Solving the Traveling Thief Problem Based on Item Selection Weight and Reverse-Order Allocation

ZITONG ZHANG, LEI YANG, PEIPEI KANG, XIAOTIAN JIA, AND WENSHENG ZHANG

1College of Mathematics and Information, South China Agricultural University, Guangzhou 510642, China
2College of Computer, Guangdong University of Technology, Guangzhou 510006, China
3Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
4Guangdong Provincial Key Laboratory of Food Quality and Safety, South China Agricultural University, Guangzhou 510642, China

Corresponding author: Lei Yang (yanglei_s@scau.edu.cn)

This work was supported in part by the Natural Science Foundation of Guangdong Province of China under Grant 2020A1515010691, in part by the Science and Technology Project of Guangdong Province of China under Grant 2018A0124, in part by the Guangdong Provincial Key Laboratory of Food Quality and Safety under Grant 2020B1212060059, and in part by the National Natural Science Foundation of China under Grant 61573157 and Grant 61703170.

ABSTRACT

The traveling thief problem (TTP) is a challenging combinatorial optimization problem that has attracted many scholars. The problem interconnects two well-known NP-hard problems: the traveling salesman problem and the 0-1 knapsack problem. Various approaches have increasingly been proposed to solve this novel problem that combines two interdependent subproblems. In this paper, the TTP is investigated theoretically and empirically. A novel method to calculate the score value of each item for item selection, which expands the effect of the item’s weight, was introduced. Furthermore, the approach adopted reverse-order allocation, which selects items in an inverse order according to the traveling route. Different approaches for solving the TTP are compared and analyzed. The experimental investigations suggest that our proposed approach is competitive for many instances of various sizes and types compared to other heuristics.

INDEX TERMS

Traveling thief problem, interdependence, item selection weight, reverse order allocation.

I. INTRODUCTION

Numerous practical applications include two or more subproblems, many of which can be summarized as combinatorial optimization problems. Combinatorial optimization is one of the most challenging problems, which usually involves traversing a search space to determine an optimal or approximately optimal solution from a bounded solution set while maximizing (or minimizing) the objective function. Many interdependent components make it challenging to solve such problems. Optimally solving each component does not ensure obtaining an optimal solution to the overall problem. This type of problem is prevalent in supply chain management (e.g., distribution, scheduling, loading, transportation, etc.) [3], [11], vehicle routing problems, logistics problems, and others. Some optimization problems are difficult to address because interdependency among components in operational or dynamic problems plays a crucial role in the complexity of the problems [18].

The TTP combines two combinatorial problems: the traveling salesman problem (TSP) and the 0–1 knapsack problem (KP). A benchmark problem called the traveling thief problem (TTP) was introduced by Bonyadi et al. [2] in 2013 to demonstrate the complexity that arises from interdependency in multi-component problems. This problem can be illustrated in the following way.

A thief travels a cyclic journey through n cities, uses a picking plan, selects m items to put into a rented knapsack with a constrained capacity. As items are picked up at each subsequent city to fill the knapsack, the total profit based on items and weight in the knapsack increases. The knapsack becomes heavier, the thief becomes slower, increasing the overall travel time and renting cost. The overall goal of the TTP is to concurrently maximize the total profit of the chosen items and minimize the renting cost. From the above statement, the two components of the TTP interact with each other. When the weight of the knapsack increases, it affects thief’s speed, increasing the rental time of the knapsack.
items in the corresponding city also changes. This interdependence between the two components makes the problem complicated.

Since the TTP was introduced by Bonyadi et al. [2] as a benchmark problem for solving multi-component and interdependence problems, many scholars have proposed corresponding algorithms to solve this problem. Some approaches have been introduced to solve this problem, such as heuristics, cooperative methods, and so on. Polyakovskiy et al. [22] were the first to create many benchmark instances and propose several heuristic algorithms to solve the TTP. An initial cyclic tour sequence is generated for the TSP component using the Lin-Kernighan heuristic (LKH) [12], and then items are selected under a fixed route until the optimal solution is obtained. In their first method for solving the TTP, called a simple heuristic, items are selected based on the score value. They also proposed iterative heuristics called random local search and (1 + 1) EA based on flipping the selected items with a specific probability.

