INTEGRATION OF NEURAL SOLVER AND PROBLEM-SPECIFIC SOLVER THROUGH BILEVEL APPROACH: A CASE STUDY OF MIN-MAX CAPACITATED VEHICLE ROUTING PROBLEM

Anonymous authors
Paper under double-blind review

ABSTRACT

In real-world operations with combinatorial structures like vehicle routing problems, similar optimization problems have to be solved repeatedly with slight parameter variations. A key challenge in such scenarios is achieving both high solution quality and fast computation time, while traditional methods like heuristics or branch-and-bound struggle to achieve both simultaneously. In contrast, problem-specific solvers can effectively balance solution quality and computation speed for specific problems. However, since real-world problems have more complex structures, they can handle only subproblems. To enhance the applicability of the problem-specific solvers, we propose a framework that integrates a problem-specific solver and a neural solver. Our framework decomposes the optimization problem into subproblems so that some of which can be solved by problem-specific solvers, such as the traveling salesperson problem. For the remaining portions of the problem, we utilize the similarities of the problems and design a neural solver. By integrating two solvers, we can utilize the strengths of the problem-specific solver in balancing solution accuracy and computation speed, as well as the neural solver ' s ability to infer a solution from the similarity of optimization problems. Based on the case study with the min-max capacitated vehicle routing problem, we demonstrate that it outperforms the state-of-the-art solver regarding both high solution quality and short computation time.

1 INTRODUCTION

When solving combinatorial optimization problems for real-world operations, optimization problems often need to be solved repeatedly, with similar structures and slightly varying parameters. For example, in the delivery of liquefied petroleum gas cylinders, the problem needs to be solved per day for the same area (Yoshida et al., 2024), the waste collecting problem is solved per day (Han et al., 2024), and in food delivery, the problem of assigning each order to each driver needs to be solved every few minutes (Simoni & Winkenbach, 2023). In such frequently occurring tasks, quickly obtaining high-quality solutions is crucial to ensure operational efficiency.

Although various studies have been conducted to solve combinatorial optimization problems, existing methods do not fully balance the solution accuracy and the computation time for real-world problems. (see Figure 1) As exact methods, branch-and-bound or branch-and-cut (Achterberg, 2007) can be used to obtain the optimal solutions, while it is challenging to incorporate such algorithms into real-world operations because the computation time exponentially increases along with the problem size. On the other hand, heuristics such as genetic algorithms have been studied. While better solutions can be obtained with appropriate algorithm selection and hyperparameter settings, it is difficult to obtain high-quality solutions quickly for large-scale problems with many constraints. For specific problems, algorithms capable of both speed and accuracy have been developed, such as those for the traveling salesperson problem (Applegate, 2006) and the capacitated vehicle routing problem (Helsgaun, 2017).

We observed that the combinatorial optimization problems in real-world operations often contain subproblems, such as the traveling salesperson problem (TSP) and knapsack problem, for which

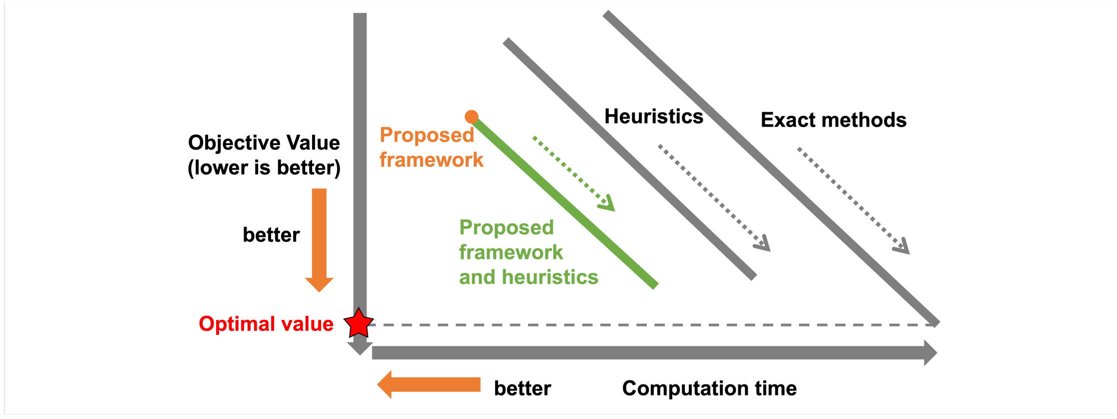


Figure 1: Comparison of the proposed framework with heuristics and branch-and-bound in terms of computation time and objective function value. While exact methods guarantee optimal solutions, they require considerable computation time. Heuristics can quickly find moderately good solutions but struggle to reach high-quality solutions. The proposed framework balances computation speed and solution quality by learning similar problems in the training dataset.

problem-specific solvers have been developed. For example, in a multi-day delivery planning problem, once the customers for each vehicle are determined for a given day, the problem of finding the delivery route can be decomposed to solve the TSP per vehicle independently. Considering this fact, we can effectively solve the large-scale combinatorial problem with a problem-specific solver for subproblem computation if we can solve the remaining problem without a problem-specific solver. The field that deals with optimizing through subproblems solutions is known as bilevel optimization. Bilevel optimization refers to problems where decision-making occurs in two stages, with decisions in the first stage affecting those in the second stage. The optimization problems in the first and second stages are called the upper- and lower-level problems, respectively. A typical bilevel optimization method is the nested approach, in which the lower-level problem corresponding to the upper-level problem's solution is solved, and based on the result, the solution to the upper-level problem is sequentially updated (Mathieu et al., 1994; Yin, 2000; Sinha et al., 2017; Calvete et al., 2008; Liang & Miikkulainen, 2015; Daniel Handoko et al., 2015). To solve real-world problems, the bottleneck in the nested approach is the number of updates of the upper-level solution. Thus, reducing the number of updates is a key challenge to achieving high-quality solutions using the nested approach with the problem-specific solver.

