ising model by introducing a penalty coefficient to the permutation constraint terms. While classical and quantum annealing approaches have been proposed to solve QUBOs to date, they face issues with optimality and feasibility. Here we treat a given QUBO solver as a black box and propose techniques to enhance its performance. First, to ensure an effective search for good quality solutions, a smooth energy landscape is needed; we propose a data scaling approach that reduces the amplitudes of the input without compromising optimality. Second, we need to tune the penalty coefficient. In this paper, we illustrate that for certain problems, it suffices to tune the parameter by performing random sampling on the penalty coefficients to achieve good performance. Finally, to handle possible infeasibility of the solution, we introduce a polynomial-time projection algorithm. We apply these techniques along with a divide-and-conquer strategy to solve some large-scale permutation-based problems and present results for TSP and QAP.

CCS CONCEPTS

• **Mathematics of computing** → **Solvers**; • **Applied computing** → **Operations research**.

KEYWORDS

QUBO, combinatorial optimization, traveling salesman problem, QAP, scheduling, routing, permutation constraint

*Corresponding author

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07…$15.00
https://doi.org/10.1145/3520304.3533982

ACM Reference Format:
Siong Thye Goh, Jianyuan Bo, Sabrish Gopalakrishnan, and Hoong Chuin Lau. 2022. Techniques to Enhance a QUBO Solver For Permutation-Based Combinatorial Optimization. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3520304.3533982

1 INTRODUCTION

Combinatorial optimization problems such as the Traveling Salesman Problem (TSP), the Flow Shop Scheduling Problem (FSP), and the Quadratic Assignment Problem (QAP) are NP-hard problems underlying many route planning and machine scheduling problems [28]. Traditionally these problems are modeled as integer programs and solved with mathematical programming solvers such as CPLEX[8] and Gurobi[11]. These solvers are based on the branch and bound paradigm which are exponential time algorithms in the worst case.

Recently, motivated by the promise of quantum computing, an alternative approach is gaining traction. The idea is to first formulate the problems as Ising models (or equivalently, quadratic unconstrained binary optimization problems (QUBO) [18], [5]), which are amenable to quantum annealing-based solutions. A QUBO problem can be described as

$$\min_{x \in \{0,1\}^n} x^T Q x.$$

It is an NP-hard problem. Special quantum hardware such as D-Wave's quantum annealer has been developed that demonstrates effectiveness in solving targeted problems like Max-SAT and Max Cut (see e.g. [31]). Nurse-scheduling problem and flight gate assignment problem have also been solved in [14] and [24] respectively. Some other interesting problems have been formulated as a QUBO as well. In [22], the problem of estimating the density of states of Boolean satisfiability problem is formulated as a QUBO.

On a related front, technology companies have developed fast QUBO solvers (QS) such as Alpha-QUBO [10] (a software solver) and Fujitsu's Digital Annealer (DA) [3], a CMOS machine with specialized hardware architecture. These QUBO solvers face several challenges including hardware limitations and no guarantee of solution quality.

Currently, these solvers assume that the given problem is unconstrained, although it has been proven in [18] that a constrained problem such as those stated in a typical integer programming

(IP) can be converted to an unconstrained problem using a penalty method[23]. The penalty method achieves this by ensuring that an infeasible solution is not the optimal solution. More details on how to convert a quadratic binary optimization problem with permutation constraints to a QUBO will be covered in the background section.

In this paper, we focus on permutation-based combinatorial optimization problems, i.e. problems whose constraints are permutation constraints. For a routing problem (such as TSP), this corresponds to deciding the order of nodes to visit; for a scheduling problem (such as FSP), this corresponds to deciding the order of tasks to be served by a machine. It is known that these problems have dedicated or handcrafted heuristics that perform well: for example for TSP, Concorde [2] is known to be an effective dedicated solver for TSP. For FSP, there are standard heuristics such as the NEH algorithm, such as [19]; while for QAP, various handcrafted metaheuristics have been proposed (see for example [17]). Heuristics and dedicated solvers may not perform well in specific problem instances, e.g. some hard TSP instances were listed [13], and there is no one-size-fits-all heuristic for QAP benchmark instances.

There are a few challenges that we have to address to use the solvers effectively. These challenges are

- The data describing the QUBO varies greatly in magnitude and a QUBO solver focuses on only certain bits as a result.
- Choosing the right penalty coefficient so that we obtain high-quality solutions.
- Solution returned by QUBO solver is not feasible for the original problem, and
- Problem size constraints

The research question that we address in this paper is what are the possible techniques to overcome these challenges. We address these challenges for permutation-based problems as follows:

We introduce a data scaling technique to control the magnitude of the data and yet ensuring that the problem is equivalent to the original problem in Section 3.1. In the penalty method, we have a penalty parameter A in Equation 3 where it plays an important role in the solution quality. While theoretically, a large A ensures that the resulting QUBO is equivalent to the original problem; in practice, it is observed that there is a 'Goldilocks region' that contains penalty values that work well as discussed in [10], if it is too large, it leads to sub-optimality while if it is too small, it leads to infeasibility. Hence, we investigate penalty coefficient tuning to obtain good performance in Section 3.2. Furthermore, the solution that we obtained from a QUBO solver need not be feasible for the original problem. We discuss how to compute projection to ensure that the solution that we obtain is feasible in 3.3. In Figure 1, we present a framework to solve large scale permutation-based optimization problem.

