SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

GECCO QuantOpt workshop 2022

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

Siong Thye Goh, Jianyuan Bo, Sabrish Gopalakrishnan, Hoong Chuin Lau*
Singapore Management University
Singapore

July 2022

Presented by Bo Jianyuan

Overview

- Introduction, Motivation, and Contributions
- Permutation-based Problems
- Techniques to Improve Solution Quality
- Application to TSP and QAP
- Numerical Experiments
- Conclusion

Introduction

- Permutation-based Combinatorial Optimization Problems (COPs), e.g.
  - Traveling Salesman Problem (TSP)
  - Flow Shop Scheduling Problem (FSP)
  - Quadratic Assignment Problem (QAP)
- Traditionally these problems are modeled as integer programs and solved with mathematical programming solvers such as
  - CPLEX
  - Gurobi
- Motivated by the promise of quantum computing, the idea of first formulating the problems as Ising models (or equivalently, quadratic unconstrained binary optimization problems (QUBO) is gaining traction which are amenable to quantum annealing-based solutions

Motivation

- Besides special hardware such as D-Wave's quantum annealer, there are there are technology companies have developed fast QUBO solvers (QS) such as
  - Alpha-QUBO (a software solver)
  - Fujitsu's Digital Annealer (DA)
- These QUBO solvers face several challenges including hardware limitations and no guarantee of solution quality
  - The data describing the QUBO **varies greatly in magnitude**
  - Choosing the right **penalty coefficient**
  - **Feasibility** of the solution returned by QUBO solver
  - Problem **size** constraints

Contributions

- ***Data-scaling***: to enhance solution quality, 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.
- ***Projection***: project infeasible solutions obtained by the QUBO solver to feasible solutions using a polynomial-time weighted assignment algorithm.
- ***Study the effects of the penalty parameters*** in the QUBO formulation to balance between quality and feasibility.
- ***Experimental results***: apply these techniques on DA QUBO solver to solve large Euclidean TSP (E-TSP) and 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).

Permutation-based Problems

- A permutation-based combinatorial optimization problem involves **permuting n objects** to minimize a certain objective function. Such problems can be modeled as minimizing a quadratic objective function of the following form:

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

$$s.t. \quad \sum_{u=1}^{n} x_{u,i} = 1, \forall i \in \{1, \dots, n\}$$ **Column sum constraints**

$$\sum_{i=1}^{n} x_{u,i} = 1, \forall u \in \{1, \dots, n\}$$ **Row sum constraints**

Permutation QUBO

- We can convert this formulation to an unconstrained QUBO model by **squaring the constraint violations** and adding them to the original objective function provided if A is large enough:

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

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

Data Scaling

Given quadratic objective function with permutation constraint (Large variation in magnitude)

Data Scaling

Equivalent quadratic objective function with permutation constraint (smaller variation in magnitude)

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

- Pick an index and change the magnitude
- Magnitudes are chosen to minimize the variation of the resulting objective function.

Penalty Parameter Tuning

- Tuning the penalty parameter A in Equation 3 is an important step in ensuring solution feasibility and quality for the original problem.

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

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

Penalty Parameter Tuning

- 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
  - **Online parameter search**
    - **Hyperopt** and **Optuna**: model-based Bayesian approaches
    - **Particle Swarm Optimization** (PSO): model-free evolutionary algorithm
  - **Statistical sampling:** Draw the parameter values from the fitted distributions based on the collected data by learning the average value and standard deviation of penalty parameter of the problem instance that performs well

Projection to the Feasible Space

- To restore feasibility of the original constrained problem, we solve the following optimization problem where $z \in \{0,1\}^{n \times n}$ is the infeasible solution returned from QS

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

$$s.t. \quad \sum_{i=1}^{n} x_{i,j} = 1, \forall j \in \{1, \dots, n\}$$

$$\sum_{j=1}^{n} x_{i,j} = 1, \forall i \in \{1, \dots, n\}$$

- Here z would be a given constant and hence this reduces to the standard **Weighted Assignment Problem** which can be solved in polynomial time with the Hungarian algorithm.