To reduce the number of updates, obtaining a good initial solution for the upper-level problem is important. When the upper-level problem is solved repeatedly, solutions obtained by past computations can be utilized to obtain a better initial solution. When the upper-level problems are repeatedly solved, constraints change little, and the optimization problem solved repeatedly only involves slight changes in parameters. Generally, under slight changes in coefficients, the optimal solution to one problem often serves as a near-optimal solution for another. Under this assumption, we can train a neural network to learn the characteristics of the problem and rapidly acquire good heuristic solutions. In fact, when the upper problem has no constraints, a machine learning method is proposed to infer a solution to the upper-level problem (Guo et al., 2024).

Regarding the use of machine learning models quickly to obtain better solutions to optimization problems, there are challenges in obtaining solutions that satisfy the constraints. An example of post-processing to satisfy simple constraints is using an autoregressive model that repeatedly selects one node at a time when solving the TSP with neural networks, as widely studied (Kwon et al., 2020). In this model, masking is applied to already visited nodes, allowing for selecting the next node from unvisited nodes, thus obtaining a path that does not visit the same node twice. Similarly, Hou et al. (2023) proposed a method that considers capacity constraints using a mask matrix, but this is limited to vehicle routing problems. Another well-known example is using the softmax function at the output of the final layer in classification problems to ensure that all elements are non-negative and sum to one. The LinSatNet was proposed (Wang et al., 2023), which improves the differentiable optimal transport, Sinkhorn algorithm (Cuturi, 2013). It extended the types of constraints it can handle to linear constraints composed of non-negative coefficients. However, it cannot be applied to constraints like flow conservation involving non-negative coefficients.

We summarize these considerations; (i) bilevel strategy incorporating a neural-solver and problem-specific solver is a promising method to achieve both high solution quality and short computation time for target problems (see Figure 1), (ii) constraint-enforcing for the neural network output can be achieved by the differentiable optimal transport for non-negative linear constraints. Based on them, we propose a framework that integrates machine learning, optimal transport, and a problem-specific solver to obtain high-quality solutions quickly for large-scale combinatorial optimization problems. Our framework adopts a self-supervised learning method for upper-level problems, whereas specialized methods can be utilized for lower-level problems. For the upper-level problem, we train a neural network to learn problem characteristics with satisfying the constraints by utilizing the differentiable optimal transport module as the output layer. This makes it possible to construct a solution while ensuring feasibility, including the lower-level problem, by satisfying multiple linear constraints composed of non-negative coefficients. Moreover, since the proposed framework separates the neural solver that learns problem characteristics and the optimal transport module that projects the solution onto the constraints, it is a method that can adapt to adding new constraints during inference. After constructing a sufficiently trained model, we can acquire a high-quality solution quickly compared to existing methods such as heuristics and exact methods.

To verify our proposed framework's effectiveness, case studies were conducted with the min-max capacitated vehicle routing problem (min-max CVRP). It contains the TSP for which the problem-specific solver has been developed after obtaining the assignment locations for vehicles. Compared to LKH-3[*], our framework obtained high-quality solutions by comparing quickly. This tendency is much more obvious as the problem size increases.

Our contributions are summarized as follows,

- We propose the self-supervised learning framework integrating the neural solver and the problem-specific solver by reformulating an optimization problem as a bilevel one. (Section 3)
- Based on the case study with min-max CVRP, we show that our framework can obtain better solutions than state-of-the-art-solver LKH-3 in a short time computation. (Section 4) Also, an appropriate data selection method for training the neural solver is discussed. (Section 5)

2 PRELIMINARIES

This section provides the preliminaries about bilevel optimization and constraint-encoding for neural networks, which need understanding our framework. Then, we denote the min-max capacitated vehicle routing problem, which is utilized as the case study.

2.1 BILEVEL OPTIMIZATION

Bilevel optimization is a method in which parameters influencing the solution of one optimization problem are determined by solving another optimization problem. The former and latter problems are called upper- and lower-level problems, respectively. The general formulation is as follows:

$$\begin{aligned} \underset{x \in X}{\text{minimize}} \quad & F(x, y^*(x)) \\ \text{subject to} \quad & y^*(x) = \underset{y \in Y(x)}{\text{argmin}}\, f(x, y) \end{aligned}$$

Here, x and y are the decision variables of the upper-level and lower-level problems, respectively. $F(x, y)$ is the objective function of the upper-level problem, while $f(x, y)$ represents the objective function of the lower-level problem. $y^*(x)$ is the solution to the lower-level problem for a given x.