More precisely, we show in this paper that by incorporating our techniques, we can enhance the performance of a black-box heuristic QUBO solver that is not guaranteed to solve QUBO instances optimally. An example is the DA which is based on simulated annealing with a simple bit-flip neighborhood. In order to tackle large-scale problems that is beyond what the QUBO solver is capable of solving (due to hardware limitation), a divide-and-conquer approach can be used to decompose the problem into sub-QUBOs. Note that our proposed techniques are hardware-agnostic, and as the quantum hardware matures, QUBO solvers enhanced with our proposed techniques can provide a strong competitor to commercial exact solvers like CPLEX and GUROBI.

Our key contributions are as follows:

- To enhance solution quality, we propose a data-scaling method to convert an instance to one with smaller cost variations while preserving the ranking of solutions for the original problem instance for QUBO with permutation constraint, and prove that our data scaling method applies to all permutation-based problems.
- To ensure feasibility, we propose a method to project infeasible solutions obtained by the QUBO solver to feasible solutions using a polynomial-time weighted assignment algorithm.
- To balance between quality and feasibility, we study the effects of the penalty parameters in the QUBO formulation and compare different parameter tuning approaches experimentally.
- We apply our techniques to the DA QUBO solver to solve large Euclidean TSP (E-TSP) and Quadratic Assignment Problem (QAP) instances via a divide-and-conquer process. We respectively evaluate our approach by comparing the Concorde solver (for E-TSP) and the qbsolv framework running the same QUBO solver (for QAP).

2 PERMUTATION-BASED PROBLEMS

A permutation-based combinatorial optimization problem involves permuting n objects to minimize a certain objective function. Common examples of permutation-based problems include TSP, FSP and QAP. Such problems can be modeled as minimizing a quadratic objective function of the following form:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{u=1}^{n}\sum_{v=1}^{n} x_{u,i}Q_{u,i,v,j}x_{v,j} \tag{1}$$

subject to $\sum_{u=1}^{n} x_{u,i} = 1, \forall i \in \{1,\dots,n\}$ and $\sum_{i=1}^{n} x_{u,i} = 1, \forall u \in \{1,\dots,n\}$ where $x_{u,i}$ takes value 1 if object u is assigned to slot i and it takes value 0 otherwise. The two constraints ensure that each object is given a slot and vice versa. We call the first group of constraint the column sum constraints and the second group of constraint the row sum constraints.

We can convert this formulation to an unconstrained QUBO model by squaring the constraint violations and adding them to the original objective function:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{u=1}^{n}\sum_{v=1}^{n} x_{u,i}Q_{u,i,v,j}x_{v,j} \tag{2}$$

$$+ A\left[\sum_{u=1}^{n}\left(\sum_{i=1}^{n} x_{u,i} - 1\right)^2 + \sum_{i=1}^{n}\left(\sum_{u=1}^{n} x_{u,i} - 1\right)^2\right] \tag{3}$$

where A is a single parameter that we have to tune to penalize constraint violation. Henceforth, we call a QUBO expressed in that form a "permutation QUBO".

For this work, we choose to tune a single parameter A following the formulation given in [18], although one could have assigned

a new parameter for each of the $2n$ constraints. By doing so, the number of parameters to tune does not grow as n increases. One can observe that the symmetry in the permutation constraints suggests that we do not have to emphasize some constraints over the others intuitively.

3 TECHNIQUES TO IMPROVE SOLUTION QUALITY

In this section, we introduce our techniques to perform data scaling, investigate the effect of parameter tuning, and propose a projection scheme to improve the quality of solution when we use a QUBO annealing solver.

3.1 Data Scaling

In [12], it has been shown that for the TSP problem, one can reduce all the distances to or from a particular city by a constant and preserve the ranking of the solutions. This result can be extended to the quadratic binary optimization problem with permutation constraints as follows.

Given the optimization problem (1), pick any $\hat{j} \in \{1, \ldots, n\}$ and define $\tilde{Q}$ as follows:

$$\forall u, i, v \in \{1, \ldots, n\}, \tilde{Q}_{u,i,v,j} = \begin{cases} Q_{u,i,v,j} & \text{if } j \neq \hat{j} \\ Q_{u,i,v,j} + \Delta & \text{if } j = \hat{j} \end{cases}$$

That is, we change those matrix entries with index $j = \hat{j}$ by a constant Δ. We call this process data scaling.

Consider the resulting scaled optimization problem:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{u=1}^{n} \sum_{v=1}^{n} x_{u,i} \tilde{Q}_{u,i,j,v} x_{j,v} \tag{4}$$

subject to $\sum_{u \in I} x_{u,i} = 1$ and $\sum_{i \in J} x_{u,i} = 1$.

The following lemma shows that the ranking of optimality is preserved under data scaling:

Lemma 1. *A feasible solution y that is better than feasible z for optimization problem* (1) *remains better for optimization problem* (4)*. That is if y and z are feasible solution optimization problem* (1)*, then $y^T Q y \leq z^T Q z \implies y^T \tilde{Q} y \leq z^T \tilde{Q} z$.*

Note that our theoretical result can be further generalized to constraint of the form of $\sum_{i \in I} x_{i,j} = B$ and $\sum_{i \in I} x_{i,j} = C$ where B and C satisfy $B|J| = C|I|$. We state and prove the more general version of the result here:

Consider the following quadratic programming problem with binary variables.

$$\min \sum_{\substack{i,u \in I \\ v,j \in J}} x_{i,j} Q_{i,j,u,v} x_{u,v} \tag{5}$$

subject to $\sum_{i \in I} x_{i,j} = B$ and $\sum_{j \in J} x_{i,j} = C$ where $|J|B = |I|C$. That is we impose the conditions that the row sum is a constant and the column sum is another constant and the problem is feasible.