Application to TSP and QAP

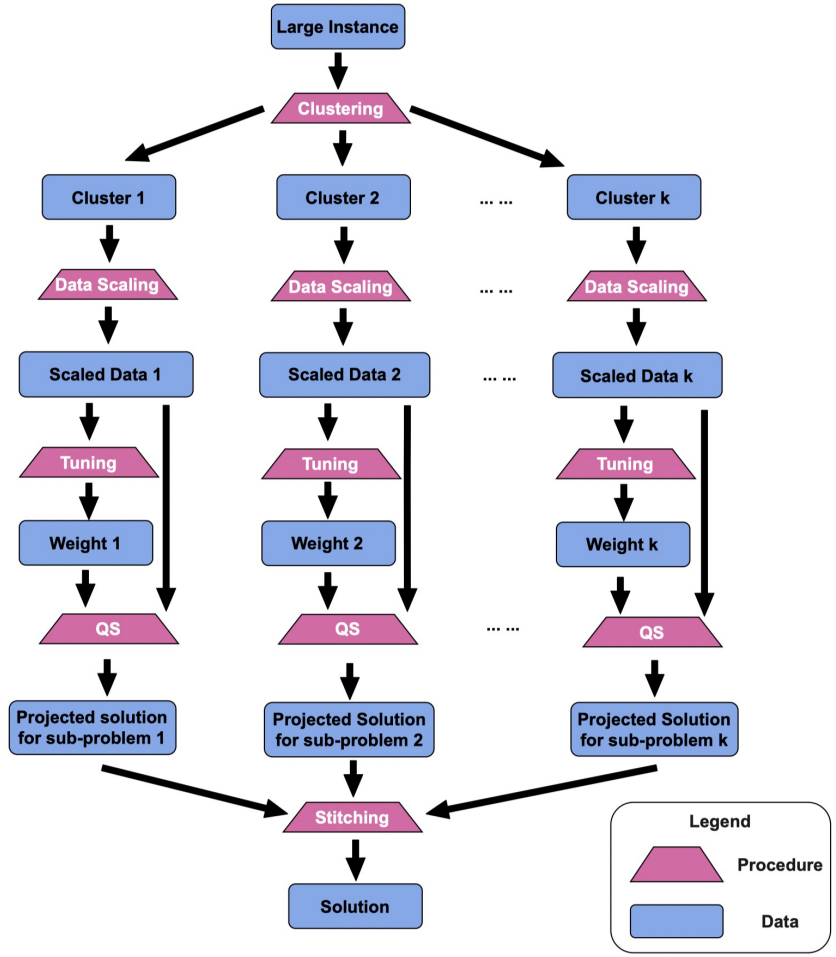


Figure 1: Divide and Conquer framework

E-TSP

- QUBO formulation for E-TSP is $\min_x H_B(x) + A \cdot H_A(x)$ where

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

Total distance travelled

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

Constraints to be a feasible cycle

- Notations
  - d_{uv} is the distance between city u and city v
  - $x_{v,j}$ is the indicator variable that the city v is the j-th city to be visited. And we require total n^2 variables for an n-city instance.

Stitching the Clusters for E-TSP

- Let k be the number of clusters and largest cluster size be $|V_C|$.
- We then 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** instead of cycle) with Δ_{ij} on the edges.

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.
- The total flow can be written as a QUBO with indicator variable x_{ij} denoted that facility i is assigned to location j:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} f_{ij} d_{kl} x_{jk} x_{jl}$$

Clustering Scheme for QAP

- 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.

Numerical Experiments

- Investigate the effect of data scaling on TSP solution quality
- Compare the relative performance on TSP solution quality under different parameter tuning approaches.
- 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 as well as hard TNM instances. We also compare the relative performance when using an exact solver (CPLEX) directly instead of a heuristic QS.
- 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.