Bonyadi et al. [4] proposed a heuristic method to address the TTP. In their approach, the TTP is disintegrated into subproblems (TSP and KP). They processed the two sub-problems while maintaining communication between them and composed solutions to obtain a final solution called the CoSolver. They also proposed an approach called density-based heuristics, where a tour is generated for TSP, and then a solution for KP is generated in a fixed tour.

Mei et al. [17] introduced two evolutionary heuristic approaches for solving TTP. The first approach is cooperative co-evolution which is to solve each subproblem independently without considering the dependencies. The second approach is the memetic algorithm that solves this problem as a whole and considers the dependencies between each subproblem. The efficient memetic algorithm with two-stage local search (MATLs) was proposed by Mei et al. [16] to solve the large-scale TTP with several complexity reduction methods.

In the KP component of the TTP, an optimized picking plan called PackIterative was proposed by Faulkner et al. [10]. To avoid the bias toward the KP component, they proposed an insertion operator to optimize the tour iteratively for a fixed picking plan generated by LKH [12]. Several simple iterative heuristics (S1-S5) and some complex heuristics were proposed. According to the performance analysis, an simple iterative heuristic called S5 exhibits the best performance on average among all approaches.

Yafrani and Ahiod [7] introduced two heuristic algorithms. They compared two traditional types of search heuristics: population-based heuristics and single-solution heuristics. The first method is the memetic algorithm with 2-opt and bit-flip (MA2B), which uses a genetic algorithm based on population evolution. The other approach is a single-solution based heuristic method, called the CoSolver-based with 2-opt and simulated annealing (CS2SA), which applies 2-opt steepest ascent hill-climbing heuristic and adapted simulated annealing for efficient item picking to solve the TTP.

These two algorithms perform more competitive compared to MATLs and S5 on many TTP instances. After that, CS2SA* and CS2SA-R were introduced based on CS2SA. In addition, CS2SA* is an implementation of CS2SA with instance-based parameter tuning, whereas CS2SA-R uses random restarts when no improvements occur in returning the so-far best solution.

Wagner [24] studied the swarm intelligence approach, focusing on short TSP tours and good TTP tours for solving the TSP component of the TTP based on ant colony optimization. This method is effective and computationally efficient for small instances of the TTP. However, its performance deteriorates significantly for many large instances. Neumann et al. [21] investigated the underlying non-linear packing while traveling Problem of the TTP where the items are selected for a fixed route. They provide an exact dynamic programming approach and a fully polynomial time approximation scheme to solve this problem while maximizing the benefit.

Yafrani and Ahiod [8] proposed two simple iterative neighborhood algorithms based on the local search. The first approach, called joint N1-BF, is a neighborhood-based heuristic that combines the N1 neighborhood (swapping two adjacent cities) of TSP and the one-bit-flip of KP. The second is joint 2-opt-BF, a combination of the 2-opt heuristic and one bit-flip heuristic. Martins et al. [15] introduced heuristic selection based on estimation of distribution algorithm approach. This method applies the estimation of distribution algorithm probabilistic model using an approximation function to find better heuristics to solve the TTP. Martins et al. confirmed that this approach outperforms other algorithms on most the medium-sized TTP instances.

Wu et al. [25] proposed three exact algorithms and a hybrid approach to solve the TTP. These are dynamic programming, the branch and bound search, and constraint programming.

Alharbi et al. [1] introduced a modified artificial bees colony algorithm based on swarm intelligence to solve the TTP interdependently. This algorithm is efficient in mid-sized TTP instances compared to the state-of-the-art approaches. Namazi et al. [19], [20] proposed an extended and modified form of the reversing heuristic to consider both the TSP and KP components concurrently. Items regarded as less profitable and those selected in cities beginning in the reversed segment are substituted with items that tend to be equally or more profitable and are not selected in later cities. Maity et al. [13] introduced a scoring value calculated using the proposed formulation to choose items for a fixed picking plan generated by the chained LKH.

