Algorithm Selection for Traveling Salesman Problem with Simplified PointNet++

Ya Song, Laurens Bliek, and Yingqian Zhang

Eindhoven University of Technology, 5600 MB Eindhoven, Netherlands
{y.song,l.bliek,yqzhang}@tue.nl

Abstract. Selecting algorithms for the Euclidean Traveling Salesman Problem (TSP) is a well-studied topic. In this line of research, researchers investigate how to construct useful features representing TSP instances and then apply feature-based machine learning models to predict which algorithm works best with the given instance. In recent years, Convolutional Neural Network (CNN) has become a popular approach to select algorithms for TSP. Compared to traditional feature-based machine learning models, CNN has an automatic feature learning ability and demands less domain expertise. However, it is still required to generate intermediate representations, i.e., multiple images to represent TSP instances first. In this study, we propose a novel Graph Neural Network (GNN) called Simplified PointNet++ to select algorithms for TSP. This model handles the TSP instance as a point cloud, which simply takes the coordinates of cities as input, and no intermediate representations such as features or images need to be designed and generated. By hierarchically aggregating information from the neighborhood of points, the proposed model can effectively capture local features with multiple scales. We evaluate this model on two benchmark datasets, and the results show that it can outperform traditional feature-based machine learning methods and is comparable to CNN.

Keywords: Algorithm Selection · Traveling Salesperson Problem · Graph Neural Network · Point Cloud Classification · PointNet++.

1 Introduction

The Euclidean Traveling Salesman Problem (TSP) is one of the most intensely studied NP-hard combinatorial optimization problems. It relates to many real-world applications and has significant theoretical value. TSP can be described as follows. Given a list of cities with known positions, find the shortest route to visit each city and return to the origin city. Researchers have developed various exact, heuristic, and learning-based algorithms to solve this routing problem [1]. As these algorithms' performance is highly variable depending on the characteristics of the problem instances, selecting algorithms for each instance helps to improve the overall efficiency [2]. In [3], the authors firstly studied this algorithm selection problem and proposed an Automatic Algorithm Selection framework.

The framework has been further developed in [4, 5]. This framework interprets the algorithm selection as a classification problem, and we need to identify the mapping from Problem Space to Algorithm Space [6]. Traditionally, domain experts design a group of features [7–9] that can represent the characteristics of TSP instances. Then, we can train a machine learning classifier such as Support Vector Machine (SVM) to be the selector using these features. This feature-based method has several limitations: high requirement for domain knowledge, insufficient expressiveness of the features, and required feature selection process [10].

Deep learning models, especially Convolutional Neural Networks (CNN), have recently been applied to select TSP algorithms. By employing images to represent TSP instances, the algorithm selection problem is transformed into a computer vision challenge. Since CNN has sufficient automatic feature learning capability, this approach no longer requires man-designed features and may perform better. In [10], the authors generate three images: a point image, a Minimum Spanning Tree (MST) image, and a K-Nearest-Neighbor (KNN) image to represent each TSP instance. Then they apply an 8-Layer CNN architecture to predict which algorithm is better. In [11], researchers use a gridding method to transform TSP instances into density maps, and then apply Residual Networks (ResNet) [12] to do the classification. In [2], a similar instance transformation approach is used to generate images, and then a 3-Layer CNN model is designed to predict algorithms' temporal performance at different time steps.

Although the experimental results in [2, 10, 11] show CNN can outperform traditional feature-based machine learning models in the algorithm selection task for TSP, this approach still has four main drawbacks:

1. **Require to generate Intermediate representations**: Similar to feature-based methods, the instances' intermediate representations, in this case, the images, need to be generated as the inputs of CNN. It is usually a tedious process to transform TSP instances into images. In [10], generating MST and KNN images for each instance requires time-consuming calculations. When applying the gridding method to obtain images, the authors perform several up-scaling operations to improve the resolution [11]. Besides, data augmentation techniques, like random rotation/flipping, are widely used to enhance CNN's generalization ability [2, 10, 11]. As a result, multiple images must be generated to represent one TSP instance.
2. **Introduce problem-irrelevant parameters**: When generating images, the first parameter we need to set is the image size or the number of grids [11]. In [10], authors use solid dots to represent cities and solid lines to connect cities in MST and KNN images. The dot size and line width are irrelevant to the properties of the TSP instance. Adding these parameters increases the input data's complexity and the effort required for parameter tuning.
3. **Potentially lose problem-relevant information**: In the image generation procedure, the TSP instance is divided into multiple grids, with the value for each grid representing the number of cities that fall into it [2, 11]. After gridding, portions of the instance's local structure will be lost. In ad-

dition, [2] sets a maximum number for the value of grids, leading to more information distortion.