Let $\hat{j} \in J$. Consider the optimization problem with perturbed data:

$$\min \sum_{\substack{i,u \in I \\ v,j \in J}} x_{i,j} \tilde{Q}_{i,j,u,v} x_{u,v} \tag{6}$$

subject to $\sum_{i \in I} x_{i,j} = B$ and $\sum_{j \in J} x_{i,j} = C$.

where

$$\tilde{Q}_{i,j,u,v} = \begin{cases} Q_{i,j,u,v} & \text{if } j \neq \hat{j} \\ Q_{i,j,u,v} - \Delta & if\, j = \hat{j} \end{cases}$$

The following lemma shows that the ranking of optimality is preserved under data scaling:

Lemma 2. *A feasible solution y that is better than feasible solution z for optimization problem* (5) *remains better for optimization problem* (6)*. That is if y and z are feasible solution optimization problem* (5)*, then $y^T Q y \leq z^T Q z \implies y^T \tilde{Q} y \leq z^T \tilde{Q} z$.*

Proof. It suffices to show that the difference in the objective values of the two feasible solutions y and z remain the same before and after scaling. More precisely,

$$\begin{aligned} &\textstyle\sum_{\substack{i,u \in I \\ v,j \in J}} z_{i,j} Q_{i,j,u,v} z_{u,v} - \sum_{\substack{i,u \in I \\ v,j \in J}} y_{i,j} Q_{i,j,u,v} y_{u,v} \\ &= \textstyle\sum_{\substack{i,u \in I \\ v,j \in J}} z_{i,j} \tilde{Q}_{i,j,u,v} z_{u,v} - \sum_{\substack{i,u \in I \\ v,j \in J}} y_{i,j} \tilde{Q}_{i,j,u,v} y_{u,v}. \end{aligned}$$

By definition we have,

$$\begin{aligned} &\sum_{\substack{i,u \in I \\ j,v \in J}} z_{i,j} \tilde{Q}_{i,j,u,v} z_{u,v} - y_{i,j} \tilde{Q}_{i,j,u,v} y_{u,v} \\ &= \sum_{\substack{i,u \in I \\ j,v \in J}} [z_{i,j} Q_{i,j,u,v} z_{u,v} - y_{i,j} Q_{i,j,u,v} y_{u,v}] \\ &\quad - \Delta \sum_{i,u \in I, v \in J} [z_{i,\hat{j}} z_{u,v} - y_{i,\hat{j}} y_{u,v}] \end{aligned} \tag{7}$$

Since both solutions y and z are known to be to be feasible by our assumption, i.e. $\sum_{i \in I} y_{i,j} = \sum_{i \in I} z_{i,j} = B$ and $\sum_{j \in J} y_{i,j} = \sum_{j \in J} z_{i,j} = C$,

$$\sum_{\substack{i,u \in I \\ v \in J}} y_{i,\hat{j}} y_{u,v} = \sum_{i \in I} y_{i,\hat{j}} \sum_{u \in I} \sum_{v \in J} y_{u,v} = B \sum_{u \in I} C = BC|I|$$

and similarly $\sum_{\substack{i,u \in I \\ v \in J}} z_{i,\hat{j}} z_{u,v} = BC|I|$

Hence the term (7) is equal to 0.

□

Specifically when $B = C = 1$, and the index set $I = J = \{1, \ldots, n\}$, we have the special case for permutation based optimization problems which is Lemma 1.

Even though we have shown that we can change the objective value corresponding to certain indices by a constant and yet the two problems remain equivalent, it is interesting to determine the value to be scaled for a particular index i, i.e. Δ_i. One approach is to use the result from [33] to minimize the variance of all the scaled distances. Intuitively, this corresponds to smoothing the landscape of the objective function.

3.2 Penalty Parameter Tuning

As discussed above, tuning the penalty parameter A in Equation 3 is an important step in ensuring solution feasibility and quality for the original problem.

There are multiple approaches for hyper-parameter tuning. Note that unlike machine learning tasks where the hyper-parameters are tuned offline with large training sets, in solving combinatorial optimization problems using a QS, each call to the QS takes substantial time, and hence we need to perform online tuning with as few calls to the QS as possible. In this paper, we investigate the following approaches:

Online parameter search: One approach is to perform an online search for suitable parameters. Bayesian approaches are model-based approaches that updates the search regions dynamically. Hyperopt [4] and Optuna[1] are frequently used to improve the quality of the solution iteratively. Model-free approaches based on Particle Swarm Optimization (PSO) [21] are also able to achieve reasonable results. Furthermore, while there are multiple hyper-parameters that can be set for PSO, interestingly we notice that the performance is not sensitive to them for our problems.

Statistical sampling: Another approach is to draw the parameter values based on some distributions. For example, we can collect data to learn the average value and standard deviation of penalty parameter of the problem instance that performs well and fit a distribution to it. During the prediction stage, we draw samples from the fitted distribution repeatedly.

The choice of which parameter tuning scheme to use depends largely on the applications, and in the experiments that follow, we compare the results of these methods.

3.3 Projection to the Feasible Space

While ideally, QS should return feasible (though not optimal) solutions, this need not always be the case. In this paper, we propose a projection algorithm to map an infeasible solution to a feasible solution.

Suppose $z \in \{0,1\}^{n \times n}$ is an infeasible solution returned by QS. To restore feasibility of the original constrained problem, we solve the following optimization problem:

$$\min \sum_{i,j=1}^{n} (x_{i,j} - z_{i,j})^2 = \min \sum_{i,j=1}^{n} ((1 - 2z_{i,j})x_{i,j} + z_{i,j})$$

subject to $\sum_{i=1}^{n} x_{i,j} = 1, \forall j \in \{1, \ldots, n\}$ and $\sum_{j=1}^{n} x_{i,j} = 1, \forall i \in \{1, \ldots, n\}$.

