BiGNN: Bipartite Graph Neural Network with Attention Mechanism for Solving Multiple Traveling Salesman Problems in Urban Logistics

Haojian Liang\textsuperscript{a} Shaohua Wang\textsuperscript{b,c,*} Huilai Li\textsuperscript{a,d} Liang Zhou\textsuperscript{e} Xueyan Zhang\textsuperscript{f} and Shaowen Wang\textsuperscript{g,*}

\textsuperscript{a}School of Artificial Intelligence, Jilin University, Changchun 130012, China; \textsuperscript{b}Key Laboratory of Remote Sensing and Digital Earth Chinese Academy of Sciences, Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China; \textsuperscript{c}International Research Center of Big Data for Sustainable Development Goals, Beijing 100094, China; \textsuperscript{d}School of Mathematics, Jilin University, Changchun 130012, China; \textsuperscript{e}Faculty of Geomatics, Lanzhou Jiaotong University, Lanzhou 730070, China; \textsuperscript{f}Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90089, USA; \textsuperscript{g}CyberGIS Center for Advanced Digital and Spatial Studies, University of Illinois at Urbana-Champaign, USA

*Corresponding author: Shaowen Wang\textsuperscript{a}, Shaohua Wang\textsuperscript{b}

Email: \textsuperscript{a}shaowen@illinois.edu \textsuperscript{b}wangshaohua@aircas.ac.cn

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4716309
BiGNN: Bipartite Graph Neural Network with Attention Mechanism for Solving Multiple Traveling Salesman Problems in Urban Logistics

Abstract

The multiple traveling salesman problems (MTSP), such as vehicle routing and production process optimization, which arise from real-world \(NP\)-hard problems, are essential in urban logistics. The MinMax-MTSP is an important variation of MTSP that distributes the workload equally among different salesmen equally and reduces the overall labor force that salesmen travel through a city to the minimum. Bounded-MTSP is also a variation of MTSP, which imposes constraints on the minimum and maximum number of cites salesmen are required to visit. Branch-and-bound is an exact algorithm to solving combinatorial optimization problems. Learn to Branch (L2B) uses learning to guide branch node selection. In this study, we take imitate learning techniques to assist branch-and-bound in space exploration solutions, which is a combination of an attention mechanism and mathematical modeling with Bipartite Graph Neural Network (BiGNN). The problems are framed to formulate mixed integer linear programming, which is different from conventional algorithms. It is proposed that a bipartite graph network approach makes a feature representation by setting a structure of constraints and variables. Experimental results showed that our model can generate more accurate solutions than three benchmark models. The BiGNN model can effectively learn the strong branch strategy, which reduces solution time by replacing complex calculations with fast approximations. Additionally, the small-scale instances model can be applied to larger-scale ones.

Keywords: Bipartite graph; Graph neural network; Attention mechanism; Multiple traveling salesman problem; Branch-and-bound
1 Introduction

In recent years, deep learning has been widely applied across various domains, including medical image analysis, protein structure prediction, and urban planning. Significant progress has also been made in addressing challenges related to disaster prediction, traffic flow forecasting, and path planning (Hu et al., 2019; Janowicz et al., 2020; Gui et al., 2021; Zhang et al., 2020a, 2020b). The emergence of deep learning has introduced new perspectives for addressing various geospatial optimization problems. Therefore, it is crucial to design efficient solving algorithms based on deep learning methods to meet the growing complexity in the fields of urban planning and logistics distribution (Liu et al., 2017; Liang et al., 2023; Zhou et al., 2023; Russell et al., 2023). This holds significant research importance, aiming to develop solutions that can effectively address the evolving and intricate demands in these domains.

Traveling Salesman Problem (TSP) is a classic spatial optimization problem that aims to find the shortest path for a salesman to visit all given cities and return to the initial city, considering the distances (or costs) between cities. It has been extensively studied in fields such as operations research, geography, computer science, and artificial intelligence, and plays a significant role. Multiple Traveling Salesman Problem (MTSP) is an extension of the TSP. It involves multiple salesmen instead of just one, and requires planning of paths among cities for all of them. In the MTSP, the salesmen start from an initial city, visit all cities together, and then return to the starting city. The aim is to identify a set of routes that minimizes the total distance (or
cost) travelled by all salesmen, subject to the constraint that each city, except the starting city, must be visited exactly once. The MTSP is significant in various fields, such as logistics distribution, traffic planning, and telecommunication network maintenance. As a result, several variant forms of MTSP have emerged, including Standard-MTSP, MinMax-MTSP, and Bounded-MTSP, among others.

The objective of Standard-MTSP is to minimize the total distance or cost traveled by all salesmen, which maximizes overall efficiency but may result in an uneven workload distribution among them. MinMax-MTSP and Bounded-MTSP are variant forms introduced based on fairness principles. MinMax-MTSP optimizes the objective of minimizing the maximum distance traveled among all salesmen, ensuring that the longest distance in any salesman’s route is as short as possible, thereby enhancing overall efficiency and fairness. The Bounded-MTSP restricts the minimum and maximum number of cities salesmen are required to visit which aims to balance the workloads for the salesmen. Fig. 1. shows the comparison results of the Standard-MTSP, MinMax-MTSP and Bounded-MTSP on the same instance.

The Standard-MTSP can be extended to solve various vehicle routing problems (VRP). While there are extensive studies on TSP (Shi et al., 2020; Niendorf et al., 2015, 2017), and VRP studies have already received significant attention (Duan et al., 2021; Wang et al., 2015; Azad et al., 2017; Feng et al., 2019), there are limited investigations on MTSP (Li et al., 2015). MTSP, in its many variants, captures various real-world problems that can be modelled using mixed-integer linear programming (MILP). Each variant corresponds to different constraints.
Fig. 1. The demonstration of the Standard-MTSP, MinMax-MTSP and Bounded-MTSP

There are numerous algorithms available for solving the MILP problem. These algorithms use various optimization techniques and heuristics to efficiently explore the solution space and find optimal or near-optimal solutions. One of the exact algorithms for solving MILP is Branch-and-Bound (B&B) (Ibaraki, 1978). A modern solver, SCIP (Gamrath et al., 2020), employs the B&B approach to solving the MILP problem by designing branch strategies to build the smallest search tree. In this context, the main challenge is to select branch strategies that result in the smallest search tree. Although conventional methods are useful in evaluating the significance of candidate nodes, they can be time-consuming. Deep learning presents a promising approach to developing new strategies. Imitation learning involves training a strategy by mimicking the state and action of experts interacting in the environment, which can be considered the optimal strategy. Therefore, we opted for imitation learning to quickly approximate and replace complex calculations without creating new explicit
algorithms. The input data consists of graph-structured data, which is a non-Euclidean type of data that requires specialized distance measures to capture the underlying relationships and patterns within the graph. Our research extracts constraints and variable features from the MILP problem and converts them into a bipartite graph. Based on the B&B framework, we combined Bipartite Graph Neural Network (BiGNN) and the attention mechanism to solve the Standard-MTSP, MinMax-MTSP, and Bounded-MTSP.