4. **Hard to generalize to other routing problems**: Researchers can apply gridding methods to convert TSP instances to images since cities are in 2D Euclidean space. Consider selecting algorithms for TSP variants, such as Asymmetric Traveling Salesman Problem (ATSP) and Capacitated Vehicle Routing Problem (VRP). In these cases, generating images to represent instances will be challenging, and a graph with assigned node/edge features may be a better representation form.

To remedy the above issues, in this work, we propose a Graph Neural Network (GNN) named Simplified PointNet++ to select TSP algorithms. The main novelties of this work are as follows:

- We are the first to regard TSP instances as point clouds and successfully apply GNN in the TSP algorithm selection field.
- The proposed model merely takes the coordinates of cities as inputs, and there is no need to design and generate intermediate representations for TSP instances.
- The adopted point cloud representation methodology has few parameter settings and can retain complete information about the original TSP instance.
- The proposed model can capture local features with multiple scales by hierarchically aggregating information from the neighborhood points. Its robust performance is demonstrated on two public TSP datasets.
- We show that the proposed model can easily generalize to other routing problems by adding node features or modifying distance metrics.

The rest of this paper is organized as follows: Section 2 introduces the research background and related works. Section 3 presents the proposed Simplified PointNet++. Section 4 shows the proposed model's experimental results on two public datasets. At last, Section 5 gives conclusions and our future work plans.

2 Related Work

Algorithm selection for optimization problems No free lunch (NFL) theorem states that no algorithm can outperform others on all optimization problems. Researchers propose and investigate algorithm selection problems to improve overall solving performance [13]. Most researchers focus on designing features for problem instances and solving the algorithm selection problem by traditional feature-based machine learning models. The collection of features for classical optimization problems like Satisfiability Problem [14], AI planning [15], Knapsack Problem [16], TSP [7–9], and VRP [17, 18] have been well designed. These features are restricted to specific problems and usually need large efforts to generate.

Deep learning has been shown to perform various classification/regression tasks effectively. If we apply deep learning models to capture instance features for algorithm selection automatically, the tedious feature design work can be eliminated. In addition to the algorithm selection models using CNN for TSP mentioned above [2, 10, 11], researchers have proposed a few feature-free algorithm selection models for other optimization problems. By generating images from the text documents for SAT problem instances, CNN can be applied to selecting algorithms [19]. In [20], researchers sample landscape information from instances and transform it into images, then apply CNN to select algorithms for Black-Box Optimization Benchmarking (BBOB) function instances. In [21], researchers treat online 1D Bin-Packing Problem instances as sequence data and apply Long Short-Term Memory (LSTM) to predict heuristic algorithms' performance. In the feature-free algorithm selection field, instances are usually converted to images or sequences, and graph representations are seldom used.

GNN for TSP The TSP instance can be naturally expressed by a graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is a group of cities, and $E = \{\langle v_i, v_j\rangle : v_i, v_j \in V\}$ is a set of routes between cities. Applying GNN in TSP-related problems to learn graph embedding is a reasonable choice.

GNN for TSP solving: GNN has been successfully applied in learning-based TSP algorithms, either in the manner of reinforcement learning or supervised learning [22]. In reinforcement learning methods, researchers use graph embedding networks such as *structure2vec* [23] and Graph Pointer Networks (GPN) [24] to represent the current policy and apply Deep Q-Learning (DQN) to update it. To tackle larger graphs, [25] introduces a two-stage learning procedure that firstly trains a Graph Convolutional Network (GCN) [26] to predict node qualities and prune some of them before taking the next action. In supervised learning methods, GNN models are commonly used as the Encoder tool [27, 28] in the upgraded version of Pointer Network [29], a sequence-to-sequence architecture. Additionally, some hybrid models combine GNN with heuristic algorithms to improve searching efficiency. [1] applies GCN to predict the probabilities of edges occurring on the optima tour and uses Beam Searching to obtain a feasible solution. [30] utilizes Graph Attention Network (GAT) [31] to estimate the regret of including a certain edge in the solution, and then Guided Local Search (GLS) explores solutions according to these predictions.