Note that here z would be a given constant and hence this reduces to the standard Weighted Assignment Problem which can be solved in $O(\left(\frac{n}{k}\right)^3)$ time with the Hungarian algorithm [15].

4 APPLICATION TO TSP AND QAP

QUBO solvers generally suffer limitation on the size of the model that can be solved directly, e.g. a limit of 2000-qubits for D-Wave, 3000 binary variables for Alpha-QUBO, and 8192 binary variables for the Generation 2 Digital Annealer (DA) (and 10^5 for the third generation DA). Even with prospective hardware enhancements, it is challenging to cope with large-scale combinatorial optimization problems as problem size increases. Hence, to date, it is commonly agreed that hybrid methods are needed. For instance, in [16], [29], [30], and [32], hybrid quantum-classical approaches to solving scheduling problems have been proposed.

Our proposed techniques can be embedded within a divide-and-conquer framework portrayed in Figure 1. The framework decomposes a large permutation-based problem into multiple smaller instances via a clustering step. The techniques (data scaling, parameter tuning, and projections) can then be used to ensure that each sub-problem can be solved to a high quality. Finally, to obtain a feasible solution to the original problem, we need to have a stitching step to combine the solutions into a solution for the original problem.

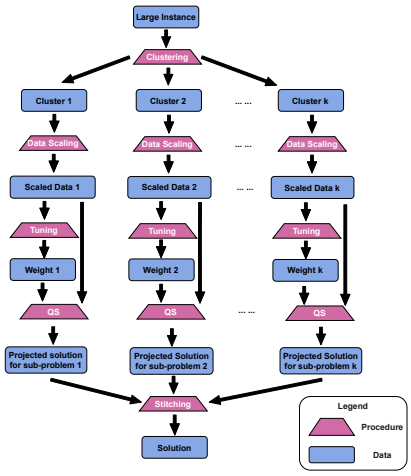


Figure 1: Divide and Conquer framework

When we decompose a large instance into smaller sub-instances, depending on the specific problem at hand, there are a few factors to take into consideration. Typically the size should be chosen such that the smaller problem performs well in terms of solution quality and solving time. Also, we do not want the sub-problem instances to be too small as that would affect the performance when we stitch the solutions to form a solution to the original problem. We let the number of sub-problems be denoted by k. Having solved the sub-problems, stitching involves combining the solutions. A nice property of permutation-based problems is that any permutation of $\{1, \ldots, n\}$ is a feasible solution. Hence finding good feasible solution hinges on finding an effective scheme to stitch the solutions of the clusters together. For example, for TSP, we can define the distance between two clusters to be the smallest distance of a city in the first cluster with a city in the second cluster. General decomposition of a permutation-based constraint optimization problem to a clustering is a challenging task that is not in the scope of the paper.

4.1 Euclidean Traveling Problem (E-TSP)

We focus on E-TSP (instead of the general TSP) in this paper since the constrained k-means algorithm [6] can be applied easily for clustering.

In [18], a QUBO formulation that only involves a quadratic number of terms in the number of cities is proposed. Without loss of generality, we can focus on the case where the graph is fully connected, as we can always introduce edges of infinite distances otherwise. We let d_{uv} be the distance between city u and city v. We require n^2 variables for an n-city instance. The first subscript of x represents the city and the second indicates the order that the city is going to be visited at. That is $x_{v,j}$ is the indicator variable that the city v is the j-th city to be visited. Notice that the constraint implies that this satisfies the permutation condition.

The formulation is as follows: $\min_x H_B(x) + AH_A(x)$, where

$$H_B(x) = \sum_{(u,v)\in E} d_{uv} \sum_{j=1}^{n} x_{u,j} x_{v,j+1}$$

describes the total distance travelled and

$$H_A = \sum_{v=1}^{n} \left(1 - \sum_{j=1}^{n} x_{v,j}\right)^2 + \sum_{j=1}^{n} \left(1 - \sum_{v=1}^{n} x_{v,j}\right)^2 \tag{8}$$

describes the constraints to be a feasible cycle.

For the clustering step, we use the constrained k-means clustering algorithm [6]. The benefit of this approach is that we have better control over the size of the clusters suitable for QS.

Let k be the number of clusters and largest cluster size be $|V_c|$. We define the least cost flip value Δ_{ij} between all cluster pairs to be the least cost of performing 2-opt to stitch the two clusters i and j together. The time complexity is $O(k^2|V_c|^2)$. To determine the ordering of stitching clusters, we solve a minimum cost Hamiltonian path problem by reusing our QUBO E-TSP model presented above (except it seeks a minimum Hamiltonian path [18] instead of cycle) with Δ_{ij} on the edges.

4.2 Quadratic Assignment Problem (QAP)

Given n facilities and n locations, the flow matrix (f_{ij}) denoting the flow quantity between facilities i and j, and a distance matrix d_{kl} which denotes the distance (weight) between locations k and l, the QAP is to find an assignment of facilities to locations that minimizes the weighted flow.

In [10], the indicator variable x_{ij} is used to denote that the facility i is assigned to location j. The total flow can be written as a QUBO, that is we want to minimize $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ij}d_{kl}x_{ik}x_{jl}$. Again, we need to impose the permutation constraints as we need to ensure that each location is assigned to exactly one facility and each facility is assigned to exactly one location.

If we have the coordinates of the location and facilities, we can perform some clustering that is based on the Euclidean metric. Unfortunately, there is a lack of clustering in the literature for QAP. We design a clustering scheme as such. For each facility, we associate it with the sum of the flow values that is associated with the facility, we then sort them in an increasing order. For each location, we associate it with the sum of the distances that is associated with the location, we then sort them in a decreasing order. We then match the facility that is associated with the high flow value with the location of low distance. We then compute the product value of the scores and perform a 1-D clustering on the scores. If the cluster is too large for the QS, we further divide them into smaller sub-problem.