Our study makes the following key contributions:

- This study uses the idea of Learn to Branch (L2B) to solve the Standard-MTSP, MinMax-MTSP and Bounded-MTSP by the imitation learning and the approach can be extended to other variant forms of MTSP if it follows the form of MILP.
- Integrating BiGNN with attention mechanism improves the performance of branching and selecting nodes.
- Our models are far better than several machine learning benchmark methods in terms of performance.

The remainder of this paper is organized as follows: First, Section 2 describes related works. Section 3 lists the preliminary knowledge, and Section 4 presents the details of our methodological framework. Section 5 illustrates the experimental designs and the results. Finally, the conclusion and the future research prospects are discussed.
2 Related work

Several heuristic algorithms have been developed to solve Standard MTSP and its variants. However, there is limited research on using deep learning to solve these problems. Recently, the L2B approach was proposed to guide branch node selection using machine learning. This section will review related work from two perspectives. First, we will briefly introduce research on Standard-MTSP, MinMax-MTSP, and Bounded-MTSP. Later, we will discuss research progress based on the L2B strategy.

2.1 Literature Review of Multiple Travelling Salesman Problem

The Standard-MTSP has significant implications for real-world applications. It is crucial to determine the allocation and sequence of visited cities. Since the mid-1990s, researchers have taken an objective and rational approach to this problem with the development of neural network. Some researchers have attempted to solve the Standard-MTSP using neural networks (Wacholder et al., 1989; Torki et al., 1997; Somhom et al., 1999). Later, several heuristic algorithms were developed, including genetic algorithms (Zhang et al., 1999), Tabu search (Ryan et al., 1998), and simulated annealing (Song et al., 2003), to solve the MTSP problem. Bektas (2006) conducted a review of research on MTSP and proposed that heuristic algorithms are important for solving such problems. Carter et al. (2006) introduced the genetic algorithm to address the Standard-MTSP. Hosseinabadi et al. (2014) proposed a novel hybrid algorithm for solving the Standard-MTSP, which can obtain optimal solutions even in high-complexity scenarios. Wang et al. (2020) combined the minimum spanning tree with Ant Colony Optimization (ACO) algorithm to solve the Standard-
MTSP with time window. Recently, several studies have applied the ACO algorithm (Lu et al., 2017) and evolutionary search (Lupoaic et al., 2019) to solve the MinMax-MTSP. He et al. (2023) proposed a unified memetic method to solving both cases of MinMax-MTSP and MinMax multidepot MTSP. Mahmoudinazlou et al. (2024) proposed a hybrid genetic algorithm for solving the MinMax-MTSP.

Nowadays, deep learning is widely used in numerous fields (Gao, 2021). There is already research using deep learning or reinforcement learning (RL) to study Combinatorial Optimization Problems (COPs). Vinyals et al. (2017) first proposed a pointer network (Ptr-Net) to tackle COPs. Ptr-Net was introduced to tackle problem where the output space is dynamic and can vary in size based on the input data. Bello et al. (2017) combined Ptr-Net with RL and used actor-critic RL algorithm to train Ptr-Net. Ma et al. (2019) used the unique combination of RL and graph embedding to solve the maximum cut and TSP. While TSP has been studied in this context, little research on the Standard-MTSP has been done. Kaempfer et al. (2019) extended the structure of Ptr-Net to problems involving multiple sets. By encoding the salesman, the depot, and the cities as three sets, a new proposal ensures that the obtained solution is suitable for the Standard-MTSP. Hu et al. (2020) proposed an innovative approach to tackle Standard-MTSP, offering a near-optimal solution with a learning-based model. Gao et al. (2023) proposed a deep reinforcement learning method for solving MinMax-MTSP.

Most studies on MTSP rely on heuristic algorithms, which cannot guarantee optimality. Deep learning methods for MTSP mostly use end-to-end solving to
provide an approximate solution. However, it is worth exploring the combination of deep models with exact algorithms based on the idea of L2B to accelerate problem-solving.

2.2 Literature Review of Learn to Branch

L2B adopts deep learning to guide the strategy of branch node selection in the B&B method. He et al. (2014) first proposed a machine learning algorithm to learn branch and node selection strategies, which can effectively improve the solution's efficiency. Lodi et al. (2017) wrote a comprehensive review to summarize the research progress on L2B. Alvarez et al. (2017), Khalil et al. (2016), and Hansknecht et al. (2018) improved three machine learning algorithms to assist L2B. Based on previous studies, Gauss et al. (2019) used graph neural network and imitation learning to learn branch and node selection for the first time and achieved state-of-the-art (SOTA) results on four COPs. Gupta et al. (2020) proposed a new hybrid architecture to handle the situation without high-end graphics processing units (GPUs). Zhang et al. (2023) provide a review of machine learning methods used to solve the MILP. These recent studies motived our research to apply the idea of L2B to solve the Standard-MTSP, MinMax-MTSP, and Bounded-MTSP as these problems are practically important, but received relatively limited attention from various research communities.
3 Preliminary

3.1 Standard-MTSP, MinMax-MTSP and Bounded-MTSP

Given a set of nodes, each of them can be regarded as a city. M salesmen depart from the same city (named depot node), travel through all the cities together, and finally back to the destination. The aim of MTSP is to minimize the total cost. According to different scenarios, the cost can be weighted by time, distance, etc. Many variants have generated from specific practical considerations based on the Standard-MTSP problem. Typically, the total distance of $m$ salesmen needs to be minimized. However, it is possible to reduce the difference between each salesman in actual problems, while keeping traveling distances among salesmen as close as possible, minimizing the total distance.