GNN for TSP searching space reduction: Searching space reduction for TSP instances is another GNN-related research task, and it can be viewed as an edge classification problem. Suppose a learned model can predict which edges in the TSP instance graph are likely to be included in the optimal solution. In that case, we can reduce the searching space and improve computational efficiency in the following searching procedure [32]. In [33], authors have designed a benchmark TSP dataset for edge classification. Here a TSP instance is represented as a K-Nearest Neighbor graph, where node features are node coordinates and edge features are Euclidean distances between two nodes. The authors test several classical GNN models and find Residual Gated Graph Convnets [34] outperform others. Many researchers use this benchmark dataset to assess the proposed GNN architectures [35–37].

GNN for TSP algorithm selection: To the best of our knowledge, no GNN models have been applied in TSP algorithm selection. However, representation learning on graphs is extensively applied in solving TSP and reducing search space for TSP, as was described above. Researchers have investigated utilizing both CNN and GCN to select TSP algorithms and discovered that CNN performs better [11]. The authors analyze the drawbacks of GCN, including the lack of relevant node features, the over-smoothing problem [38], and high time complexity. Since GNN models have various architectures, it is possible to solve the TSP algorithm selection problem by adopting a suitable GNN that considers the nature of instances, which is the purpose of our work.

Point Clouds Classification The point cloud is a type of practical 3D geometric data. Identifying point clouds has attracted much attention in recent years. It is a crucial object recognition task with multiple real-world applications, such as remote sensing, autonomous driving, and robotics [39, 40]. Unlike image data made up of regular grids, the point cloud is unstructured data as the distance between neighboring points is not fixed. As a result, we can not directly apply the classic convolutional operations on point clouds to aggregate local information. Converting point clouds to fixed-size voxel grids is the traditional way to address this issue [41, 42]. Another strategy is to project point clouds into a group of 2D images from multiple angles and then apply 2D CNN algorithms to the resulting images [43–45]. Due to the manipulation of the original point cloud data and the inefficiency of the new representation, these two approaches have potential drawbacks [40].

PointNet [46] is the first deep learning model that can directly consume raw point cloud data for 3D object classification and part/scene semantic segmentation. The authors emphasize that point clouds have the following three main properties. Firstly, unlike pixel arrays in an image, a point cloud is a set of unordered points whose permutation is not fixed. Secondly, the classification results will remain unaltered if a point cloud experiences specific transformations, such as rotation and translation. At last, points are not isolated from one another, and neighboring points constitute a meaningful subset [46]. PointNet is designed to take these three properties into account. Specifically, PointNet uses a shared Multi-Layer Perceptron (MLP) to map each point to higher-dimensional space, and a symmetric operator, i.e., Maxpooling, is applied to aggregate the global feature vector of all the points. Additionally, a geometric transformer called T-Net provides pose normalization and learn transformation invariance. The fundamental disadvantage of PointNet is the absence of local context acquisition [47]. Thus a type of GNN, PointNet++, is proposed to achieve hierarchical local feature learning by recursively applying PointNet to neighborhood points [47].

The TSP instance comprises an array of city coordinates and can be considered as 2D point cloud data. The TSP instance has the following three properties, similar to the point cloud:

- Permutation Invariance: cities in the TSP instance are unordered. The learning model has to be invariant to the permutations of the cities.
- Transformation Invariance: the characteristics or hardness of the TSP instance will be unchanged if we rotate/translate cities' positions. The learned representation of the cities should be invariant to these transformations.
- Point Interactions: the cities are in the Euclidean distance metric space. It is meaningful to consider the interactions among neighboring cities. Thus the learning model should be able to represent local structures.

In conclusion, handling TSP instances as point clouds may be preferable rather than converting them to images. And it is reasonable to apply an architecture that conforms to the specific properties of the TSP instance.

3 Simplified PointNet++ for TSP algorithm selection

Problem Statement The TSP algorithm selection problem can be defined as follows: given a TSP instance set $I = \{I_1, I_2, ..., I_l\}$, a TSP algorithm set $A = \{A_1, A_2, ..., A_m\}$, and a certain algorithm performance metric, the goal is to identify a per-instance mapping from I to A that maximizes its overall performance on I based on the given metric. As discussed in previous sections, the TSP instances can be represented by features or images, and we can apply a supervised learning model such as SVM or CNN to learn this mapping.