Note that there is no stitching required for QAP as the assignments of facilities to the locations is what is required.

5 NUMERICAL EXPERIMENTS

In this section, we present our experimental results. Four sets of experiments are conducted:

- We investigate the effect of data scaling on TSP solution quality;
- We compare the relative performance on TSP solution quality under different parameter tuning approaches.
- We benchmark our approach to solve TSP. First, we compare the relative performance with direct QS call and with the popular TSP solver Concorde on TSPLIB [20] as well as hard Tnm instances. We also compare the relative performance when using an exact solver (CPLEX) directly instead of a heuristic QS.
- Finally, we compare the performance of our approach with qbsolv (running DA) on QAP instances. As baselines, we also compare against best published results on those instances.

We explain the experimental setup below.

The default QUBO solver is the Fujitsu Digital Annealer Generation 2 that runs in the Parallel Tempering mode.

For E-TSP clustering, we use the constrained k-means algorithm [6] to control the size of each sub-QUBO.

For the model-free PSO approach, the position of the particle represents the parameter to be tuned. The cost function of the PSO is set to be the objective function according to the application we are solving. Besides, we need to tune several hyperparameters for PSO including the inertia weight ω, two learning factors c_1 and c_2. ω controls the contribution rate of the particle's previous velocity to the particle's velocity at the current time step. A combination of c_1 and c_2 determines the convergence rate by balancing the global and local search capabilities. We update ω, c_1, and c_2 according to the linear time varying formulation in [25]. In order to have total 40 parameters evaluated, PSO runs 4 iterations (including the random initialization) with number of particles equals to 10. The search space is same as the Bayesian approaches.

The details of the parameter used are summarized in Table 1 where $D_{\max}$ refers to the maximum distance. Our code is available at https://github.com/BMDroid/Permute-QUBO-Tech.

5.1 Effect of Data Scaling on TSP

Data scaling reduces the variance of the input data describing the QUBOs. In Figure 2, we illustrate that data scaling improves the performance of TSP in that the solutions obtained will be strictly better than those without data scaling.

5.2 Effect of Parameter Tuning on TSP

We perform online parameter tuning on the TSP instances of size up to 800 obtained from TSPLIB and also on the TSP instances found in [13]. The x-axis on the both diagrams shows the size of the input.

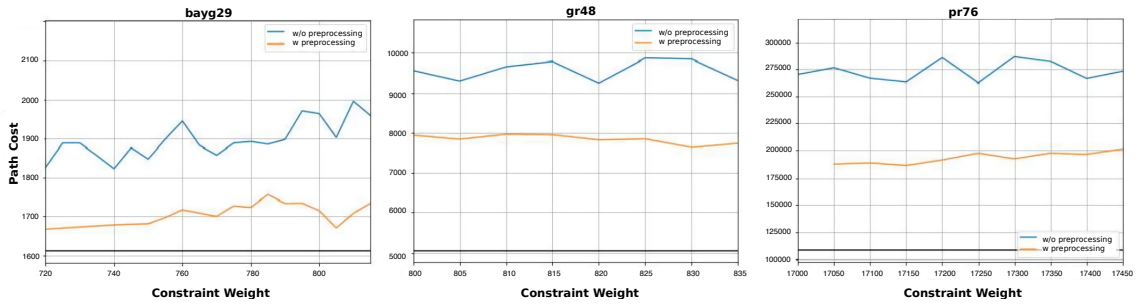
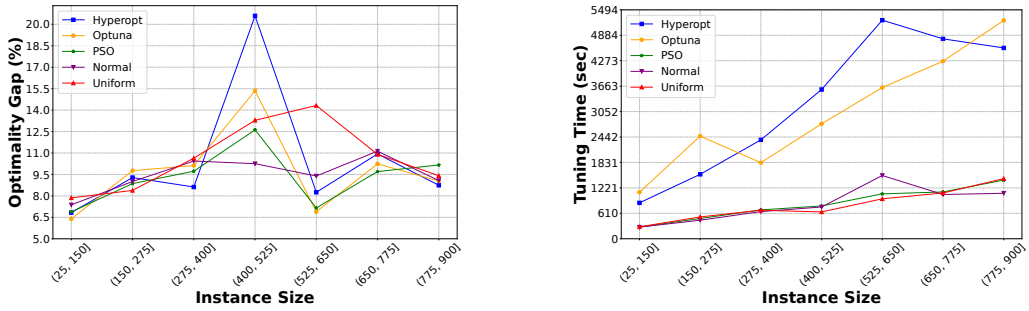
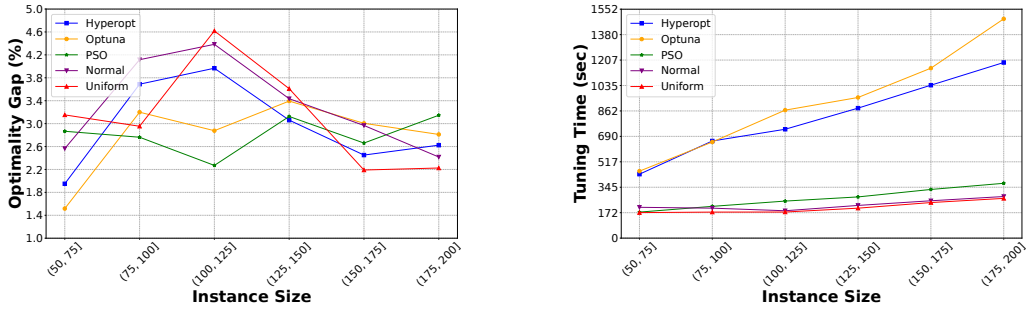