3.1.1 Standard-MTSP

Although classic TSP is widely used, it is not suitable for representing complex real-world problems. Standard-MTSP is a generalization of TSP. Standard-MTSP extends the problem scale from one salesman to $m$ salesmen, which can represent cooperative and competitive relationships. The Standard-MTSP defines a depot city as both a starting point and a destination. There exists a cooperative relationship between all salesmen to achieve the minimum total cost. Note that all cities need to be visited, and each city can only be visited once.
3.1.2 The variants of Standard-MTSP

Compared with TSP, Standard-MTSP is more suitable for representing real-world situations and applying to various path planning and time scheduling, such as print press scheduling (Gorenstein, 1970), school bus routing problem (Angel et al., 1972), crew scheduling (Huckfeldt, 1973), interview scheduling (Gilbert and Hofstra, 1992), mission planning (Brumitt and Stentz, 1996), hot rolling scheduling (Tang et al., 2000), etc. The above cases can be directly modeled as Standard-MTSP. Meanwhile, the Standard-MTSP can be used as a sub-problem to represent more general problems as follows:

- Workload balancing
- The MTSP with time window
- Dealing with all kinds of VRP

An important indicator for measuring salesmen’s satisfaction is the fairness of each workload. Okonjo-Adigwe (1988) first considered this indicator in routing problem and focused on balancing the workload of each salesman. Next, we will introduce two variants of MTSP, MinMax-MTSP and Bounded-MTSP, both of which are based on the indicator of workload balancing.

3.1.3 MinMax-MTSP

In Standard-MTSP, the optimization goal is to reduce the overall labor force. However, it is also important to balance the labor force and reduce the overall labor force in various real-world scenarios. Specifically, other than minimizing the total
distance that salesmen travel through the city, efforts should also be made to make the
distance that each salesman travel about the same. The MinMax-MTSP is framed to
consider these two optimization factors. It should make efforts to keep the distance
salesmen travel close to each other. Otherwise, the solution may waste resources or
lead to labor imbalance. However, research on MinMax-MTSP is limited.

3.1.4 Bounded-MTSP

Like MinMax-MTSP, the Bounded-MTSP also balances the workloads for the
salesmen. The difference is that the Bounded-MTSP restricts the minimum and
maximum number of cites salesmen are required to visit which aims to balance the
workloads for the salesmen. In real-world situations, we may measure workload by
the number of visited nodes rather than distance. The Bounded-MTSP can capture this
situation. Sometimes, we can only set the minimum number of nodes without the
maximum number to ensure each salesman has a minimal number of tasks.

3.2 Branch-and-bound

B&B is a classic method to solve MILP (Ibaraki, 1978). The main idea of B&B
is to decompose an NP-hard problem into linear programming (LP) problems. The
original problem is tracked between the upper and lower bounds during the solution
search process. We consider the original problem as the root node. Starting from the
root node, the significance of the branch is to break down the origin problems into
subproblems. The process of the branch is continuously adding child nodes to the tree.
The bound process is to check whether the upper and lower bounds are satisfied of the

This preprint research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstract=4716309
sub-problem during the branching. The branch should be cut if the sub-problem cannot produce a better solution than the current optimal solution until all sub-problems cannot generate an optimal solution, the algorithm ends. Through the continuous partitioning of subproblems, searching the solution space, and pruning, the branch and bound approach offers an effective method for solving origin problem.

Choosing the variables is the key step in the B&B during the branching process. Various branching strategies lead to different sizes of the search tree and would affect efficiency of the solution. In the modern optimization solvers, branch rules play a crucial role in branching process. So far, the most effective strategy is the strong branch strategy which selects branches by calculating the expected bounds of each candidate variable, resulting in each candidate variable requirements of solving two LP problems. It is challenging to run strong branches on each node in real-world applications. Modern solvers rely on mixed branching, which only uses strong branch strategy at the beginning of the solution process and gradually switches to a more straightforward heuristics process.

3.3 Mixed Integer Linear Programming

Our study adopts the B&B framework to provide a general method to solve these problems. In this part, we provide the mathematical definition of MILP, then give the MILP formulation for the Standard-MTSP, MinMax-MTSP (Bektas, 2006) and Bounded-MTSP (R. Necula et al., 2015).

Firstly, we provide the mathematical formulation of MILP:
\[
\arg\min_{x} c^{T}x. \tag{1}
\]
\[
\text{s.t. } Ax \leq b. \tag{2}
\]
\[
l \leq x \leq u. \tag{3}
\]
\[
x \in \mathbb{Z}^{p} \times \mathbb{R}^{n-p}. \tag{4}
\]

where \( c \in \mathbb{R}^{m} \) represents the coefficient of the target vector, \( A \in \mathbb{R}^{m \times n} \) is the matrix of constraint coefficients, \( b \in \mathbb{R}^{m} \) is the constraint right-hand-side vector. \( l \) and \( u \) are the lower and upper bounds respectively, \( p \leq n \) denotes the number of integer variables.

Then the idea of B&B to solve MILP is given. While the original MILP is not solvable, the problem is relaxed to LP. For such problems, the simplex method can efficiently solve. If the solution \( x^* \in \mathbb{Z}^{p} \times \mathbb{R}^{n-p} \), the optimal solution of origin MILP is \( x^* \). If \( x \notin \mathbb{Z}^{p} \times \mathbb{R}^{n-p} \), we get a lower bound of the origin problem. There exists \( i \leq p \), so that \( x^* \notin \mathbb{Z} \). Recursively split the relaxed LP by \( x^* \) and we can get
\[
x_i \leq \lfloor x^*_i \rfloor \text{ and } x_i \geq \lceil x^*_i \rceil \tag{5}
\]

The minimum of LP corresponding to all leaf nodes is taken as the lower bound \( L \), and the minimum value of LP corresponding to all integer leaf nodes is taken as the upper bound \( U \). Next, give up or create a new branch according to \( L \) and \( U \). When \( L = U \), we find the optimal solution of MILP. When \( L = \infty \), MILP has no solution. When \( L - U \leq \text{threshold} \), stop and get the approximate optimal solution of MILP.