In this work, we treat a TSP instance I_i as a 2D point cloud $\{\mathbf{p}_j|, j = 1, 2, ..., n\}$, where each point $\mathbf{p}_j$ is a vector of its (x_j, y_j) coordinate. Thus selecting TSP algorithms can be viewed as a point cloud classification/regression task. We apply a GNN model called Simplified PointNet++, which can directly take TSP instances as inputs to accomplish this task. Next, we will describe the architecture of this model in detail.

Simplified PointNet++ PointNet++ iteratively processes 3D point clouds utilizing a scheme comprised of three fundamental layers: sampling, grouping, and neighborhood aggregation [47]. The original PointNet++ can be simplified as the inputs in our task are 2D point clouds. We eliminate the sampling layer to retain more instance information. Thus our PointNet++ scheme, as shown in Fig. 1, has two main layers:

- Grouping Layer: creates a local graph by connecting neighboring points. The original PointNet++ chooses the ball query method to define the neighborhood for the query point. Here we employ the KNN method, which is often applied in creating TSP instance graphs [33].

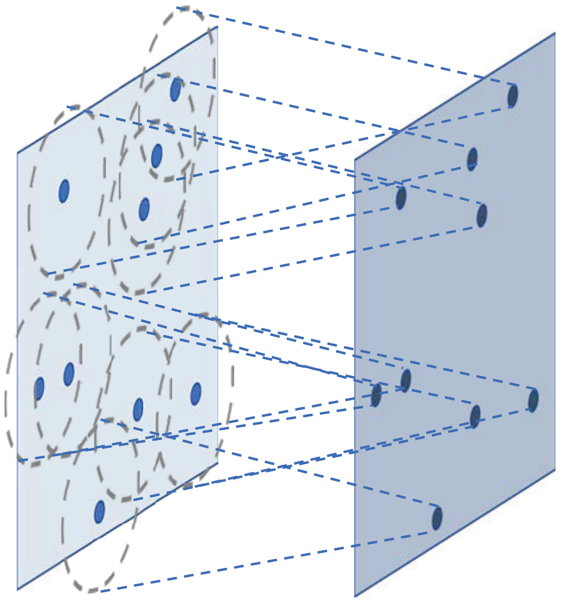

Fig. 1. Illustration of PointNet++ scheme with the neighborhood defined by the ball query method.

- Neighborhood Aggregation Layer: aggregates local features from the constructed neighborhood for each point. The neighborhood aggregation and message-passing formulation is shown as follows:

$$\mathbf{h}_i^{(\ell+1)} = \max_{j \in \mathcal{N}(i)} \mathrm{MLP}\left(\mathbf{h}_j^{(\ell)}, \mathbf{p}_j - \mathbf{p}_i\right) \tag{1}$$

  where $\mathbf{h}_i^{(\ell)}$ denotes the hidden features of point i in layer ℓ, $\mathcal{N}(i)$ denotes the constructed neighborhood for point i, and $\mathbf{p}_i$ denotes the coordinate of point i.

The Simplified PointNet++ neural architecture is shown in Fig. 2, where we stack two PointNet++ schemes sequentially to capture local context at different scales. The input of the first PointNet++ scheme is an $N \times 2$ matrix, where N is the number of cities in the TSP instance. The output matrix size of the Grouping Layer is $N \times K \times 2$, and K is the number of neighbors in the KNN method we set. The Neighborhood Aggregation Layer outputs a $N \times C$ matrix, where C is the dimension of the feature representing the local context. Then a global Max-pooling is applied across the node dimension to obtain a $1 \times C$ array. At last, we have a linear layer to map the learned features to the class label. In our algorithm selection problem, the classes are different algorithms. Both two PointNet++ schemes contain a group normalization layer, and a dropout layer is added to improve regularization before the last linear layer.