Earlier work by Yafrani and Ahiod [9] demonstrated that the KP component of the TTP is more critical compared to the TSP component for optimization. For this reason, our work focus on the KP component of the TTP. In this paper, the LKH generates a nearly optimal tour (the TSP component of the TTP), we focus whether the weight is a more significant determinant of the final profit than other item attributes (e.g., location in the tour and item value). Therefore, a method
for calculating the score value of the item to determine its effect on the final profit is proposed. We believe that the final profit is related not only to the value, weight, and location of the items but also to the order in which the items are selected. The effect of the item selection order on the final profit is also discussed in this article. The contributions of this paper are as follows: An approach based on item selection weight and reverse-order allocation is proposed. A novel method to calculate the score value of each item for item selection, which expands the item’s weight’s effect, is described. In addition, reverse-order allocation, which selects items in the inverse order to the traveling route, is adopted.

The rest of this paper is organized as follows. In Section 2, the backgrounds of the TTP and some of the heuristics are introduced. In Section 3, the proposed algorithms are applied to some TTP instances, and the experimental results are reported and discussed in Section 4. Finally, Section 5 concludes the paper and outlines some future directions of research.

II. BACKGROUND

In this section, we present a brief background introduction of the TTP. Some common heuristics for the TSP and KP are briefly reviewed.

A. TRAVELING THIEF PROBLEM

The TTP is a combination of two well-known benchmark problems: the TSP and the KP. In the TTP, we consider \( n \) cities and their associated distance matrix \( \{d_{i,j}\}_{i \neq j} \). The distance between each pair of cities is \( d_{i,j} = d_{j,i} \) \((i, j \in \{0, \ldots, n\})\). There are \( m \) items scattered in these cities. Each item \( j \) \((j \in \{0, \ldots, m\})\) is located at city \( l_{j} \) with a profit \( p_{j} > 0 \) and a weight \( w_{j} > 0 \). A thief starts in the first city and travels to visit all these cities only once, choosing a subset of the available items in each city. Each item is available in only one city, and we note \( A_{j} \in \{1, \ldots, n\} \) is the availability vector that contains the reference of the city where item \( j \) is located.

The cyclic tour is designed using a permutation of \( n \) cities and their associated distance matrix \( \{d_{i,j}\}_{i \neq j} \). The distance between each pair of cities is \( d_{i,j} = d_{j,i} \) \((i, j \in \{0, \ldots, n\})\). There are \( m \) items scattered in these cities. Each item \( j \) \((j \in \{0, \ldots, m\})\) is located at city \( l_{j} \) with a profit \( p_{j} > 0 \) and a weight \( w_{j} > 0 \). A thief starts in the first city and travels to visit all these cities only once, choosing a subset of the available items in each city. Each item is available in only one city, and we note \( A_{j} \in \{1, \ldots, n\} \) is the availability vector that contains the reference of the city where item \( j \) is located.

The tour \( c \) is designed using a permutation of \( n \) cities.

Given a tour \( c \), we define \( c_{k} = i \), where \( i \) is the \( k \)-th city in the tour \( c \), and we define \( c(i) = k \), where the location of city \( i \) in tour \( c \) is \( k \). A knapsack with a maximum weight capacity \( W \) and a rent rate \( R \) per time unit is rented by the thief to carry the selected items. The minimum velocity is \( v_{\text{min}} \) and the maximum velocity is \( v_{\text{max}} \) when the knapsack is full. The total weight of the items in the knapsack must not exceed the maximum weight limit. The speed of the thief varies with the weight of the backpack. The thief becomes slower in the tour when the knapsack becomes heavier.

A solution of the TTP is represented as follows:

- The tour \( c = (c_{1}, c_{2}, \ldots, c_{n}) \) is a vector containing the permutation of cities.
- The picking plan \( z = (z_{1}, z_{2}, \ldots, z_{m}) \) is a binary vector, which determines that an item is picked if \( z_{j} = 1 \) or is not picked if \( z_{j} = 0 \).