### 3.3.1 MILP formulation for Standard-MTSP

The number 1, 2, ..., \( n \) represent the cities and define a decision variable:

\[
x_{ij} = \begin{cases} 
1, & \text{from city } i \text{ to city } j \\
0, & \text{otherwise}
\end{cases} \tag{6}
\]
$c_{ij}$ is the distance from city $i$ to city $j$. For $i = 1, 2, 3, \ldots, n$, $u_i$ is an auxiliary variable. It can be understood as the visited order of city $i$ in optimal path. For example, $u_2 = 3$ indicates that city 2 is the third visited by a salesman.

The Standard-MTSP can be formulated as a MILP as follows:

$$\min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{7}$$

s.t.

$$\sum_{j=2}^{n} x_{1j} = m \tag{8}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{9}$$

$$\sum_{i=1}^{n} x_{ij} = 1, i = 2, 3, \ldots, n \tag{10}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 2, 3, \ldots, n \tag{11}$$

$$x_{1i} + x_{ii} \leq 1, i = 2, 3, \ldots, n \tag{12}$$

$$u_i - u_j + (n - m)x_{ij} \leq n - m - 1 \tag{13}$$

$$2 \leq i \neq j \leq n$$

$$x_{ij} \in \{0, 1\}, i = 1, 2, \ldots, n; j = 1, 2, \ldots, n.$$

Constraints (8) and (9) ensure that $m$ salesmen are starting from the depot and return to the depot eventually. Constraints (10) and (11) ensure that only one salesman enters and exits each node, and they are called degree constraints. Inequality (12) requires each salesman to visit at least 2 nodes, except the depot. Constraints (13) are the classical sub tour elimination constraints proposed by Miller et al. (1960).
3.3.2 MILP formulation for MinMax-MTSP

The formulation for the MinMax-MTSP requires a third dimension \((k)\) to clearly distinguish among the arcs assigned to each salesman:

\[
\begin{align*}
\text{min} & \quad Q#(14) \\
\text{s.t.} & \quad \sum_{j=2}^{n} x_{1jk} = 1, k = 1,2,\ldots,m#(15) \\
& \quad \sum_{j=2}^{n} x_{jik} = 1, k = 1,2,\ldots,m#(16) \\
& \quad \sum_{i=1}^{n} \sum_{k=1}^{m} x_{ijk} = 1, j = 2,3,\ldots,n, i \neq j#(17) \\
& \quad \sum_{j=1}^{n} \sum_{k=1}^{m} x_{ijk} = 1, i = 2,3,\ldots,n, i \neq j#(18) \\
& \quad \sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{ijk} = 2,3,\ldots,n \\
& \quad k = 1,2,\ldots,m, i \neq j#(19) \\
& \quad u_i - u_j + (n - m) \sum_{k=1}^{m} x_{ijk} \leq n - m - 1, \\
& \quad 2 \leq i \neq j \leq n#(20) \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ijk} \leq Q, k = 1,2,\ldots,m#(21) \\
\end{align*}
\]

\(x_{ijk} \in \{0,1\}, i = 1,\ldots,n; j = 1,\ldots,n; k = 1,2,\ldots,m.\)

Constraints (15) - (20) are similar to constraints (8) - (13). The objective function here is to minimize the longest tour and is clearly expressed by introducing inequality...
(21) in conjunction with the new variable $Q$, where $Q$ represents the upper bound of the total travel path for each traveling salesman.

### 3.3.3 MILP formulation for Bounded-MTSP

The Bounded-MTSP adds the parameter $K$ and $L$ to constrain the minimal and the maximal number of cities that a salesman should visit. $K$ is the minimum number of cities a salesman must visit and $L$ is the maximum number of nodes a salesman may visit. The MILP formulation for Bounded-MTSP is as follows (R. Necula et al., 2015):

$$ \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} $$

subject to

$$ \sum_{j=2}^{n} x_{1j} = m $$

$$ \sum_{j=2}^{n} x_{j1} = m $$

$$ \sum_{i=1}^{n} x_{ij} = 1, j = 2, 3, \ldots, n $$

$$ \sum_{j=1}^{n} x_{ij} = 1, i = 2, 3, \ldots, n $$

$$ x_{1i} + x_{i1} \leq 1, i = 2, 3, \ldots, n $$

$$ u_i + (L - 2)x_{1i} - x_{1i} \leq L - 1, i = 2, 3, \ldots, n $$

$$ u_i + x_{1i} + (2 - K)x_{i1} \geq 2, i = 2, 3, \ldots, n $$

$$ u_i - u_j + Lx_{ij} + (L - 2)x_{ji} \leq L - 1, 2 \leq i \neq j \leq n $$

$$ x_{ij} \in \{0, 1\}, i = 1, 2, \ldots, n; j = 1, 2, \ldots, n. $$
The significance of constraints (23)-(27) is the same as the constraints (8)-(12). Inequality (28) is the upper bound constraint on the number of cities. Inequality (29) is the lower bound constraint on the number of cities. Constraints (30) are the classical subtour elimination constraints. It is worth noting that the formulation is valid if $2 \leq K \leq \lfloor (n - 1)/m \rfloor$ and $L \geq K$.