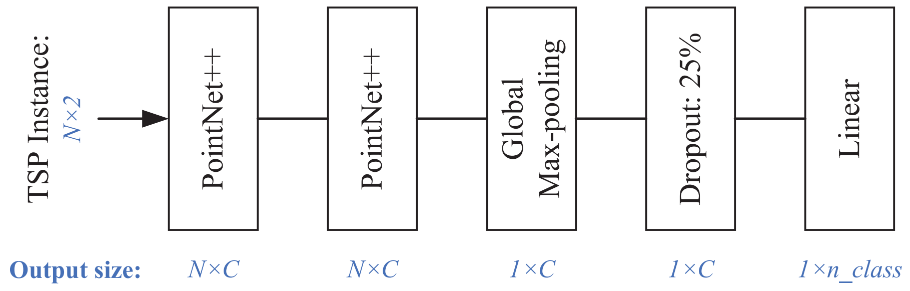

Fig. 2. The Simplified PointNet++ neural architecture

4 Experiments

4.1 Dataset

We evaluate the proposed Simplified PointNet++ on two public TSP algorithm selection datasets. The first dataset is generated to assess the Instance Space Analysis (ISA) framework [4], and the second is for evaluating the proposed CNN-based selector [10]. The main difference between the two datasets is the size of the instances. The TSP instances in the first dataset all contain 100 cities, while instances in the second dataset contain 1000 cities. Applying the proposed model to two different datasets helps us examine its adaptability and compare it with other models. The following part is a detailed description of the two datasets.

TSP-ISA dataset : includes 950 TSP instances with 100 cities, and it is divided equally into seven groups based on instance characteristics: RANDOM, CLKeasy, CLKhard, LKCCeasy, LKCChard, easyCLK-hardKLCC and hardCLK-easyLKCC. Here Chained Lin-Kernighan (CLK) and Lin-Kernighan with Cluster Compensation (KLCC) are famous heuristic algorithms for solving TSP. The aim is to predict whether CLK or KLCC is better for each instance, and we can view it as a binary classification task. According to the mean effort of the two algorithms, KLCC is the Single-Best-Solver. As the dataset is not balanced, choosing KLCC for all instances can achieve 80% accuracy. [4] designs 11 features to represent TSP instances and trains a multilayered feedforward neural network to be the selector.

TSP-CNN dataset : includes 1000 TSP instances with 1000 cities. There are two algorithms to choose from Edge-Assembly-Crossover (EAX) and Lin-Kernigham Heuristic (LKH), and the PAR10 score serves as the performance matrix. It is a well-balanced dataset, and choosing Single-Best-Solver (EAX) for all instances can only achieve 49% accuracy [10].

4.2 Baseline model

In addition to comparing with the model proposed in other articles, we create a baseline model, i.e., a GCN-based TSP selector. We can recognize which representation form and corresponding learning model performs better by comparing GCN with Simplified PointNet++.

To apply GCN to select algorithms for TSP, firstly, we create a TSP graph dataset. We keep in line with the graph setting in [33] for TSP edge classification and use KNN graphs to represent TSP instances, here $K = 0.2 \times N$, N is the number of cities in the TSP instances. We set the node feature to be node coordinates, and it is a $N \times 2$ matrix. Let the edge feature be the distance between two linked cities, and the matrix size is $[0.2 \times N^2, 1]$. We apply the standard GCN model to classify the instance graphs. The GCN message-passing formulation is:

$$\mathbf{h}_i^{(\ell+1)} = \sigma \sum_{j \in \mathcal{N}(v) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{h}_j^{(\ell)} \tag{2}$$

where $e_{j,i}$ denotes the edge weight from source node j to target node i, $\hat{d}_i$ denotes the node degree of node i. We keep the neural architecture of GCN the same as Simplified PointNet++ by just replacing PointNet++ schemes with GCN layers.

4.3 Result and analysis

For a fair comparison with baseline models, we process the datasets in the same way as [4] and [10]. For the TSP-ISA dataset, we randomly split the entire dataset into training and test datasets, where 80% instances for training and the remaining 20% instances for the test. We apply exactly the same 10-fold cross-validation on the TSP-CNN dataset as the data grouping information was released in [10]. We set the KNN method parameter K to be 10, the hidden channel dimension for the GCN layer, and the Simplified PointNet++ scheme to be 32. We use an Adam optimizer with a 0.001 learning rate to reduce Cross Entropy loss and train 100 epochs for each model.

The average classification accuracy of 10 runs on the TSP-ISA dataset is listed in Table 1. We can see that the proposed Simplified PointNet++ can outperform the traditional feature-based machine learning model. As the proposed method does not require any domain knowledge of TSP and has high prediction accuracy, it can be a promising approach in this field. Same with [10], we record the best result of the trained GCN and Simplified PointNet++ on TSP-CNN dataset in Table 2. We can find CNN-based model performs slightly better than Simplified PointNet++ in terms of overall performance. We calculate the T-test for the means of two performance lists and find they do not differ significantly.