The interdependence of the two subproblems in the TTP problem is reflected in the dependence of the thief’s speed and the total knapsack weight. The total weight of the items picked from city \( i \) is given in Equation 1, and the total weight of the items picked from the first city to the \( k \)-th city in the cyclic tour \( c \) is given in Equation 2. The velocity of the thief decreases linearly with the increase in the total knapsack weight. We note that \( v_{c,z}(k) \) is the velocity at the city \( c_{k} \) in Equation 3, and \( C = \frac{v_{\text{min}} - v_{\text{max}}}{W} \).

\[
W_{c}(i) = \sum_{j=1}^{n} w_{j} z_{j},
\]

\[
W_{c,z}(k) = \sum_{k'=1}^{n} W_{c}(c_{k'}),
\]

\[
v_{c,z}(k) = v_{\text{max}} - W_{c,z}(k) \times C.
\]

The goal of the TTP is to determine a proper tour \( c \) and a picking plan \( z \) to maximize the total gain \( G(c, z) \) defined in Equation 4. In other words, the goal is to maximize the total profit while minimizing the total renting cost of the knapsack. The total weight of the picked item must not exceed the capacity of the knapsack:

\[
\text{maximization} \quad G(c, z) = \sum_{i=1}^{m} p_{i} z_{i} - R \times T(c, z),
\]

\[
T(c, z) = T_{c,z}(n + 1) = T_{c,z}(n) + \frac{d(c_{n}, c_{1})}{v_{c,z}(n)},
\]

\[
T_{c,p}(k) = \sum_{k'=1}^{k-1} \frac{d(c_{k'}, c_{k'+1})}{v_{c,z}(k')},
\]

\[
\text{s.t.} \quad \sum_{k=1}^{m} w_{k} z_{k} \leq W.
\]

B. THE TSP COMPONENT

The LKH introduced by Lin and Kernighan [12] is a generalization of the 2-opt search algorithm for solving TSP. This algorithm and the chained LKH are often used to optimize TSP problems and to initialize the TSP component of the TTP. The 2-opt (a segment-reversing heuristic) is often used to modify the tour \( c \) to solve the TSP component of the TTP. On a tour given by two positions \( i \) and \( j \) \((1 < i < j \leq n)\), the order of the visited cities between these two positions is reversed to obtain a new tour. The 2-opt function is defined as follows:

\[
c'(j - k, i + k) = 2OPT(c(i + k, j - k)),
\]

\[
\text{s.t.} \quad 0 < i < j \leq n; \quad 0 \leq k \leq j - i.
\]

The Delaunay triangulation method [6] is used as a candidate generator for the 2-opt heuristic. Generated candidate objects using the Delaunay triangulation can reduce the time complexity without significantly reducing the solution quality. Moreover, tracking the time and weight information at each city of a given tour in the TTP can also reduce the total time budget.
C. IN THE KP COMPONENT

To solve the KP component, the bit-flip operator introduced by Faulkner et al. [10] is often used to optimize the packing plan \( z \). The bit-flip operator works iteratively by flipping each bit in the packing plan. Given a picking plan and a selected item \( j \), the picking state \( z_j \) is flipped from 0 to 1 or vice versa to obtain a new picking plan \( z' \). If the performance is improved after the bit-flip operation, this state is kept; otherwise, the bit-flip operation continues until the termination condition is reached.

III. PROPOSED APPROACH

This section describes our idea of optimizing the TTP, illustrates with examples, and proposes an algorithm for solving the TTP.