3.4 The theory of Graph Neural Networks

The classic deep learning models can extract features based on Euclidean distance, have achieved great success. However, there exists numerous graph data, which is in the non-Euclidean space, in the real-world scenarios. For example, the Standard-MTSP can be described as a graph, including nodes and edges. The graph data structure, which is irregular, is typical for represent in non-Euclidean distance. Convolution and other traditional operations are not well-suited for direct processing of graph data. To overcome this limitation, graph neural networks (GNNs) have been developed as a specialized approach to deep learning on graphs. Graph neural networks are closely related to graph embedding. The main graph neural networks are Graph Convolution Networks (GCN) (Niepert et al., 2016), Graph Attention Networks (GAN) (Velickovic et al., 2018), Graph Autoencoders (GA) (Kipf et al., 2016), Graph Generative Networks (GGN) (Li et al., 2018). We built a bipartite graph neural network suitable for bipartite graph structure based on graph embedding and attention mechanism. The details of the network are provided in Section 4.
3.5 Attention Mechanism

Attention mechanism is widely used in various deep learning assignments, such as Nature Language Translation (NLP), computer vision, machine translation, and speech recognition. The idea of attention mechanism, initially developed in 1990s, was first applied in image processing. Mnih et al. (2014) first used the attention mechanism on the RNN for image classification. Later, Vaswani et al. (2017) applied the attention mechanism in machine translation and quickly gained popularity. In our study, due to the high dimension of features in graph structure, we mainly added self-attention and make simple adjustments to fit our network. Self-attention is a type of attention mechanism. Just as its name implies, it can be understood as its weight. The following formulation describes self-attention:

Suppose we have an encoded matrix \( Y \in \mathbb{R}^{n \times d} \), denoted by \( Y^T = (Y_1,Y_2,\cdots,Y_n) \). \( Y_i \) is a row vector,

\[
Y_i = (y_{i1},y_{i2},\cdots,y_{id}), \quad i = 1,2,\cdots,n.
\]

where each \( y_{ij} = 1,2,\cdots,d \) is an eigenvalue of vector \( Y_i \). First, the query, key, and value vectors are generated by \( Y \).

\[
Q_i = Y_iW_Q, \\
K_i = Y_iW_K, \\
V_i = Y_iW_V,
\]

where \( W_Q,W_K,W_V \in \mathbb{R}^{d \times d} \) are the learnable parameters. Then, calculating the weight \( \Omega_i \) of value. \( \Omega_i \) can be interpreted as the similarity between \( Q \) and \( K \).
\[ S_i = \frac{Q_i K_i^T}{\sqrt{d}}, S_i \in \mathbb{R}^{1 \times n}, \]

\[ \Omega_i = \text{softmax}(s_i) = \frac{e^{s_{ik}}}{\sum_{k=1}^{n} e^{s_{ik}}}, \Omega_i \in \mathbb{R}^{1 \times n}. \]

Self-attention uses the dot product to calculate the similarity. In our problem, \( n \gg d \) is a variable. Therefore, replacing \( Q_i K_i^T \) with \( Q_i K_i^T \) can reduce memory consumption effectively. When the dimension is too high, the value after softmax is not smooth enough, as the value is not 0 or 1. \( \sqrt{d} \) is used to control the scale of the inner product to ensure the numerical stability. Finally, a weighted sum of values is conducted to derive the attention for the query, \( \text{Att}_i = \Omega_i \ast V_i \).

That is,

\[ \text{Att} = \text{softmax}\left(\frac{Q K^T}{\sqrt{d}}\right)V. \]

Fig. 2. illustrates the architecture of self-attention.
Fig. 2. An illustration of the self-attention

4 Method

There are two primary approaches to using deep learning for solving spatial optimization problems. The first approach assumes that traditional optimization algorithms possess robust expert knowledge. The goal is to leverage deep learning methods to achieve rapid approximations of complex computations performed by traditional algorithms, thereby accelerating problem-solving. This process does not require the generation of new explicit algorithms. The second approach assumes that optimization algorithms do not possess specialized knowledge. In this case, deep learning methods are used to explore the solution space of optimization problems. The objective of the optimization problem serves as the reward function, guiding the model to continuously learn and improve its performance. The research in this chapter is based on the first approach. MTSP can often be represented as Mixed Integer Linear Programming (MILP) and solved using the Branch-and-Bound method. To acquire a robust branching strategy, we propose using imitation learning.

Our study is based on the first approach. MTSP can often be represented as MILP and solved using the B&B. As the branching process involves intricate computations, we consider employing imitation learning to acquire a robust branching strategy. A MILP problem consists mainly of an objective function, constraints, and variables. Given a problem instance, we construct a bipartite graph of constraints and variables. A Bipartite Graph Neural Network (BiGNN) is proposed to learn feature
embeddings between constraints and variables. Attention mechanisms are incorporated to enhance the model's performance.

### 4.1 The workflow of proposed method

There are two sets of features, constraints, and variables, in the optimization. In our method, we construct a bipartite graph based on the connection between constraints and variables, and propose a new structure of neural network. It focuses on graph neural network and attention mechanisms to learn deep embeddings, representing and imitating the expert branching strategy to save solution time. The proposed workflow is shown in Fig. 3. First, converting the MTSP into a MILP and record the sub-problem as the input of the network. Then, BiGNN or BiGNN with attention (BiGNN+ATT) is used to learn the hidden embedding between constraints and variables, and output the probability of every candidate variable. Finally, adopting the prediction results to guide the SCIP solver.

![Fig. 3. The workflow of the proposed method](image-url)
This workflow is under mainly algorithm process. The machine learning model is frequently used to assist lower-level decision-making. Expert decision data \( \{ \tau_1, \tau_2, \cdots, \tau_n \} \) is generated by running expert strategies on training example, with each sequence decision containing a series of states and actions \( \tau_i = (< s^1_i, a^1_i, s^2_i, a^2_i, \cdots, s^n_i, a^n_i >) \). The model extracts all state-action pairs to generate an expert strategy set \( D = \{ (s_i, a^*_i) \}_{i=1}^N \). Next, it classifies the state as the feature and the action as the label, and finally learn the strategy by minimizing the cross-entropy loss

\[
L(\theta) = -\frac{1}{N} \sum_{(s, a^*) \in D} \log \pi_\theta (a^* | s)
\]

### 4.2 Feature Representation and Bipartite Graph Network

In a search tree building process, we choose variables according to the state of the current node. It can be regarded as a classification or ranking problem. Here, we treat it as a classification problem. Take the generated expert branch strategy

\[
D = \{ (s_i, a^*_i) \}_{i=1}^N
\]

as training sets, where \( s_i \) are features and \( a_i \) are labels. We expected to input the state \( s \) at time \( t \) in our BiGNN model and output the optimal variable which doesn't generate an explicit method. Based on the study of Gauss et al. (2019), we encoded \( s_t \) into a bipartite graph

\[
G = (C, V, E)
\]

where \( C \in R^{m \times c} \) represents constraint features, \( V \in R^{m \times d} \) represents variable features, \( E \in R^{k \times 1} \) represents edge features and \( k < m \times n \) is an integer.
Due to the specialist of bipartite graphs, the bipartite graph network (BGN) is proposed to extract the bipartite features. The basic process is shown in Fig. 4. We used a bipartite graph network to decompose initial features into two consecutive transfers. One is from variable to constraint, and another is vice versa. The structure of BGN is shown in Fig. 5. New constraints or variables are generated according to the edge features about constraints and variables, containing the information from their neighbors.