Effect of Data Scaling on TSP

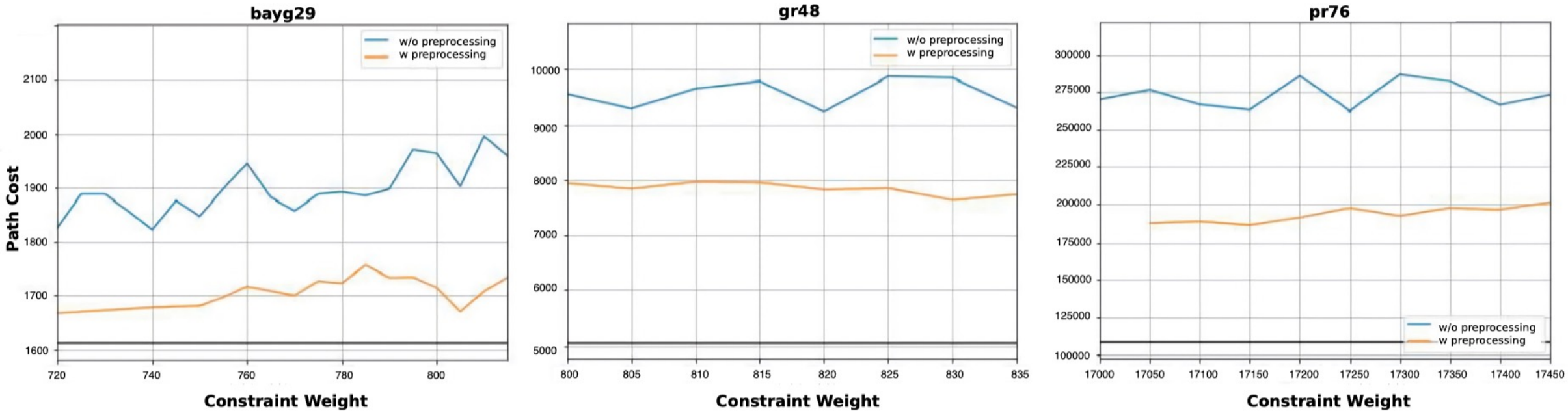
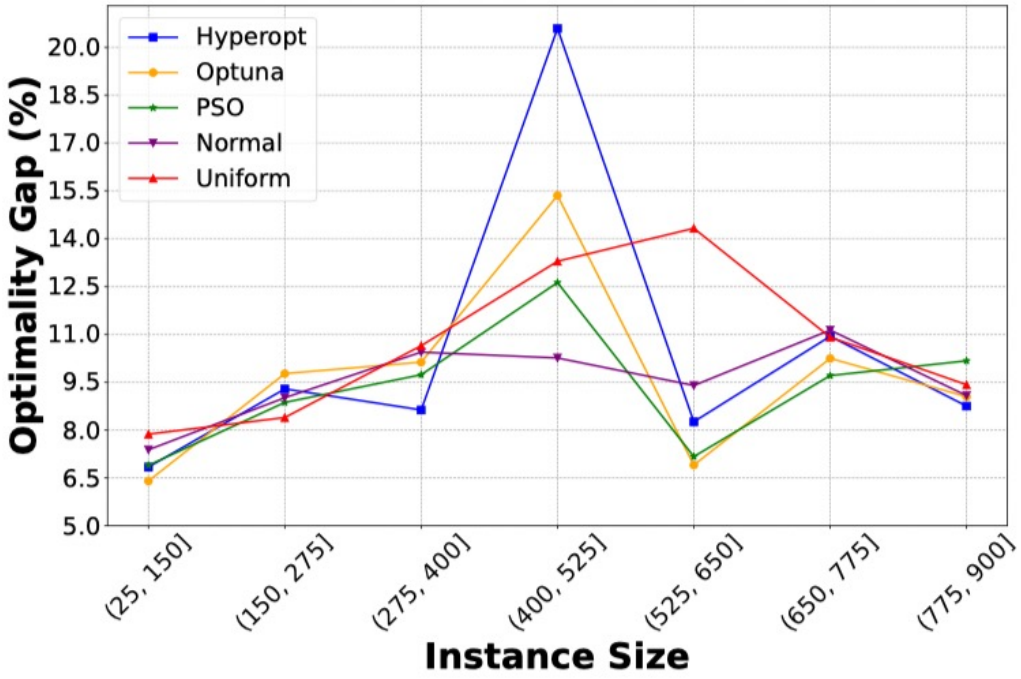
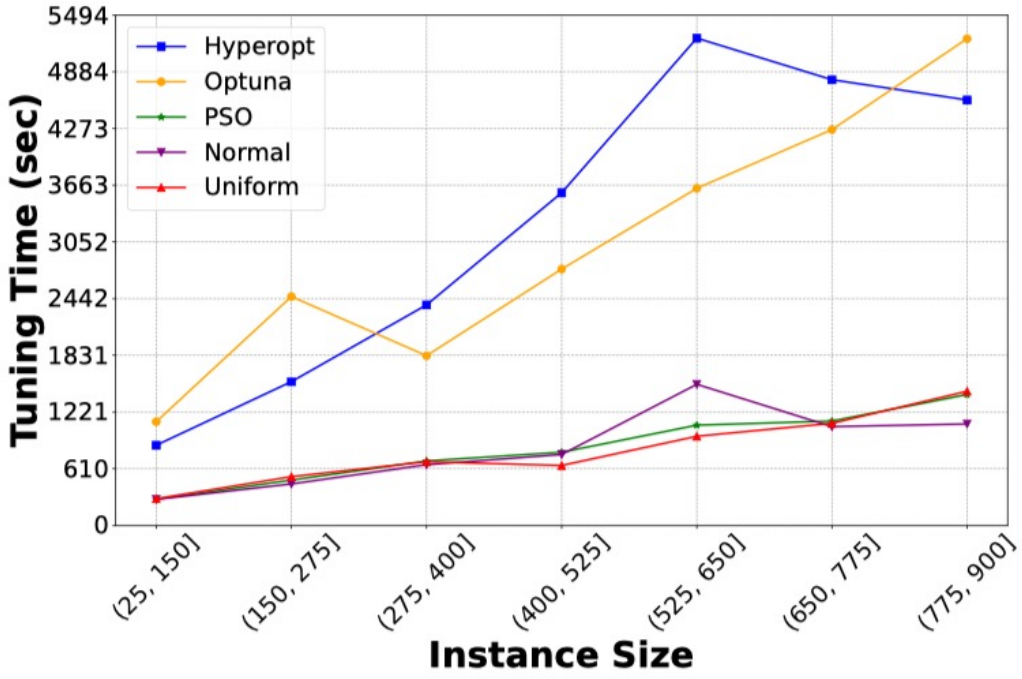