In the definition of the TTP given in Section 2, a tour and an item picking plan are required. First, the TSP search heuristic renders a TTP solution with an empty tour. Then, the items must be inserted into the empty tour to increase the profit. It is common to employ a proper measure of the elements of a problem to make a judgment. A scoring function for picking items is introduced to determine which items should be picked. This function is commonly based on profit, weight, and distance from the city where items are picked to the final city. A typical function used for this is \( \text{ScoreValue}_{i,k}(c) = \frac{p_k}{w_k \times \sum d_{i,j}} \) (or another form), where \( c \) is the tour, \( p_k \) is the profit, \( w_k \) is the weight of the item \( k \) in city \( i \), and \( \sum d_{i,j} \) is the distance from the city \( i \) where the item is picked to the end of the tour, \( j \in (2, \cdots, n) \), and \( k \in (1, \cdots, m) \). The higher the score of an item, the more likely it is to be selected. However, if an item is of high profit but very heavy and is close to the start city in the tour, the score of the item is also relatively high, according to the principle that the greater the score of the item, the more likely the item is to be chosen. Picking up this item affects other items in the tour that are close to the end city, and items with a high score value may not be chosen, which may slow the thief, making the total profit smaller.

Without considering picking up other items, the change in profit caused by inserting an item \( k \) in city \( i \) into the overall profit is \( \Delta p_{i,k} = p_k - R \times \frac{\sum d_{i,j}}{v_{\text{max}} - w_k \times C} \). However, the fact that a single item changes the overall profit is closely related to the previously selected items. The actual change in profit caused by inserting an item \( k \) in city \( i \) is \( \Delta p'_{i,k} = p_k - R \times \frac{\sum d_{i,j}}{v_{\text{max}} - w_k \times C} \). In addition, \( \Delta p'_{i,k} \) may be a positive number (\( \Delta p'_{i,k} > 0 \)), due to the accumulated weight of the previously chosen items. \( \Delta p'_{i,k} \) may become a negative number (\( \Delta p'_{i,k} < 0 \)). Therefore the effect of a single item on the overall profit must be considered in the cumulative effect of the previously chosen items.

Furthermore, owing to the cumulative effect of the weight of the selected items, the order in which the items are chosen must be considered. In the following section, we use a function of a sequence of numbers or sets called reverse-search. For any position \( k \) of a given sequence of \( n \) numbers \( S = (s_1, s_2, \cdots, s_n) \), the reverse-search function is defined as follows:

\[
S'(s_n, \cdots, s_1) = \text{Rev}(S(s_1, \cdots, s_n)).
\]

The function defined in Equation 9 is similar to the 2-opt function; the difference is that the elements in the 2-opt function are numbers, and the elements in this function can be numbers or sets. For the TTP, the elements in this function are the sets of item attributes (weight, value, scoring value, etc.) in the city.

The following example illustrates the function. We consider the simple TTP instance depicted in Fig. 1, which illustrates an example of the TTP with \( n = 5 \) cities and \( m = 4 \) items. Each city is assigned with a set of items except the first city. The nodes represents the cities. For example, Node 2 is associated with the item of profit \( p_1 \) equal to 101 with a weight \( w_1 \) of 10. Suppose that the knapsack capacity \( W \) is 10, the renting rate \( R \) is 1, the maximum speed \( v_{\text{max}} \) is 1, and the minimum speed \( v_{\text{min}} \) is 0.1. Furthermore, we assume that an inter solution has a tour \( c \) equal to \( [1, 2, 3, 4, 5] \) and that the picking plan \( z \) is \( [1, 0, 0, 0] \). If the item picking plan is based on the score function mentioned above \( \text{ScoreValue}_{i,k}(c) = \frac{p_k}{w_k \times \sum d_{i,j}} \), then the score values of the items are \( s_1 = 1.01, s_2 = 0.8, s_3 = 1, \) and \( s_4 = 1 \). High-scoring items are selected first, where the current total weight \( W_c \) of the selected items must not exceed the maximum capacity \( W \). In the subset of the tour 1-2, no item is picked up; hence the thief travels with the maximum velocity \( v_{\text{max}} = 1 \). From City 2, Item 1 is chosen, which changes the current speed \( v_c \) to 0.1, and the knapsack is full. The optimal objective value is \( G(c, z) = 101 - 1 \times (1 + \frac{10}{0.1}) = 0 \), assuming that the tour is fixed and the picking plan \( z' \) is \( (0, 1, 1, 1) \). Travel time from Cities 1 to 3 is \( t = 1 + 5 = 6 \). From City 3, Item 2 is selected, resulting in \( v_c = 0.82 \), and \( W_c = 2 \). Thus, the travel time from Cities 3 to 4 is \( t = \frac{3.66}{2} = 1.83 \). From City 4, Item 3 is picked resulting in \( v_c = 0.46 \), and \( W_c = 6 \). The travel time from Cities 4 to 5 is \( t = \frac{2.17}{2} = 1.085 \). Item 4 is selected from City 5, which makes \( v_c = 0.28 \), and \( W_c = 8 \). The travel time from Cities 5 to 1 is \( t = \frac{3.57}{3} = 1.19 \). Therefore, \( T(c, z') = 6 + 3.66 + 2.17 + 3.57 = 15.40 \), and the objective value \( G(x, z') = 18 - 1 \times (15.40) = 2.60 \).