[*]http://webhotel4.ruc.dk/~keld/research/LKH-3/

2.2 Constraint-encoding for neural networks

In general, there is no guarantee that the output of a neural network satisfies specific constraints. Some recent research has been done on the differentiable optimal transport that can be inserted into the intermediate or last layer. One approach to impose constraints on intermediate neural network outputs is the differentiable optimal transport method known as the Sinkhorn algorithm (Cuturi, 2013). Wang et al. (2023) proposed the LinSatNet to handle the more general problem with the multiple linear constraints with non-negative coefficients. This paper applies LinSatNet to ensure the solution satisfies linear constraints in the upper-level problem, detailed in Sections 3 and 4.

2.3 min-max Capacitated Vehicle Routing Problem

In this paper, we used the min-max Capacitated Vehicle Routing Problem (min-max CVRP) as the case study to verify the proposed framework's effectiveness. In order to apply our proposed framework to min-max CVRP, we formulate it as a bilevel optimization problem (Problem 2.1) where n, K, Q, d_{ij} denote the total number of locations, the total number of vehicles, the capacity of the vehicles, and the distance between locations i and j, respectively. For simplicity, we assume that each vehicle has the same capacity Q, and the amount of cargo transported to each location is one. The upper-level problem is assigning customers to vehicles, and the lower-level problem is solving a TSP for each vehicle. The upper-level problem contains capacity constraints and the need to ensure that vehicles visit each location exactly once, and the coefficients of these constraints are non-negative; hence, LinSatNet can handle both constraints. Note that the index 1 of location represents the depot. The formulation of min-max CVRP is as follows,

Problem 2.1 Min-Max Capacitated Vehicle Routing Problem (min-max CVRP) formulated as the bilevel optimization

$$\begin{aligned}
\underset{x\in X}{\text{minimize}} \quad & F(x, y^*(x)) = \max_k \left(\sum_i \sum_j d_{i,j} y^*_{(i,j),k} \right) \\
\text{subject to} \quad & y^*(x) = \underset{y\in Y(x)}{\text{argmin}}\, f(x,y) = \underset{y\in Y(x)}{\text{argmin}} \sum_i \sum_j d_{i,j} y_{(i,j),k}
\end{aligned}$$

$$\text{where } X = \left\{ x \;\middle|\; \sum_k x_i^k = 1 \quad (\forall i), \quad \sum_i x_i^k \le Q \quad (\forall k) \right\},$$

$$\begin{aligned}
Y(x) = \Bigg\{ y \;\Bigg|\; & \sum_j y_{(i,j),k} = x_i^k \quad (\forall i,k), \sum_j y_{(1,j),k} = 1 \quad (\forall j,k), \quad \sum_i y_{(i,1),k} = 1 \quad (\forall i,k), \\
& x_i^k - x_j^k + (n-1) y^*_{(i,j),k} \le n-2 \quad (\forall i,j \in \{2,3,\cdots,n\}, k) \Bigg\}.
\end{aligned}$$

- x_i^k : Variable that is assigned 1 when the vehicle k visits location i otherwise it is assigned 0
- $y_{(i,j),k}$: Variable that is assigned 1 when the vehicle k travels from location i to location j otherwise it is assigned 0

The additional examples for applying the proposed framework to other use cases are discussed in Section 5.3.

3 Proposed Framework

In this section, we propose a framework for obtaining a high-quality solution to combinatorial optimization problems quickly through a bilevel approach. When we solve similar problems repeatedly, high-quality solutions share a similar structure (e.g., assigning vehicles to locations in a vehicle routing problem) while the local structure is different (e.g., visiting the order of locations in a vehicle routing problem). Based on the assumption, we treat the problem as bilevel optimization, where the problem-specific solver can effectively solve the lower-level problem to deal with the local characteristics per problem. In fact, the combinatorial optimization problem often contains the subproblem for which the problem-specific solvers have been developed, such as Concorde for TSP, LKH-3 for vehicle routing problems, and CP-SAT solver [†] for scheduling problems. Note that a general mathematical optimization solver, such as Gurobi Optimizer, can be used as an alternative to a problem-specific solver when it can efficiently solve the lower-level problem. Given that similar problems are repeatedly solved, we can use a dataset of these solutions to train a neural network for the upper-level problem. In our framework, we utilize self-supervised learning to train the neural network, allowing the training without high-quality solutions for the dataset. There are four following components and the workflow is shown in Figure 2.

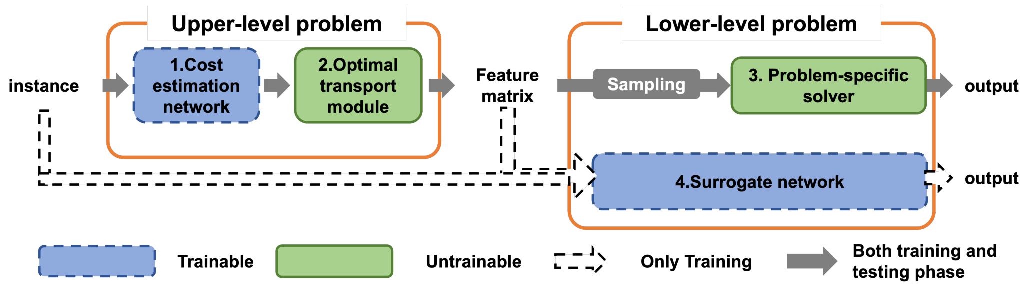


Figure 2: Proposed self-supervised learning framework through bilevel approach

1. Cost estimation network (for upper-level problem)
2. Optimal transport module (for upper-level problem)
3. Problem-specific solver (for lower-level problem)
4. Surrogate network (for lower-level problem)