Table 3 gives a summary of properties of CNN-based method and Simplified PointNet++. Though the proposed Simplified PointNet++ does not outperform CNN in prediction accuracy, it is still a competitive method. Firstly, CNN takes multiple images as inputs, i.e., Points image, MST image, and KNN image.

Generating these images may be time-consuming, and it is unclear which image can better represent TSP instances. Contrary to CNN, Simplified PointNet++ directly takes the 2D coordinate matrix of cities as inputs, and we do not need to prepare intermediate representations like images. Secondly, when generating images for CNN, several problem-irrelevant parameters must be set, such as image size, dot size, and line width in MST and KNN images. Tuning these parameters can be a heavy workload, Although theoretically, these parameters should not affect the learned mapping from instances to algorithms. While in Simplified PointNet++, the TSP instances are treated as point clouds, and there are no parameters to be designed or adjusted. Besides, when setting the image resolution in the CNN-based method, we should consider the city number in the TSP instance. Otherwise, the representation ability of the image is inadequate, and problem instance information is lost.

At last, generating images for TSP instances and applying CNN to select algorithms is not very difficult because cities in TSP are homogeneous and in 2D Euclidean space. If we look into some complex routing problems, we will find that applying the CNN-based method is challenging. For VRP algorithm selection, it is hard to differentiate the depot and customer with image representations. While in Simplified PointNet++, we can simply add the point features to tell them apart. Considering the routing problem in Non-Euclidean space such as ATSP, drawing the problem instance on a 2D plane is nearly impossible. While Simplified PointNet++ can naturally recognize the neighborhood in ATSP, we also can modify the message-passing formulation in Simplified PointNet++ to aggregate more valuable edge features.

Table 1. Comparison of baselines with our models on the TSP-ISA dataset.

Models	Input data	Accuracy
ANN [4]	Man-designed 11 features	97.00%
GCN	KNN graphs	96.37%
Simplified PointNet++	Point clouds	**98.42%**

Table 2. Comparison of baselines with our best models on the TSP-CNN dataset and cross all ten folds.

Fold	Accuracy		
	CNN [10]	GCN	Simplified PointNet++
1	0.78	0.68	0.73
2	0.67	0.63	0.65
3	0.80	0.74	0.74
4	0.76	0.68	0.68
5	0.57	0.60	0.68
6	0.71	0.59	0.63
7	0.70	0.67	0.72
8	0.77	0.69	0.69
9	0.58	0.63	0.62
10	0.71	0.65	0.76
Average	**70.50%**	65.60%	69.00%

Table 3. Properties comparison between CNN-based method and Simplified PointNet++.

Properties	CNN-based method [10]	Simplified PointNet++
Intermediate representations preparation	Points image MST image KNN image	None
Problem-irrelevant parameters	Image size Dot size Line width	None
Data Augmentation	Random rotation Random flipping	None
Problem-relevant information loss	Affected by resolution	None
Distinguish different points	Hard	Easy to add node features
To Non-Euclidean Metric Space	Hard	Easy [47]
Accuracy	70.50%	69.00%

5 Conclusion

In this work, we proposed a novel GNN, i.e., Simplified PointNet++, to select algorithms for TSP. This model handles TSP instances as 2D point clouds and only takes cities' coordinates as inputs. Thus no intermediate representations for problem instances, such as features or images, need to be designed and generated before model training. In contrast to converting TSP instances to images, the point cloud representation is more natural and elegant. It neither introduces problem-irrelevant parameters nor loses problem-relevant information. The proposed method is promising as it is easy to generalize to other routing problems.

For example, we can distinguish points in the problem instances by adding node features. Furthermore, we can modify the model to learn the representation for problem instances in Non-Euclidean space. We think this work can be a powerful starting point for selecting algorithms for combinatorial optimization problems defined on graphs. In the future, we will apply similar GNN architectures to more complex routing problems like ASTP and VRP.

Acknowledgements Ya Song would like to thank the China Scholarship Council (CSC) for the financial support of this research.