![Fig. 4. The basic process of bipartite graph network](image-url)
The following list a simple example for better understanding of the bipartite graph network. Assume that

\[ C \in \mathbb{R}^{2 \times 5}, V \in \mathbb{R}^{3 \times 19}, E \in \mathbb{R}^{4 \times 1}, \]

Here, the notation of \( m = 2 \) and \( n = 3 \) means that there are two constraints and three variables. \( k = 4 \) means that there are four edges related to constraints and variables. The corresponding bipartite graph is shown in Fig. 6. The column of the tensor \( C, V, E \) represents the dimension of each feature, which is consistent with Gauss et al. (2019). The BiGNN structure is shown in Fig. 7.
First, pass the original features $C$ and $V$ through preformed layer and two convolution layers to obtain $C_1$ and $V_1$. Second, input $C_1$, $V_1$ and $E$ into BGN.

Referring to the edge features and concatenating $C_1$ and $V_1$, new constraint features $C_2$ are obtained. Then, passing $C_2$, $V_1$ and $E$ through BGC to generate new variable features $V_2$. Next, the self-attention is introduced in our network. It can be jointed to
BiGNN as an independent block. All bipartite graphs nearly generated by MILP have high dimensions in real-world problems. When the network adds self-attention, the calculation and memory usage will increase dramatically. The structure of self-attention has already been fine-tuned to adapt to our BiGNN with the purpose of avoiding this problem. Finally, after one layer of fully connected layer, the probability distribution $\pi(x)$ is generated by combining the softmax function without activation function.

Table 1. The description of constraint, edge, and variable features Tensor

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj_cos_sim</td>
<td>Cosine similarity with objective.</td>
</tr>
<tr>
<td>bias</td>
<td>Bias value, normalized with constraint coefficients</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>is_tight</td>
<td>Tightness indicator in LP solution.</td>
</tr>
<tr>
<td>dualsol_cal</td>
<td>Dual solution value, normalized</td>
</tr>
<tr>
<td>age</td>
<td>LP age, normalized with total number of LPs.</td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>coef</td>
<td>Constraint coefficient, normalized per constraint</td>
</tr>
<tr>
<td>type</td>
<td>Type (binary, integer, impl. integer, continuous) as a one-hot encoding,</td>
</tr>
<tr>
<td>coef</td>
<td>Objective coefficient, normalized.</td>
</tr>
<tr>
<td>has_lb</td>
<td>Lower bound indicator.</td>
</tr>
<tr>
<td>has_ub</td>
<td>Upper bound indicator.</td>
</tr>
<tr>
<td>V</td>
<td></td>
</tr>
<tr>
<td>sol_is_at_lb</td>
<td>Solution value equals lower bound</td>
</tr>
<tr>
<td>sol_is_at_ub</td>
<td>Solution value equals upper bound</td>
</tr>
<tr>
<td>sol_frac</td>
<td>Solution value fractionality</td>
</tr>
<tr>
<td>basis_status</td>
<td>Simplex basis status (lower, basic, upper, zero) as a one-hot encoding</td>
</tr>
<tr>
<td>reduced_cost</td>
<td>Reduced cost, normalized.</td>
</tr>
<tr>
<td>age</td>
<td>LP age, normalized</td>
</tr>
</tbody>
</table>
5 Experiments

To demonstrate that our approach can learn the strong branch strategy, we compared the BiGNN and BiGNN with attention (BiGNN + ATT) with three machine learning branches: TREES (Alvarez et al., 2017), SVMRANK (Khalil et al., 2016) and LMART (Hansknecht et al., 2018). Moreover, we generated various instances about the Standard-MTSP, MinMax-MTSP and Bounded-MTSP to evaluate our model. During all experiments, we set the maximum solving time to 1.5 hours in the SCIP. All SCIP parameters are set to default for ensuring comparisons fair and repetitive as possible.

All experiments are running on a GPU environment with an Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz and an NVIDIA TITAN RTX GPU (cuda9.2, cudnn7.6.5, and tensorflow-gpu1.12.0). The source code is available at https://github.com/CO-RL/DeepMTSP.

5.1 Date generation

Our idea is to make a learning guidance for the expert branching strategies in branch-and-bound generate the smaller search trees, instead of directly solving the combinatorial optimization problem.

Therefore, data is classified as two categories: instances and samples. Instances are randomly generated according to the MILP form of the Standard-MTSP, MinMax-
MTSP and Bounded-MTSP, mentioned in Section 3. Each instance can represent a realistic combinatorial optimization problem.

Samples are supported to train machine learning models which are generated by instances. Each generated instance adopts the open-source solver SCIP to solve and record the expert branching strategies in the solution process. Each episode can be recorded as a training sample.

The data generation process is as follows:

1) We randomly generated 10,000 training, 2,000 valid and 2,000 test instances which are preparing for generating samples.

2) Using SCIP, solve corresponding instances to generate 20,000 samples as train set, 4,000 samples as validation set, and 4,000 samples as test set.

3) To discover the impact of varying data set sizes, we generated a larger dataset, including 100,000 samples as train data, 20,000 samples as validation set, 20,000 samples as test set.

4) 100 instances are generated to evaluate the results.

5.2 Test accuracy

It is necessary to illustrate that our framework effectiveness is suitable for dealing with the Standard-MTSP, MinMax-MTSP and Bounded-MTSP. Three machine learning algorithms were compared with our model. The SVMRANK and LMART both used the original features proposed by Khalil et al. (2016). TREES, BiGNN, BiGNN +ATT used the features were consistent with Gauss et al. (2019). All
the features, including constraint, edge and variable, and the description are given in Table 1.