To solve the combinatorial optimization problem through bilevel approach, it is necessary to obtain both the solution to the upper-level problem and the corresponding solution to the lower-level problem. The framework incorporates a cost estimation network and an optimal transport module to obtain the solution to the upper-level problem while satisfying the constraints. The optimal transport module ensures that the output satisfies linear constraints in the upper-level problem. To evaluate the solution of the upper-level problem, the lower-level problem must be solved to obtain the objective value of the original problem. The output of the optimal transport module does not ensure the integrality of discrete variables; we can acquire the feasible solutions to the upper-level problem by executing the postprocessing like LinSatNet (Wang et al., 2023). For example, we can use sampling from the feature matrix as the postprocessing. After determining each lower-level problem by sampling the output of the optimal transport module, a problem-specific solver is executed to obtain the solution of the lower-level problem. Due to the necessity of sampling during this process, backpropagation cannot be performed directly for updating the parameter of the cost estimation network. To address this problem, a surrogate network is prepared to approximate the output of the problem-specific solver, enabling the learning of the cost estimation network.

The training process is executed as follows: a) The cost estimation network performs a forward pass, followed by calculating the optimal transport module to obtain the solution to the upper-level problem. b-1) Sampling is performed from the output of step a), and the problem-specific solver is executed for each lower-level problem. b-2) Alternatively, a forward pass is performed through the surrogate network using the output of step a) and problem information. c) Loss functions are computed to backpropagate through the surrogate and cost estimation networks. During inference, only step b-1) is performed after step a).

[†]For example, the solver is provided by a part of Google OR-tools: `https://developers.google.com/optimization/cp/cp_solver`

We explain the loss functions which should satisfy the following three properties:

L1. The output of the surrogate network approximates the solution obtained by the problem-specific solver.

L2. The solution of the upper-level problem minimizes the objective function value of the problem-specific solver.

L3. The output of the cost estimation network avoids the similar output for different inputs, which is the regularization term.

4 Case Study : min-max Capacitated Vehicle Routing Problem

This section provides the case study using the proposed framework to solve the min-max capacitated vehicle routing problem.

4.1 Problem Setting

We conducted experiments on the min-max capacitated vehicle routing problem (min-max CVRP), whose formulation is shown in Section 2.3. It is an example of a problem that can be decomposed into several steps, and problem-specific solvers have been developed for the lower-level problem.

We denote the n and K as the number of locations and vehicles, respectively, as shown in Section 2.3. Let Z be the problem containing the coordinates of each location. The cost estimation network, surrogate network, and optimal transport module are denoted as f, g, s, respectively. The output of the cost estimation network is denoted as $P := f(Z)$, where each element $P_{i,k}$ indicates that the probability of location i is assigned to vehicle k. By utilizing the matrix P and the vehicle capacity Q, we obtain π by executing the optimal transport module. $g(\pi, Z)$ is the output of the surrogate network. $T : 2^{\{1,2,\cdots,n\}} \to \mathbb{R}$ is the function to obtain the shortest traveling path given locations by utilizing the problem-specific solver. $a_k \subset \{1, 2, \cdots, n\}$ is the set of locations assigned to the vehicle k. For min-max CVRP, the loss function design based on the proposed framework outlined in Section 3 is concretely defined as follows:

L1. $\sum_k (T(a_k) - g(\pi, Z)_k)^2$: The surrogate network accurately approximate the maximum route length.

L2. $\max_k (g(\pi, Z))_k \times (-\log \prod P_{i,k_i^*})$, where $k_i^* := \operatorname{argmax}_k \pi_{i,k}$: The upper-level problem's solution make the maximum route lengths smaller.

L3. $-\sum P_{i,k} \log P_{i,k}$: The each column ofthe cost estimation network's output is close to one-hot representations through regularizing the entropy, preventing ambiguous assignments of locations to vehicles.

We set the loss function as the summation of the above three terms. This formulation ensures that the framework effectively handles the min-max CVRP while optimizing for both upper- and lower-level problem solutions.

4.2 Experimental Setting

Following Goh et al. (2024), we utilized the realistic approach of random sampling from the base map, not utilizing randomly generated instances. As sample data, we acquired the map in Munich, Paris, and Barcelona by utilizing OSMnx[‡], each of which has different characteristics. The feature and visualization of each map are shown in Table 1 and Figure 3, respectively. The coordinate is originally normalized to $[0, 1]^2$. We sampled 100 locations per problem, and we regarded them as nodes. Then, we acquired the problem as the completed graph. The depot was also randomly sampled from one node per problem. The cost of traveling between two locations is defined as the Euclidean distance.

[‡]`https://osmnx.readthedocs.io/en/stable/`

City	Number of locations	Feature
Munich	14,046	There are many small-sized clusters.
Paris	9,484	Locations are not aggregated.
Barcelona	8,914	Locations are aggregated with different densities.