References

1. Joshi, C., Laurent, T. & Bresson, X. An efficient graph convolutional network technique for the travelling salesman problem. *ArXiv Preprint ArXiv:1906.01227.* (2019)
2. Huerta, I., Neira, D., Ortega, D., Varas, V., Godoy, J. & Asın-Achá, R. Improving the state-of-the-art in the Traveling Salesman Problem: An Anytime Automatic Algorithm Selection. *Expert Systems With Applications.* **187** pp. 115948 (2022)
3. Rice, J. The algorithm selection problem. *Advances In Computers.* **15** pp. 65-118 (1976)
4. Smith-Miles, K. & Hemert, J. Discovering the suitability of optimisation algorithms by learning from evolved instances. *Annals Of Mathematics And Artificial Intelligence.* **61**, 87-104 (2011)
5. Smith-Miles, K., Baatar, D., Wreford, B. & Lewis, R. Towards objective measures of algorithm performance across instance space. *Computers Operations Research.* **45** pp. 12-24 (2014)
6. Kotthoff, L. Algorithm selection for combinatorial search problems: A survey. *Data Mining And Constraint Programming.* pp. 149-190 (2016)
7. Hutter, F., Xu, L., Hoos, H. & Leyton-Brown, K. Algorithm runtime prediction: Methods evaluation. *Artificial Intelligence.* **206** pp. 79-111 (2014)
8. Bossek, J. Salesperson: computation of instance features and R interface to the state-of-the-art exact and inexact algorithms for the traveling salesperson problem (2017).
9. Pihera, J. & Musliu, N. Application of machine learning to algorithm selection for TSP. *2014 IEEE 26th International Conference On Tools With Artificial Intelligence.* pp. 47-54 (2014)
10. Seiler, M., Pohl, J., Bossek, J., Kerschke, P. & Trautmann, H. Deep learning as a competitive feature-free approach for automated algorithm selection on the traveling salesperson problem. *International Conference On Parallel Problem Solving From Nature.* pp. 48-64 (2020)
11. Zhao, K., Liu, S., Yu, J. & Rong, Y. Towards Feature-free TSP Solver Selection: A Deep Learning Approach. *2021 International Joint Conference On Neural Networks (IJCNN).* pp. 1-8 (2021)
12. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition.* pp. 770-778 (2016)
13. Kerschke, P., Hoos, H., Neumann, F. & Trautmann, H. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation.* **27**, 3-45 (2019)

14. Hoos, H., Lindauer, M. & Schaub, T. claspfolio 2: Advances in algorithm selection for answer set programming. *Theory And Practice Of Logic Programming.* **14**, 569-585 (2014)
15. Fawcett, C., Vallati, M., Hutter, F., Hoffmann, J., Hoos, H. & Leyton-Brown, K. Improved features for runtime prediction of domain-independent planners. *Twenty-Fourth International Conference On Automated Planning And Scheduling.* (2014)
16. Huerta, I., Neira, D., Ortega, D., Varas, V., Godoy, J. & Asın-Achá, R. Anytime automatic algorithm selection for knapsack. *Expert Systems With Applications.* **158** pp. 113613 (2020)
17. Mayer, T., Uhlig, T. & Rose, O. Simulation-based autonomous algorithm selection for dynamic vehicle routing problems with the help of supervised learning methods. *2018 Winter Simulation Conference (WSC).* pp. 3001-3012 (2018)
18. Rasku, J. Toward Automatic Customization of Vehicle Routing Systems. *JYU Dissertations.* (2019)
19. Loreggia, A., Malitsky, Y., Samulowitz, H. & Saraswat, V. Deep learning for algorithm portfolios. *Thirtieth AAAI Conference On Artificial Intelligence.* (2016)
20. He, Y. & Yuen, S. Black box algorithm selection by convolutional neural network. *International Conference On Machine Learning, Optimization, And Data Science.* pp. 264-280 (2020)
21. Alissa, M., Sim, K. & Hart, E. Automated Algorithm Selection: from Feature-Based to Feature-Free Approaches. *ArXiv Preprint ArXiv:2203.13392.* (2022)
22. Vesselinova, N., Steinert, R., Perez-Ramirez, D. & Boman, M. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access.* **8** pp. 120388-120416 (2020)
23. Khalil, E., Dai, H., Zhang, Y., Dilkina, B. & Song, L. Learning combinatorial optimization algorithms over graphs. *Advances In Neural Information Processing Systems.* **30** (2017)
24. Ma, Q., Ge, S., He, D., Thaker, D. & Drori, I. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. *ArXiv Preprint ArXiv:1911.04936.* (2019)
25. Manchanda, S., Mittal, A., Dhawan, A., Medya, S., Ranu, S. & Singh, A. Learning heuristics over large graphs via deep reinforcement learning. *ArXiv Preprint ArXiv:1903.03332.* (2019)
26. Kipf, T. & Welling, M. Semi-supervised classification with graph convolutional networks. *ArXiv Preprint ArXiv:1609.02907.* (2016)
27. Nowak, A. & Bruna, J. Divide and conquer networks. *ArXiv Preprint ArXiv:1611.02401.* (2016)
28. Sultana, N., Chan, J., Sarwar, T. & Qin, A. Learning to optimise general tsp instances. *International Journal Of Machine Learning And Cybernetics.* pp. 1-16 (2022)
29. Vinyals, O., Fortunato, M. & Jaitly, N. Pointer networks. *Advances In Neural Information Processing Systems.* **28** (2015)
30. Hudson, B., Li, Q., Malencia, M. & Prorok, A. Graph Neural Network Guided Local Search for the Traveling Salesperson Problem. *ArXiv Preprint ArXiv:2110.05291.* (2021)
31. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. & Bengio, Y. Graph attention networks. *ArXiv Preprint ArXiv:1710.10903.* (2017)
32. Fitzpatrick, J., Ajwani, D. & Carroll, P. Learning to sparsify travelling salesman problem instances. *International Conference On Integration Of Constraint Programming, Artificial Intelligence, And Operations Research.* pp. 410-426 (2021)