A. EXPAND THE EFFECT OF THE ITEM WEIGHT

Based on the above inspiration, we propose a novel approach to selecting items, and choose potential items in reverse order,
B. REVERSE-ORDER SEARCHING APPROACH

The constructing process starts by calculating \( Score_{i,k} \) for each item according to the formula introduced in Equation 10, where the items are sorted according to the non-decreasing order of \( Score_{i,k} \). We discard all items with \( Score_{i,k} < 0 \) and keep the items with a \( Score_{i,k} > 0 \). We suppose a total of \( l \) items \( (l \leq m) \) exist. We calculate the number of picked items \( l \) to the total number of items \( m \) as \( r \) \( (r = \frac{l}{m}) \). We take the average value of all items with a \( Score_{i,k} > 0 \) as \( AVG_{score} \). The function can be summarized as follows:

\[
AVG_{score}(c) = \frac{\sum_{i=1}^{l} Score_{i,k}}{l}.
\]  

For every item \( k \), if \( Score_{i,k} > AVG_{score} \), then item \( k \) is a potential selection. In city \( i \), the potential items are denoted as set \( s_i \), where \( s_j \) contains zero, or more items. The high-value items are selected in reverse order along the given tour. If inserting item \( k \) does not decrease the objective value and fits into the knapsack, then items \( k \) is selected; otherwise, the next item is processed and so on. For all items on the entire route, we mark the picked items as \( Above_{AVG}(s_1, \ldots, s_q) \). Then, the items are selected in reverse order until the items reach the knapsack capacity \( W \). The proposed approach is RWS (based on item selection weight and reverse-order allocation). The illustrative diagram of the proposed method is displayed in Fig. 2 to facilitate following our algorithm for readers. In the entire travel tour, high-value, low-weight items in cities near the end (green items in the picture) are more likely to be selected than high-value, high-weight items in cities near the beginning of the tour (red items in the picture).

Algorithm 1 describes the basic framework for solving the TTP. Based on the above ideas, the initial picking plan is introduced in Algorithm 2. The idea for Algorithm 2 can be explained as follows. First, a new initial cyclic tour is generated using the LKH. The priorities of the items are determined using the formula in Equation 10. The higher the score of the items, the higher the priority of the item being selected. Then, items with positive scores are selected (those with negative scores do not contribute to the total profit), and their average \( AVG_{score} \) and maximum \( MAX_{score} \) scores are calculated. Afterward, the items with a score greater than the average are selected in the reverse order of the cities in the travel tour. The capacity constraint is imposed as a global constraint. Any insertion that results in the violation of the capacity constraint is prohibited. The insertion heuristic is based on Mei et al. [16] in Algorithm 2. Finally, the picking plan \( z^* = z \) is restored. The \( TSPSolver \) adopts 2-opt heuristic search to optimize the TSP component. Furthermore, the Delaunay triangulation is introduced as a candidate generator for the 2-opt heuristic. In the \( KP Solver \), both the bit-flip operator and the simulated annealing metaheuristic are commonly used algorithms for the KP component. We ran some instances and found that the simulated annealing algorithm performs well on large-scale instances. In this article, the simulated annealing algorithm is applied to solve the KP component.