Table 1: Description of input data

100,000 instances, 128 instances, and 100 instances were extracted from each map as training, validation, and test data, respectively. As a preprocessing step, the position of the depot was translated to the origin. Subsequently, data augmentation was applied by flipping the coordinates along the x-axis and y-axis, each with a probability of 0.5. LKH-3 was used as a comparative method, and the time limits for each experiment were set to 30 seconds, 60 seconds, and 120 seconds. The computational environment used was one NVIDIA RTX A6000. The optimizer of the cost estimation and surrogate networks is RMSprop, where the learning rate is set to 1e-5. We set the batch size to 128. We trained 300 epochs and calculated the objective values for validation data every ten epochs to save the pretraiend model. The number of vehicles K is set to 5. The capacity of vehicle Q is set to $\lfloor \frac{n}{4} \rfloor$, so there is a gap between the total number of locations and the sum of vehicles' capacity. Therefore, we can allow the gap between the number of locations to visit per vehicle.

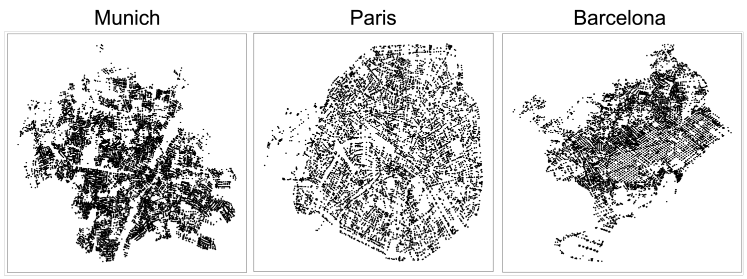


Figure 3: Scatter plot of three cities: Munich, Paris, and Barcelona.

We prepared the components of the proposed framework for min-max CVRP shown as follows;

- 1. Cost estimation network: three-layered Graph Isomorphism Network (Xu et al., 2019)
- 2. Optimal transport module : LinSatNet (Wang et al., 2023) described in Section 2.2
- 3. Surrogate network: three-layered attention network (Vaswani, 2017)
- 4. Problem specific-solver: Google OR-tools. Note that it is not an exact solver.

Additionally, the output of each method is used as an initial solution, and an evaluation is performed on the solution after conducting a local search. At the test phase, each location's assignment to the vehicle is sampled from the feature matrix per column. However, the assignment can be infeasible because the feature matrix does not consider the integrality of the variables. Therefore, we implemented the postprocessing to obtain the discrete solution, detailed in Appendix A. Note that we can obtain a feasible solution for many (but not all) instances by the sampling, detailed in Appendix B.

4.3 Result

As a quantitative evaluation, we show Figure 4 by utilizing the map of Paris, Munich, and Barcelona, respectively. Our framework succeeded in obtaining a better solution than LKH-3. Table 2 shows the evaluation metric with different methods for 300 locations per problem. The proposed framework outperformed the LKH-3 with a time limit of at most 120s. Moreover, the proposed framework is much faster than LKH-3. Regarding the "Best Results" column, when multiple methods achieved the best objective function value, both were included in the count. As a result, the total count can exceed 100, which is the total number of test problems. Figure 5 shows the relative objective value normalized by the output of LKH-3 with a limit of 120s with different problem sizes. The gap between the solution quality of our framework and that of LKH-3 becomes larger as the problem size increases, which shows that the proposed framework learns the characteristics of similar problems. As a qualitative evaluation, Figure 6 shows each vehicle's path output from our framework to LKH-3 with different timelimit settings. While LKH-3 contains redundant routes due to an imbalance of workload, our framework has almost the same workload among vehicles.

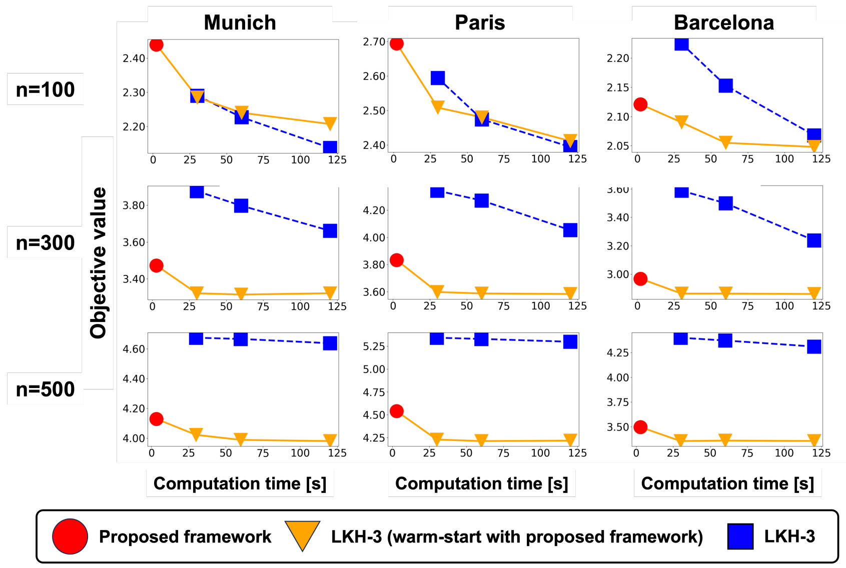


Figure 4: Average objective values with different problem sizes and inputs: Each row and column indicate the result of the same problem size and input data, respectively.