The training set was used to train each model five times, with different random seeds employed in each iteration. The trained model was tested on 4,000 test samples, and the average accuracy of five seeds was represented in the result in tables. Table 2 shows the average test accuracy in five models on Standard-MTSP, MinMax-MTSP and Bounded-MTSP with 20,000 training samples. Table 3 shows the average test accuracy in five models on Standard-MTSP, MinMax-MTSP and Bounded-MTSP with 100,000 training samples. It includes the highest-ranking decision of the model (acc@1), one of the three highest decisions (acc@3), one of the five highest decisions (acc@5). And one of the ten highest decisions (acc@10) is the variable assigned the highest strong branching score.

Test accuracy represents the ability of the model to learn strong branch strategy. The BiGNN and BiGNN +ATT is much higher than other three algorithms, which means that BiGNN is superior to the traditional machine learning algorithm in these problems. By comparing the experiment of 20,000 samples and 100,000 samples, we can aware that increasing the amount of data can effectively improve the test accuracy. In our study, the data set is randomly generated by the mathematical model of the problem. It is also a valid method to improve the accuracy by increasing the scale of samples.

Table 2. Test accuracy on five models of MTSP

<table>
<thead>
<tr>
<th>Model</th>
<th>acc@1</th>
<th>acc@3</th>
<th>acc@5</th>
<th>acc@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREES</td>
<td>41.5±0.5</td>
<td>69.9±0.6</td>
<td>84.1±0.6</td>
<td>97.2±0.1</td>
</tr>
<tr>
<td>Model</td>
<td>acc@1</td>
<td>acc@3</td>
<td>acc@5</td>
<td>acc@10</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>TREES</td>
<td>43.4±0.4</td>
<td>69.3±0.3</td>
<td>81.3±0.3</td>
<td>94.5±0.1</td>
</tr>
<tr>
<td>LMART</td>
<td>53.5±0.2</td>
<td>74.2±0.5</td>
<td>83.0±0.3</td>
<td>92.4±0.2</td>
</tr>
<tr>
<td>SVMRANK</td>
<td>49.7±0.6</td>
<td>75.2±0.3</td>
<td>85.8±0.3</td>
<td>95.6±0.3</td>
</tr>
<tr>
<td>BiGNN</td>
<td>58.2±1.3</td>
<td>82.4±1.0</td>
<td>91.5±0.7</td>
<td>96.9±0.6</td>
</tr>
<tr>
<td>BiGNN+ATT</td>
<td>58.1±1.1</td>
<td>82.2±0.7</td>
<td>91.3±0.3</td>
<td>97.0±0.4</td>
</tr>
</tbody>
</table>

**MinMax-MTSP (20,000)**

<table>
<thead>
<tr>
<th>Model</th>
<th>acc@1</th>
<th>acc@3</th>
<th>acc@5</th>
<th>acc@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREES</td>
<td>35.4±0.1</td>
<td>60.6±0.3</td>
<td>80.1±0.1</td>
<td>95.1±0.1</td>
</tr>
<tr>
<td>LMART</td>
<td>50.1±0.2</td>
<td>71.3±0.4</td>
<td>83.0±0.2</td>
<td>95.2±0.1</td>
</tr>
<tr>
<td>SVMRANK</td>
<td>48.5±0.6</td>
<td>70.2±0.2</td>
<td>82.8±0.4</td>
<td>95.7±0.1</td>
</tr>
<tr>
<td>BiGNN</td>
<td>55.1±1.0</td>
<td>79.9±0.5</td>
<td>91.6±0.6</td>
<td>96.7±0.5</td>
</tr>
<tr>
<td>BiGNN+ATT</td>
<td>55.3±0.9</td>
<td>79.8±0.2</td>
<td>91.4±0.5</td>
<td>97.1±0.6</td>
</tr>
</tbody>
</table>

**5.3 Evaluation**

To present the results clearly, we randomly generate instances with different scales for three problems, which can be solved by SCIP and our models. We mainly show six simple instances. Fig. 8 shows the optimal path obtained by solving two Standard-MTSP, Fig. 9 depicts the optimal path obtained by solving two MinMax-MTSP, and Fig. 10 presents the optimal path obtained by solving two Bound-MTSP. Table 4 compares the solution time and optimal solution of the six algorithms to solve these four problems. RBF represents the branching strategy used by SCIP. The optimal solution can always be found since the small scale of problem. We focus on comparing the solution time of several methods. Our model requires less time to obtain the optimal solution. More importantly, we explored the generalization of the model. We only trained the models on the 9 cities and 3 salesmen and transferred to larger-scale problems and real-world scenarios.
Table 3. Test accuracy on five models of MinMax-MTSP