33. Dwivedi, V., Joshi, C., Laurent, T., Bengio, Y. & Bresson, X. Benchmarking graph neural networks. *ArXiv Preprint ArXiv:2003.00982*. (2020)
34. Bresson, X. & Laurent, T. Residual gated graph convnets. *ArXiv Preprint ArXiv:1711.07553*. (2017)
35. Zhang, H., Xu, M., Zhang, G. & Niwa, K. SSFG: Stochastically scaling features and gradients for regularizing graph convolutional networks. *IEEE Transactions On Neural Networks And Learning Systems*. (2022)
36. Chen, Y., Tang, X., Qi, X., Li, C. & Xiao, R. Learning graph normalization for graph neural networks. *Neurocomputing*. **493** pp. 613-625 (2022)
37. Chen, C., Tao, C. & Wong, N. Litegt: Efficient and lightweight graph transformers. *Proceedings Of The 30th ACM International Conference On Information Knowledge Management*. pp. 161-170 (2021)
38. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J. & Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proceedings Of The AAAI Conference On Artificial Intelligence*. **34**, 3438-3445 (2020)
39. Lin, C., Kong, C. & Lucey, S. Learning efficient point cloud generation for dense 3d object reconstruction. *Proceedings Of The AAAI Conference On Artificial Intelligence*. **32** (2018)
40. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. & Bennamoun, M. Deep learning for 3d point clouds: A survey. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **43**, 4338-4364 (2020)
41. Ghadai, S., Lee, X., Balu, A., Sarkar, S. & Krishnamurthy, A. Multi-resolution 3D convolutional neural networks for object recognition. *ArXiv Preprint ArXiv:1805.12254*. **4** (2018)
42. Wang, C., Cheng, M., Sohel, F., Bennamoun, M. & Li, J. NormalNet: A voxel-based CNN for 3D object classification and retrieval. *Neurocomputing*. **323** pp. 139-147 (2019)
43. Su, H., Maji, S., Kalogerakis, E. & Learned-Miller, E. Multi-view convolutional neural networks for 3d shape recognition. *Proceedings Of The IEEE International Conference On Computer Vision*. pp. 945-953 (2015)
44. Bai, S., Bai, X., Zhou, Z., Zhang, Z. & Jan Latecki, L. Gift: A real-time and scalable 3d shape search engine. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 5023-5032 (2016)
45. Kalogerakis, E., Averkiou, M., Maji, S. & Chaudhuri, S. 3D shape segmentation with projective convolutional networks. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 3779-3788 (2017)
46. Qi, C., Su, H., Mo, K. & Guibas, L. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 652-660 (2017)
47. Qi, C., Yi, L., Su, H. & Guibas, L. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances In Neural Information Processing Systems*. **30** (2017)