\begin{algorithm}
\caption{Algorithm Framework}
\begin{algorithmic}[1]
  \State (\( c^*, z^* \)) $\leftarrow \emptyset$ \COMMENT{best solution}
  \State Set the current picking plan \( z = \emptyset \) and current weight \( W_c = 0 \)
  \State Set current tour \( c = \emptyset \), and calculate \( G(c, z) \)
  \While{not global timeout}
    \State \( c \leftarrow LKTour() \)
    \State \( z \leftarrow Init\ Picking\ Plan(c) \)
    \State \( (c, z) \leftarrow TSPSolver(c, z) \)
    \State \( (c, z) \leftarrow KP\ Solver(c, z) \)
    \If{\( G(c, z) > G(c^*, z^*) \)}
      \State \( (c^*, z^*) \leftarrow (c, z) \)
    \EndIf
  \EndWhile
  \State \textbf{return} \( (c^*, z^*) \)
\end{algorithmic}
\end{algorithm}

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{algorithm1.pdf}
\caption{Illustrative diagram of the proposed method.}
\end{figure}

Algorithm 1 describes the basic framework for solving the TTP. Based on the above ideas, the initial picking plan is introduced in Algorithm 2. The idea for Algorithm 2 can be explained as follows. First, a new initial cyclic tour is generated using the LKH. The priorities of the items are determined using the formula in Equation 10. The higher the score of the items, the higher the priority of the item being selected. Then, items with positive scores are selected (those with negative scores do not contribute to the total profit), and their average \( AVG_{score} \) and maximum \( MAX_{score} \) scores are calculated. Afterward, the items with a score greater than the average are selected in the reverse order of the cities in the travel tour. The capacity constraint is imposed as a global constraint. Any insertion that results in the violation of the capacity constraint is prohibited. The insertion heuristic is based on Mei et al. [16] in Algorithm 2. Finally, the picking plan \( z^* = z \) is restored. The \( TSPSolver \) adopts 2-opt heuristic search to optimize the TSP component. Furthermore, the Delaunay triangulation is introduced as a candidate generator for the 2-opt heuristic. In the \( KP Solver \), both the bit-flip operator and the simulated annealing metaheuristic are commonly used algorithms for the KP component. We ran some instances and found that the simulated annealing algorithm performs well on large-scale instances. In this article, the simulated annealing algorithm is applied to solve the KP component.
Algorithm 2 Initial Picking Plan
1: Compute the score of the each item $I_k \in m$ by proposed formulation for the given tour $c$
2: Sorting the items of $m$ in descending order according to their score value
3: Calculate the value of items greater than 0 and calculate their average $AVG_{score}$ and max value $MAX_{score}$
4: Set current packing plan $z = \emptyset$ and current weight of knapsack $W_c = 0$
5: Set $\beta \in [0, 1]$, which is set according to the size of the instance
6: while $W_c < W$ do
7: for $i \leftarrow n$ To 2 do
8: if $Score_{i,k} > AVG_{score} + (MAX_{score} - AVG_{score}) \times \beta$ then
9: add item $I_k$ to the picking plan $z = z \cup \{I_k\}$
10: set $W_c = W_c + w_k$
11: end if
12: if $W \times r \leq W_c$ then
13: Break
14: end if
15: end for
16: for $j \leftarrow 1$ To $m$ do
17: if $W \geq W_c$ then
18: Insertion heuristic
19: else
20: Break
21: end if
22: end for
23: if $G(c, z) > G(c, z^*)$ then
24: $z^* \leftarrow z$
25: end if
26: end while
27: return $z^*$

IV. EXPERIMENTAL STUDY
In this section, the experimental setup of the TTP is described. In addition, the comparative results are investigated with other state-of-the-art approaches.

A. BENCHMARK INSTANCES AND EXPERIMENTAL SETUP
We use a comprehensive set of TTP instances that from Polyakovskiy et al. [22] for our investigations. The two components of the TTP are balanced in these instances so that the nearly optimal solution of one subproblem does not guarantee the optimal solution of another subproblem.