Figure 2: Lower objective value is obtained after performing data scaling



(a) Optimality Gap **(b) Tuning Time**

Figure 3: Optimality gap and tuning time with different tuning approaches for TSPLIB instances



(a) Optimality Gap **(b) Tuning Time**

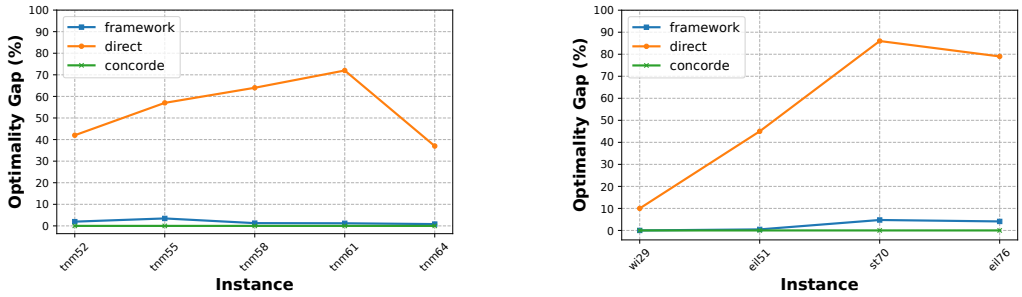
Figure 4: Optimality gap and tuning time of TSP instances from [13]

Figure 3 shows the optimality gap, which is defined to be

$$\frac{\text{obtained solution} - \text{optimal solution}}{\text{optimal solution}} \times 100\%$$

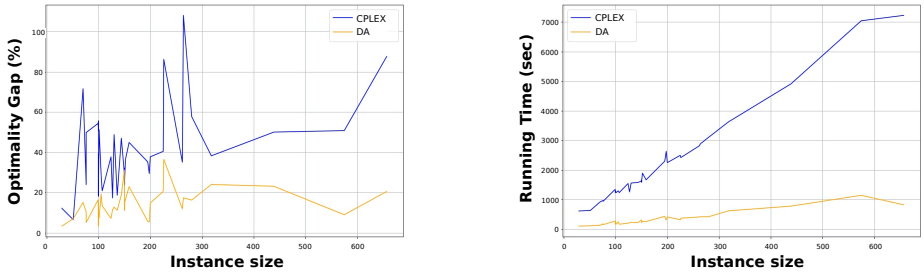
and tuning time when we apply the five online parameter tuning schemes presented earlier on the TSPLIB instances and Figure 4 shows the corresponding result on [13]. Tuning time refers to the average run time for solving the respective problem instances, which is incurred due to multiple calls to the QS as the underlying parameter is being tuned. For simplicity, we group the instances according to their size in reporting the optimality gap. The optimality gap can be obtained as our test problems are obtained from a repository where the optimal values of the instances have been reported. For both experiments, We observe that the optimality gap is comparable with each other for most instances, even though on average, Optuna performs the best for tsplib and PSO performs the best for tnm instances. In terms of computational time, PSO and the statistical approaches are faster than Hyperopt and Optuna.

We observe that tuning the penalty parameter uniformly with ratio between 0.5 to 1 of the maximum weight length suffices to obtain good results experimentally. However, this does not imply that careful tuning is not required. In general, a helpful principle we read from our experiments is that we need to first narrow the promising search space, from which we can perform a simple scheme such as uniform search to find the most appropriate values.



(a) Tnm Instances (b) TSPLIB Instances

Figure 5: Comparison of performance of our framework vs direct QS call



(a) Optimality Gap for TSP (b) Running Time for TSP

Figure 6: Comparison of CPLEX and QS run time by setting CPLEX early termination condition to reach DA optimality gap

Digital Annealer	
Iterations	10^7
Replica	72
TSP clustering	
Minimal cluster size, τ	7
Maximum cluster size, μ	30
Bayesian Tuner Approaches	
Number of trials	40
Range	$[0.5, 1] \cdot D_{\max}$
PSO hyperparameters	
Inertia weight $\omega_{\max}$	0.5
Inertia weight $\omega_{\min}$	0.25
Learning factor $c_{1,\max}$	0.5
Learning factor $c_{1,\min}$	0.25
Learning factor $c_{2,\max}$	0.9
Learning factor $c_{3,\max}$	0.6
Statistical Approaches	
Normal Distribution	$\mathcal{N}(\mu = 0.7594, \sigma^2 = 0.0141) \cdot D_{\max}$.
Uniform Distribution	$\mathcal{U}(0.5, 1) \cdot D_{\max}$
qbsolv	
sub-QUBO size	900
Optuna trials	20

Table 1: Parameter values used in our numerical experiments.

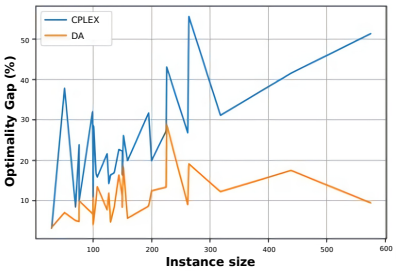


Figure 7: Comparison of CPLEX and QS optimality gap with run time limit of 2 minutes

5.3 Comparison with Other Approaches on TSP

Next we present two sets of experiments.