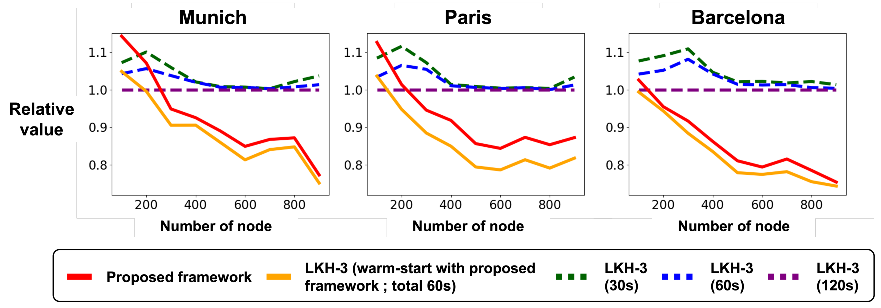
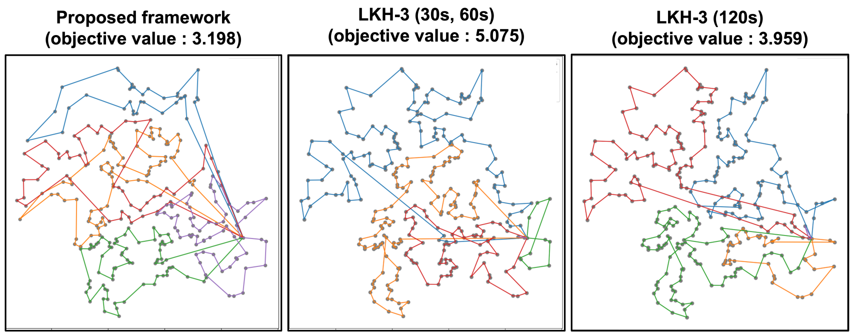


Figure 5: Absolute objective values with different problem sizes normalized by the objective value of the experiment in LKH-3 with a limit of 120s



Proposed framework (objective value : 3.198)
LKH-3 (30s, 60s) (objective value : 5.075)
LKH-3 (120s) (objective value : 3.959)

Figure 6: Visualization of output route for 300 nodes' problem obtained by proposed framework and LKH-3 with different timelimit settings

		Average objective value (↓)	Best results (↑)	computation time[s] (↓)
Munich	Proposed	3.47 (89.6%)	26/100	**2.7**
	Proposed + LKH-3 (total 60s)	**3.31 (85.5%)**	**40/100**	60.0
	LKH-3 (30s)	3.88 (100.0%)	20/100	30.0
	LKH-3 (60s)	3.80 (98.0%)	23/100	60.0
	LKH-3 (120s)	3.66 (94.5%)	26/100	120.0
Paris	Proposed	3.83 (88.3%)	20/100	**2.5**
	Proposed + LKH-3 (total 60s)	**3.59 (82.6%)**	**53/100**	60.0
	LKH-3 (30s)	4.34 (100.0%)	14/100	30.0
	LKH-3 (60s)	4.27 (98.4%)	17/100	60.0
	LKH-3 (120s)	4.05 (93.3%)	25/100	120.0
Barcelona	Proposed	2.97 (82.7%)	32/100	**2.2**
	Proposed + LKH-3 (total 60s)	**2.86 (79.8%)**	**48/100**	60.0
	LKH-3 (30s)	3.59 (100.0%)	8/100	30.0
	LKH-3 (60s)	3.50 (97.5%)	7/100	60.0
	LKH-3 (120s)	3.24 (90.2%)	18/100	120.0

Table 2: Average objective value in case 300 node for different test data; For the "Best Results" column, when multiple methods achieved the best objective function value, both were included in the count. As a result, the total count can exceed 100, which is the number of test instances.

5 Discussion

This section discusses the additional topics about the proposed framework.

5.1 Generality for other area's data

We executed the inference where the training data is different from the inference data to discuss the generalization of the trained model. We show the result in Table 3. The pretrained model with Barcelona data achieved the best results also Munich and Paris data than pretrained model with their own training data, respectively. Figure 7 shows the histograms of the distances of every pair of locations for maps utilized in the experiments and randomly generated data. Based on the figure, there are small clusters in Barcelona map and we assume that the key to enhance neural solver's generality for vehicle routing problem in real-world's data. In addition, since randomly generated data has no cluster, the characteristics of the distribution of histograms are different from those of other data. It can be assumed that the randomly generated is not suitable for training with real-world data.