Figure 2: Lower objective value is obtained after performing data scaling

Effect of Parameter Tuning on TSP



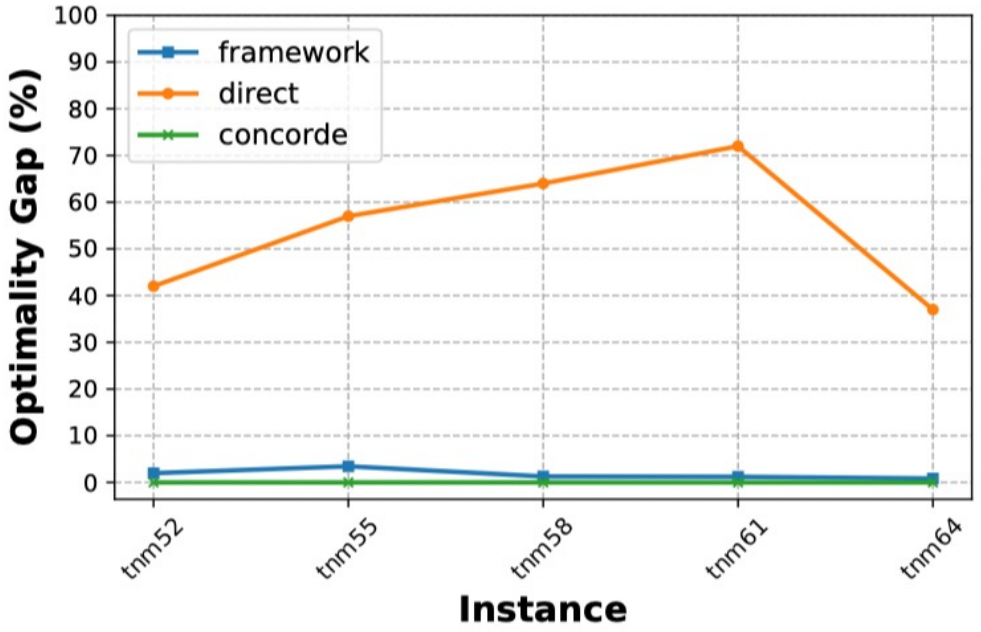
(a) Optimality Gap



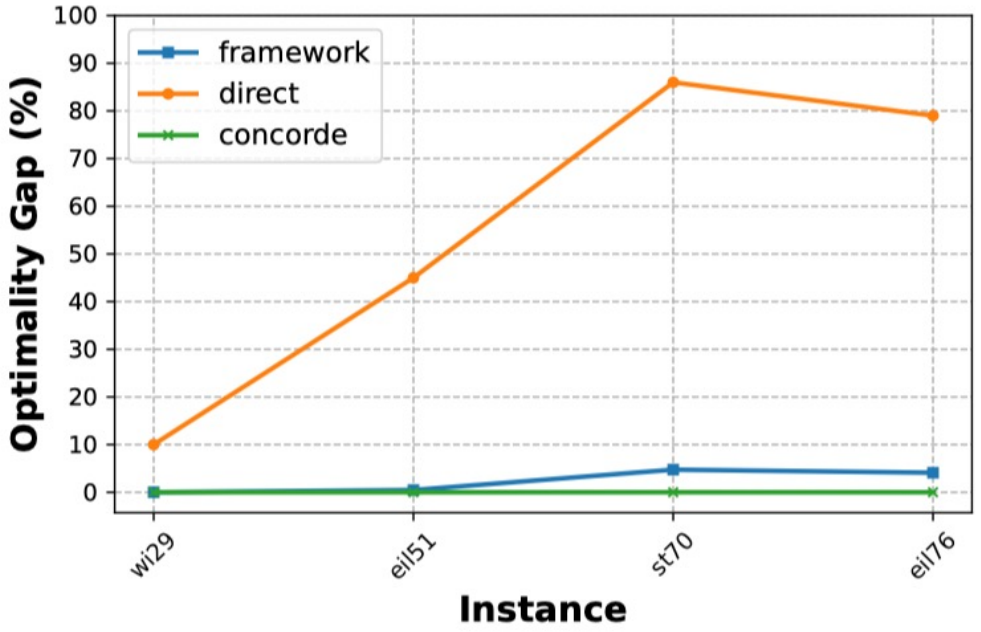
(b) Tuning Time

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

Comparison with Other Approaches on TSP



(a) Tnm Instances



(b) TSPLIB Instances

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