First, we compare the relative performance with direct call to QS and with the dedicated TSP solver Concorde. Note that due to the Digital Annealer's hardware limitation, the QS cannot directly solve instances larger than 90 cities. Hence, for purpose of direct comparison, we present results in Figure 5 based on instances of up to 90 cities from two data repositories - the Tnm instances which represents hard instances for Concorde on the left and the TSPLIB instances on the right. Using the same computational budget, we observe that the quality of solution of our approach is better, achieving an optimality gap that is significantly lower than if we directly

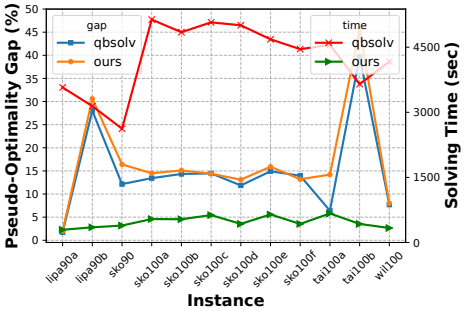


Figure 8: Comparison of performance using qbsolv vs our approach

use QS. We observe that when the instance size is more than 50 cities, the optimality gap can be more than 35% without using our approach, while it is less than 10% for our approach.

Note that we do not claim that our approach is the best in solving TSP, but to illustrate without any algorithmic trick (such as adaptive k-Opt with tabu search, as found in LKH), our approach can achieve solutions with very small optimality gap.

On the second set of experiments, we study the relative performance if we were to replace QS with an exact solver such as CPLEX instead. It is known that the required run time for an exact solver can be prohibitively long. In order to compare the performance of these two solvers fairly, we consider two experiment settings.

In the first setting, we solve each sub-QUBO model using CPLEX by setting the run time limit to be 2-minutes. The result is as shown in Figure 8. The result shows that within the time limit, *QS* outperfoms CPLEX solver and the gap grows as the instance such increases. In the second experiment setting, we fix the optimality gap and examine the computational budget required by the solvers to achieve a specified gap. For this purpose, after solving the E-TSP instances using QS, we note the optimality gap attained. Note that the optimality gap can be derived, since the problem instances are obtained from a repository with known optimal solutions. We then set a maximum time limit of 5 minutes in solving each sub-QUBO using CPLEX. Unlike the previous setting, another stopping condition that we use on CPLEX is that we set the required optimality gap for CPLEX to match the optimality gap obtained by QS. The result is as shown in Figure 6. Again, QS attains a smaller optimality gap within a shorter run time. In summary, we observe that QS performs better than CPLEX in solving the sub-QUBOs with either a fixed computational budget, or a fixed optimality gap.

5.4 Comparison with qbsolv on QAP

Finally, to demonstrate the generality of our approach, we turn to QAP. Here, we compare our approach with DWAVE's qbsolv, a state-of-the-art framework for solving QUBOs. Note that qbsolv itself is not a solver, but a framework that performs divide and conquer on a QUBO. Unlike our approach, one needs to construct the entire matrix explicitly as input to qbsolv. Furthermore, qbsolv does not ensure feasibility, since there is no notion of constraint violation.

The QAP instances and their best known solutions are obtained from QAPLIB [7], and in the interest of space, we report results of large-scale instances sampled randomly from the library. Note that the best-known solutions for the selected instances are attained by various heuristic methods including Genetic Hybrids [9] and Robust Tabu Search [26]. These solutions are termed as "pseudo-optimal" solutions in [27] (since optimality has not been proven). To compare with these approaches, we compute the pseudo-optimality gap and the results are shown in Figure **??**. Even though the pseudo-optimality gaps are significantly large compared with best-known solutions, we observe that the solution quality produced by qbsolv is very similar to our approach. And in terms of run time, we observe that our approach is at least 5 times faster (both approaches running DA as the QUBO solver).

6 CONCLUSION

We have proposed 3 techniques to improve the quality of solution of QS. These techniques can be used in a divide-and-conquer framework to solve large instances of combinatorial optimization problems. Experimentally, our approach yields solutions with very small optimality gaps on E-TSP instances (comparable with dedicated solver Concorde), and outperforms qbsolv, the state-of-the-art QUBO divide and conquer framework on QAP instances, with faster run time. Interestingly, our study shows that by using a divide-and-conquer framework along with our techniques, it outperforms the execution environment where we feed the problem to the QS directly. This could be attributed to the fact that as the problem size increases, the heuristic QS tends to perform worse due to various reasons such as large search space. Regrettably, the quality of solutions obtained by our approach for QAP is not entirely satisfactory relative to the best-known results, obtained albeit by multiple hand-crafted meta-heuristics. For our approach to be useful, the optimality gap needs to be narrowed further, perhaps with better parameter tuning and a stronger QUBO solver.

Interestingly, since our proposed ideas in this paper is agnostic to QUBO solvers, one could implement QS on a quantum hardware to derive a hybrid quantum-classical approach. When quantum annealers become more commercially viable, such approach can potentially disrupt exact solvers like CPLEX and Gurobi in solving large-scale optimization problems via mathematical modeling.

ACKNOWLEDGEMENTS

This research is funded by the National Research Foundation Singapore under its Corp Lab @ University scheme and Fujitsu Limited as part of the A*STAR-Fujitsu-SMU Urban Computing and Engineering Centre of Excellence.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2623–2631. https://doi.org/10.1145/3292500.3330701

[2] David Applegate, Ribert Bixby, Vasek Chvatal, and William Cook. 2006. Concorde TSP solver.

[3] Maliheh Aramon, Gili Rosenberg, Elisabetta Valiante, Toshiyuki Miyazawa, Hirotaka Tamura, and Helmut Katzgrabeer. 2019. Physics-inspired optimization for quadratic unconstrained problems using a digital annealer. *Frontiers in Physics* 7 (2019), 48. https://doi.org/10.3389/fphy.2019.00048

[4] James Bergstra, Dan Yamins, and David D Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*. Citeseer, 13–20. https://doi.org/10.1088/1749-4699/8/1/014008

[5] Endre Boros, Peter L Hammer, and Gabriel Tavares. 2006. Preprocessing of unconstrained quadratic binary optimization. (2006).