		Training data		
		Munich	Paris	Barcelona
Test data	Munich	2.44	2.42	**2.21**
	Paris	2.70	2.69	**2.52**
	Barcelona	2.31	2.32	**2.12**

Table 3: Average objective value when the training data is different from the testing data for problems with 100 nodes

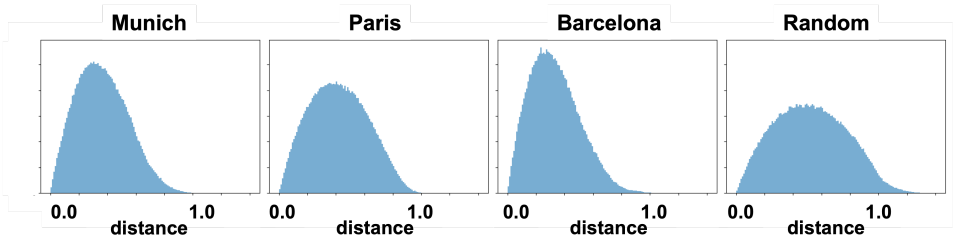


Figure 7: The histograms of the distances calculated by every pair of locations in test dataset

5.2 Applicability to additional constraint at test phase

One advantage of the proposed framework is that it can handle constraints that were not given at the training phase. This is because the optimal transport module can handle the input constraint regardless of the constraints during training. We experimentally add the constraint that half of the nodes cannot be assigned to the first truck plotted with blue color in Figure 8. Figure 8 shows the visualization when the constraint is not added or added. One node plotted by a red star in the left figure was originally assigned to the first truck, whose path is shown in blue. After adding the additional constraint to prohibit the node from assigning the first truck, it can naturally be reassigned to another truck plotted in green. It is one example of how the cost estimation network and the optimal transport module are robust when adding the constraint.

5.3 Other applications to utilize the proposed framework

As a case study, we utilized min-max CVRP to verify the proposed framework. Note that the condition of utilizing the proposed framework is that it can decompose the problem that exists in the problem-specific solver. For example, in a flexible job shop scheduling problem, when we extract the upper-level problem as the determining assignment between tasks and machines. We can formulate the lower-level problem as the job shop scheduling, which can be solved efficiently with the CP-SAT solver. Therefore, we can apply a proposed framework to solve the flexible job shop scheduling problem.

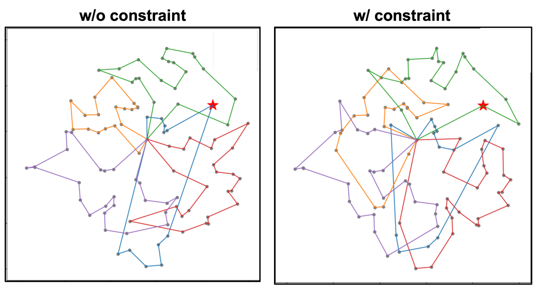


Figure 8: The left figure shows the output of a test problem, while the right figure shows the output after adding an additional constraint. The additional constraint is that the first vehicle cannot visit the former locations, and the second vehicle cannot visit the latter locations.

6 Conclusion

This study proposes a self-supervised framework for obtaining high-quality solutions quickly to large-scale combinatorial optimization problems by integrating neural and problem-specific solvers. Our framework decomposes the combinatorial optimization problem into subproblems, some of which have problem-specific solvers, such as the TSP. For the remaining problem, we construct the neural solver with differentiable optimal transport, ensuring feasibility. We validated the effectiveness of the proposed framework through experiments on the min-max capacitated vehicle routing problem.

For future work, we show two research directions. The first direction is to validate our framework with the dataset of real-world problems. Since we did not obtain the real-world problems, we made problems by random sampling from the map. The other direction is to handle more general constraints in the upper-level problem. In our framework, we currently cannot handle linear constraints with negative coefficients or non-linear constraints; if we can handle such constraints, our framework can be utilized to solve more general problems. By addressing these issues, our framework will contribute to integrating the neural and mathematical optimization solvers for fast and accurate optimization.

Reproducibility

We attached the source code as the supplementary materials through which we can reproduce our result. The user also downloads and utilizes the LKH-3, the official site.

References

Tobias Achterberg. Constraint integer programming. 2007.

David L Applegate. *The traveling salesman problem: a computational study*, volume 17. Princeton university press, 2006.

Herminia I Calvete, Carmen Gale, and Pedro M Mateo. A new approach for solving linear bilevel problems using genetic algorithms. *European Journal of Operational Research*, 188(1):14–28, 2008.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Stephanus Daniel Handoko, Lau Hoong Chuin, Abhishek Gupta, Ong Yew Soon, Heng Chen Kim, and Tan Puay Siew. Solving multi-vehicle profitable tour problem via knowledge adoption in evolutionary bi-level programming. In *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, May 2015. doi: 10.1109/cec.2015.7257225. URL `http://dx.doi.org/10.1109/cec.2015.7257225`.

Yong Liang Goh, Zhiguang Cao, Yining Ma, Yanfei Dong, Mohammed Haroon Dupty, and Wee Sun Lee. Hierarchical neural constructive solver for real-world tsp scenarios. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 884–895, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3672053. URL `https://doi.org/10.1145/3637528.3672053`.

Yifan Guo, Zhongqiang Ren, and Chen Wang. imtsp: Solving min-max multiple traveling salesman problem with imperative learning. *arXiv preprint arXiv:2405.00285*, 2024.

Jialin Han, Jiaxiang Zhang, Haoyue Guo, and Ning Zhang. Optimizing location-routing and demand allocation in the household waste collection system using a branch-and-price algorithm. *European Journal of Operational Research*, 316(3):958–975, 2024.

Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12:966–980, 2017.

Qingchun Hou, Jingwei Yang, Yiqiang Su, Xiaoqing Wang, and Yuming Deng. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time. In *The Eleventh International Conference on Learning Representations*, 2023.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Jason Zhi Liang and Risto Miikkulainen. Evolutionary bilevel optimization for complex control tasks. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO ' 15. ACM, July 2015. doi: 10.1145/2739480.2754732. URL `http://dx.doi.org/10.1145/2739480.2754732`.

R Mathieu, L Pittard, and G Anandalingam. Genetic algorithm based approach to bi-level linear programming. *RAIRO-Operations Research*, 28(1):1–21, 1994.

Michele D Simoni and Matthias Winkenbach. Crowdsourced on-demand food delivery: An order batching and assignment algorithm. *Transportation Research Part C: Emerging Technologies*, 149:104055, 2023.

Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research*, 257(2):395–411, March 2017. ISSN 0377-2217. doi: 10.1016/j.ejor.2016.08.027. URL `http://dx.doi.org/10.1016/j.ejor.2016.08.027`.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Runzhong Wang, Yunhao Zhang, Ziao Guo, Tianyi Chen, Xiaokang Yang, and Junchi Yan. Linsatnet: the positive linear satisfiability neural networks. In *International Conference on Machine Learning*, pp. 36605–36625. PMLR, 2023.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

Yafeng Yin. Genetic-algorithms-based approach for bilevel programming models. *Journal of transportation engineering*, 126(2):115–120, 2000.

Akihiro Yoshida, Haruki Sato, Shiori Uchiumi, Nariaki Tateiwa, Daisuke Kataoka, Akira Tanaka, Nozomi Hata, Yousuke Yatsushiro, Ayano Ide, Hiroki Ishikura, et al. Comprehensive and practical optimal delivery planning system for replacing liquefied petroleum gas cylinders. *Japan Journal of Industrial and Applied Mathematics*, pp. 1–41, 2024.

A Postprocessing in the case study

This section details the postprocessing for the feature matrix in Section 4. As mentioned in Section 4, the discrete solution after random sampling from feature matrix $(P_{i,k})_{i,k}$ can violate the linear constraints even the original feature matrix $(P_{i,k})_{i,k}$ satisfies the constraints (i.e., $\sum_k P_{i,k} = 1$ for all i and $\sum_i P_{i,k} \leq Q$ for all k). To obtain a feasible solution, we prepare the following two steps as the postprocessing.

The first step is to determine the number of locations visited per vehicle. We estimate the demand of vehicle k by summing up each row of feature matrix $P_{i,k}$. Based on the information, we formulate the problem of obtaining the optimal number of locations per vehicle as the minimization of the total penalty, where the penalty of each vehicle is defined as the gap between the estimated demand and the optimal number of locations per vehicle. The postprocessing problem is formulated as follows;

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \sum_{k=1}^{K} \left\| v_k - \sum_{i=2}^{n} P_{i,k} \right\|_1 \\ \text{subject to} \quad & \sum_{k=1}^{K} v_k = n-1, v_k \leq Q \\ & v_k \in \mathbb{Z} \end{aligned}$$

- v_k : an integer variable that indicates the number of locations assigned to truck

This problem is sufficiently small so that it can be optimized in a short time. By solving the above optimization problem, we can use v_k as the number of locations visited by one vehicle. After fixing v_k, the upper-level constraints $\sum_k x_i^k = 1$ $(\forall i)$, $\sum_i x_i^k \leq Q$ $(\forall k)$ can be rewritten as $\sum_k x_i^k = 1$ $(\forall i)$, $\sum_i x_i^k = v_k$ $(\forall k)$. Since the vanilla Sinkhorn algorithm can handle these constraints, we use it to obtain the discrete solution.

B Feasibility check from feature matrix

This section describes that how much the sampling strategy can obtain the feasible solution experimentally discussed in Section 4. At the test phase, we first obtain the feature matrix by utilizing the trained cost estimation network and the optimal transport module. We sampled each problem and terminated with failure if the capacity constraint was satisfied or the maximum number of iterations was reached. We set the maximum number of iterations as 10,000. Table 4 shows the number of problems that succeeded in sampling the feasible solution and the average duration for sampling and checking feasibility per problem. Although we could obtain feasible solutions for almost all of the test problems by sampling, there are several problems for which we could not find a feasible solution. This is why we prepare the postprocessing detailed in Appendix A. Note that this sampling process and the postprocessing can be executed sufficiently quickly.

Problem Size	Munich		Paris		Barcelona	
	success	computation time [s]	success	computation time [s]	success	computation time [s]
100	100	0.0001	100	0.0001	100	0.0001
200	100	0.0004	100	0.0003	100	0.0001
300	93	0.0022	99	0.0005	100	0.0001
400	92	0.0028	97	0.0010	100	0.0001
500	86	0.0041	97	0.0011	100	0.0001
600	85	0.0045	97	0.0013	100	0.0001
700	82	0.0055	96	0.0013	100	0.0001
800	81	0.0057	96	0.0014	100	0.0001
900	81	0.0060	96	0.0014	100	0.0002

Table 4: The ratio to find the feasible solutions at most 10,000 trials for 100 test problems