<table>
<thead>
<tr>
<th>Model</th>
<th>acc@1</th>
<th>acc@3</th>
<th>acc@5</th>
<th>acc@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREES</td>
<td>43.0±0.5</td>
<td>70.8±0.4</td>
<td>84.0±0.4</td>
<td>96.9±0.1</td>
</tr>
<tr>
<td>LMART</td>
<td>53.2±0.2</td>
<td>79.7±0.1</td>
<td>89.3±0.1</td>
<td>98.1±0.0</td>
</tr>
<tr>
<td>SVMRANK</td>
<td>52.5±0.2</td>
<td>79.4±0.1</td>
<td>89.2±0.1</td>
<td>98.1±0.0</td>
</tr>
<tr>
<td>BiGNN</td>
<td>60.4±0.3</td>
<td>84.2±0.1</td>
<td>92.5±0.1</td>
<td>99.0±0.0</td>
</tr>
<tr>
<td>BiGNN+ATT</td>
<td>59.9±0.4</td>
<td>84.0±0.4</td>
<td>92.4±0.2</td>
<td>98.9±0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>acc@1</th>
<th>acc@3</th>
<th>acc@5</th>
<th>acc@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREES</td>
<td>44.4±1.4</td>
<td>71.4±0.6</td>
<td>83.4±0.4</td>
<td>95.2±0.1</td>
</tr>
<tr>
<td>LMART</td>
<td>59.0±0.1</td>
<td>77.2±0.3</td>
<td>84.5±0.3</td>
<td>92.8±0.1</td>
</tr>
<tr>
<td>SVMRANK</td>
<td>55.4±0.4</td>
<td>79.4±0.2</td>
<td>89.0±0.2</td>
<td>96.4±0.2</td>
</tr>
<tr>
<td>BiGNN</td>
<td>61.8±0.6</td>
<td>85.8±0.4</td>
<td>93.6±0.1</td>
<td>98.0±0.1</td>
</tr>
<tr>
<td>BiGNN+ATT</td>
<td>61.5±0.3</td>
<td>85.6±0.3</td>
<td>93.5±0.1</td>
<td>98.0±0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>acc@1</th>
<th>acc@3</th>
<th>acc@5</th>
<th>acc@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TREES</td>
<td>38.4±0.9</td>
<td>68.5±0.3</td>
<td>82.4±0.5</td>
<td>95.2±0.1</td>
</tr>
<tr>
<td>LMART</td>
<td>55.1±0.2</td>
<td>78.2±0.2</td>
<td>83.5±0.6</td>
<td>92.8±0.1</td>
</tr>
<tr>
<td>SVMRANK</td>
<td>53.4±0.5</td>
<td>76.4±0.1</td>
<td>86.0±0.6</td>
<td>96.4±0.2</td>
</tr>
<tr>
<td>BiGNN</td>
<td>58.2±0.6</td>
<td>83.2±0.3</td>
<td>90.7±0.2</td>
<td>97.3±0.1</td>
</tr>
<tr>
<td>BiGNN+ATT</td>
<td>58.1±0.3</td>
<td>83.1±0.4</td>
<td>90.1±0.3</td>
<td>96.9±0.2</td>
</tr>
</tbody>
</table>

Table 4. The solved time and optimal solution on four cases

<table>
<thead>
<tr>
<th>Model</th>
<th>Instance 1</th>
<th>Instance 2</th>
<th>Instance 3</th>
<th>Instance 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>Optimal</td>
<td>T</td>
<td>Optimal</td>
</tr>
<tr>
<td>RBF</td>
<td>4.539</td>
<td>0.161</td>
<td>517.9</td>
<td>34.71</td>
</tr>
<tr>
<td>TREES</td>
<td>4.423</td>
<td>0.160</td>
<td>727.1</td>
<td>321.5</td>
</tr>
<tr>
<td>LMART</td>
<td>3.604</td>
<td>0.159</td>
<td>483.7</td>
<td>47.86</td>
</tr>
<tr>
<td>SVMRANK</td>
<td>4.987</td>
<td>0.157</td>
<td>494.2</td>
<td>118.2</td>
</tr>
<tr>
<td>BiGNN</td>
<td>2.977</td>
<td>0.161</td>
<td>301.9</td>
<td>306.3</td>
</tr>
<tr>
<td>BiGNN+ATT</td>
<td>3.168</td>
<td>0.161</td>
<td>318.4</td>
<td>17.36</td>
</tr>
</tbody>
</table>

Fig. 8. Standard-MTSP
Other than obtaining the optimal solution on the above synthetic data set, our models can also deal with the problem along with the road network. Based on the Surat Road network data, we randomly chose 13 locations to find the optimal solution for MTSP, Bound-MTSP, and MinMax-MTSP. The result is shown in Fig. 11, 12, and 13, respectively. The experimental results demonstrate that our model exhibits promising applicability to the road network. It is of great significance to extend our model to other road traffic problems.
Fig. 11. The solution for the Standard-MTSP along road network

Fig. 12. The solution for the MinMax-MTSP along road network
6 Conclusions

In this study, we adopted the idea of L2B to solve the Standard-MTSP, MinMax-MTSP, and Bounded-MTSP. A framework based on BiGNN and attention mechanism is proposed to learn strong branch strategy. Without deriving display algorithm, fast approximation is used to replace heavy calculation and reduce the solution time. For Standard-MTSP, the mainstream research method is heuristic algorithm. However, the design and construction of heuristic algorithm are quite difficult and usually only get the approximate solution of the problem. L2B provides a new idea for solving combinatorial optimization problems. BiGNN and BiGNN+ATT perform better in results, compared to the traditional machine learning algorithm, which therefore provides a new tool for solving these problems.

Although the research of L2B has made progress, there are still many challenges when applying to practical problems. First, there is no mature MTSP dataset that can evaluate all kinds of methods uniformly. It is effective to generate numerous expert
datasets randomly to support the learning of our model. If we can add constraints to optimize or adjust the data set according to practical problems, we have the potential to greatly enhance the model’s performance. Second, it usually takes too much time for SCIP to solve the MILP transformed by the MTSP. Even if the scale of problems remains the same, the solution time varies greatly, which is not conducive to learning. Third, although the test accuracy of our model is much higher than other algorithms, it can be further improved. Our model can learn the strong branch strategy to a certain degree, but in the evaluation process, it cannot always generate the minimum search tree according to the strong branch strategy. This also accounts for the fact that time is not a dominant factor while solving some instances.

Nowadays, deep learning has been rapidly developed. Lots of significant achievements in various fields have been made. Integrating neuroscience with geographical issues also has broad prospects (Zhong, 2020). In this work, we mainly used imitation learning to study the problems, while imitation learning is a branch of RL. Deep reinforcement learning (DRL) has significant advantages in dealing with sequential decision-making problems. The DRL model can be trained through self-learning, which can automatically find the strategy to solve the problem and improve the generalization performance of the model. In the future, we will integrate DRL with L2B, which is an exciting research direction. When DRL meets combinatorial optimization, it is destined to provide an innovative approach for solving city logistics problems in many real-world scenarios.
Acknowledgement

The research were financially supported by the innovation group project of the Key Laboratory of Remote Sensing and Digital Earth Chinese Academy of Sciences (E33D0201-5), CBAS project 2023, and the Beijing Chaoyang District Collaborative Innovation Project (E2DZ050100).

References


Lupoaie, V. I., Chili, I. A., Breaban, M. E., & Raschip, M. (2019, June). SOM-guided evolutionary search for solving MinMax multiple-TSP. In 2019 IEEE congress on evolutionary computation (CEC) (pp. 73-80). IEEE.