[6] Paul S Bradley, Kristin P Bennett, and Ayhan Demiriz. 2000. Constrained k-means clustering. *Microsoft Research, Redmond* 20 (2000).

[7] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. 1997. QAPLIB–a quadratic assignment problem library. *Journal of Global optimization* 10, 4 (1997), 391–403. https://doi.org/10.1016/0377-2217(91)90197-4

[8] IBM ILOG Cplex. 2009. V12. 1: User's Manual for CPLEX. *International Business Machines Corporation* 46, 53 (2009), 157.

[9] Charles Fleurent and Jacques A Ferland. 1994. Genetic hybrids for the quadratic assignment problem. *Quadratic assignment and related problems* 16 (1994), 173–187. https://doi.org/10.1090/dimacs/016/08

[10] Fred Glover, Gary Kochenberger, Rick Hennig, and Yu Du. 2022. Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models. *Annals of Operations Research* (2022), 1–43. https://doi.org/10.1007/s10288-019-00424-y

[11] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[12] Michael Held and Richard M Karp. 1970. The traveling-salesman problem and minimum spanning trees. *Operations Research* 18, 6 (1970), 1138–1162. https://doi.org/10.1007/BF01584070

[13] Stefan Hougardy and Xianghui Zhong. 2021. Hard to solve instances of the euclidean traveling salesman problem. *Mathematical Programming Computation* 13, 1 (2021), 51–74. https://doi.org/10.1007/s12532-020-00184-5

[14] Kazuki Ikeda, Yuma Nakamura, and Travis S Humble. 2019. Application of quantum annealing to nurse scheduling problem. *Scientific reports* 9, 1 (2019), 1–10. https://doi.org/10.1038/s41598-019-49172-3

[15] Roy Jonker and Anton Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* 38, 4 (1987), 325–340. https://doi.org/10.1007/BF02278710

[16] Xiaoyuan Liu, Hayato Ushijima-Mwesigwa, Avradip Mandal, Sarvagya Upadhyay, Ilya Safro, and Arnab Roy. 2019. On modeling local search with special-purpose combinatorial optimization hardware. *arXiv preprint arXiv:1911.09810* (2019).

[17] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. 2007. A survey for the quadratic assignment problem. *European journal of operational research* 176, 2 (2007), 657–690. https://doi.org/10.1016/j.ejor.2005.09.032

[18] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014), 5. https://doi.org/10.3389/fphy.2014.00005

[19] Muhammad Nawaz, E Emory Enscore Jr, and Inyong Ham. 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* 11, 1 (1983), 91–95. https://doi.org/10.1016/0305-0483(83)90088-9

[20] Gerhard Reinelt. 1991. TSPLIB—A traveling salesman problem library. *ORSA journal on computing* 3, 4 (1991), 376–384. https://doi.org/10.1287/ijoc.3.4.376

[21] Dian Palupi Rini, Siti Mariyam Shamsuddin, and Siti Sophiyati Yuhaniz. 2011. Particle swarm optimization: technique, system and challenges. *International journal of computer applications* 14, 1 (2011), 19–26. https://doi.org/10.5120/ijais-3651

[22] Tuhin Sahai, Anurag Mishra, Jose Miguel Pasini, and Susmit Jha. 2020. Estimating the density of states of Boolean satisfiability problems on classical and quantum computing platforms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1627–1635.

[23] Alice E Smith, David W Coit, Thomas Baeck, David Fogel, and Zbigniew Michalewicz. 1997. Penalty functions. *Handbook of evolutionary computation* 97, 1 (1997), C5.

[24] Tobias Stollenwerk, Elisabeth Lobe, and Martin Jung. 2019. Flight Gate Assignment with a Quantum Annealer. In *Quantum Technology and Optimization Problems*, Sebastian Feld and Claudia Linnhoff-Popien (Eds.). Springer International Publishing, Cham, 99–110. https://doi.org/10.1007/978-3-030-14082-3_9

[25] P. N. Suganthan. 1999. Particle swarm optimiser with neighbourhood operator. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Vol. 3. 1958–1962 Vol. 3. https://doi.org/10.1109/CEC.1999.785514

[26] Éric Taillard. 1991. Robust taboo search for the quadratic assignment problem. *Parallel computing* 17, 4-5 (1991), 443–455. https://doi.org/10.1016/S0167-8191(05)80147-4

[27] Eric D Taillard. 1995. Comparison of iterative searches for the quadratic assignment problem. *Location science* 3, 2 (1995), 87–105. https://doi.org/10.1016/0966-8349(95)00008-6

[28] Peng Tian, Jian Ma, and Dong-Mo Zhang. 1999. Application of the simulated annealing algorithm to the combinatorial optimisation problem with permutation property: An investigation of generation mechanism. *European Journal of Operational Research* 118, 1 (1999), 81–94. https://doi.org/10.1016/S0377-2217(98)00308-7

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

[30] Tony T Tran, Zhihui Wang, Minh Do, Eleanor G Rieffel, Jeremy Frank, Bryan O'Gorman, Davide Venturelli, and J Christopher Beck. 2016. Explorations of quantum-classical approaches to scheduling a mars lander activity problem. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

[31] Salvador E Venegas-Andraca, William Cruz-Santos, Catherine McGeoch, and Marco Lanzagorta. 2018. A cross-disciplinary introduction to quantum annealing-based algorithms. *Contemporary Physics* 59, 2 (2018), 174–197.

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

[33] Shengbin Wang, Weizhen Rao, and Yuan Hong. 2018. A distance matrix based algorithm for solving the traveling salesman problem. *Operational Research* (2018), 1–38. https://doi.org/10.1007/s12351-018-0386-1