Comparison with qbsolv on QAP

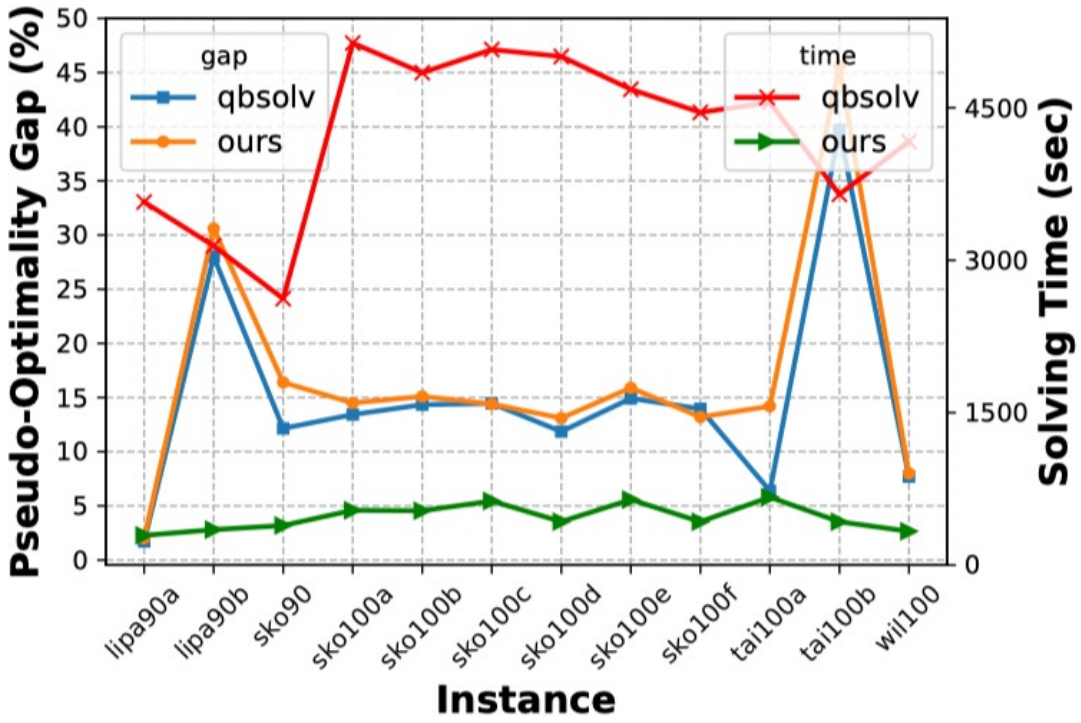


Figure 8: Comparison of performance using qbsolv vs our approach

Conclusion

- **Three techniques** to improve the quality of solution of QS.
  - Data-scaling
  - Penalty parameter tuning
  - Projection
- 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.
- 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.
- 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.