The TTP dataset introduces the following diversification parameters resulting in 9720 TTP instances, and these instances are typically based on instances from TSPLIB by Reinelt [23] and the types of the knapsacks introduced by Martello et al. [14]. We consider a subset of the TTP library to perform our tests: eil76, kroA100, ch130, u159, a280, u574, u724, dsj1000, rl1304, fl1577, d2103, pcb3038, fnl4461, pla7397, rl11849, usa13509, brd14051, d15112, d18512, and pla33810.

These cover small, medium, and large instances with different characteristics. We denote the four categories as A, B, C, and D (Categories C and D have the same KP type and item factor):
- Category A: one item in each city, where item values and weights are bounded and strongly correlated, and the knapsack has a small capacity.
- Category B: Five items in each city, uncorrelated with the KP but with similar item weights, and the knapsack has an average capacity.
- Category C: Ten items in each city, uncorrelated with the KP, and the knapsack has a high capacity.
- Category D: Nine items in each city, uncorrelated with the KP, and the knapsack has a high capacity.

According to the different data types A, B, C, and D, $\beta$ (mentioned in Algorithm 2) is 1, 0.65, 0.5, 0.5, respectively. The experiment setup is adopted in all experiments. All solvers are run on each TTP instance 10 times independently, and all algorithms have a maximum runtime limit of 600 s. All experiments were performed on a computer with an Intel Core i5-8500 CPU (3.00 GHz).

B. COMPARISON OF ALGORITHMS
To gain further insight into the performance of each solver for solving the TTP, we performed a statistical analysis of Friedman’s test [26] for all methods. This analysis is an alternative to the repeated-measure one-way analysis of variance [5]. It is a non-parametric test. When the dependent variable is ordinal, this test is used to determine the difference between groups and is used for continuous data. The relative standard deviation (RSD) is introduced to measure the consistency of the results. The formula is defined as $RSD = \frac{\bar{x}}{\bar{x}} \times 100\%$, where $\bar{x}$ is the mean of the standard deviation, and $\bar{x}$ is the arithmetic mean. For quality measures of all methods, we adopted the average ranking and Friedman’s test ranking and calculated the ranking of each method based on the target value of each TTP instance for the average ranking method. Then, we computed the average ranking for each method.

In Friedman’s test ranking, the formula for the test statistic is defined as $F = \frac{12}{n(k+1)} \sum r^2 - 3n(k+1)$, where $n$ denotes the number of instances, and $k$ is the number of methods. First, we calculated the ranking of each method for each instance, then computed the sum of the rank ($r$) of each method. Then, the probability value ($p$) and the degrees of freedom ($d$) were applied to calculate the critical chi-square value. The null hypothesis is rejected if the $F$ value is greater than the critical chi-square value. Finally, we calculated the average rank of each method. The size of Friedman’s test value can measure the quality of the various listed algorithms.

The comparison study results between the proposed method and three other state-of-the-art algorithms are shown in Tables 1-3. For each instance, 10 independent runs were performed. The best mean objective values are highlighted in bold, and the mean objective value is
regarded as the quality of the solution to compare the algorithm performance. Friedman’s test-based ranking results for each method are presented in the last row of each table.

As described in Section 3, we argue that the item weight has a greater influence on the final profit than other item attributes (value, location, etc.). The variant in Equation 10 is configured as follows to verify our speculation:

- Solver 1: The value of exponent $\alpha$ is set to 1.5.
- Solver 2: The value of exponent $\alpha$ is set to 1.

The results in Table 4 indicate that Solver 1 performs better in many TTP instances. This result verifies that our reasoning is plausible.

### C. RESULT ANALYSIS AND DISCUSSION

According to the presented results, the proposed algorithm surpasses other state-of-the-art algorithms (e.g., MATLS [16], S5 [10], and CS2SA* [9]) for many instances of the TTP. The proposed algorithm adopts a reverse-order picking plan based on sorting the items according to the profits, weights, and locations of the items in the given tour, selecting items with a score greater than the average. The algorithm uses a different travel tour instead of a fixed one in a given time budget to avoid falling into a local optimum. The algorithm exhibits some competitiveness in Category A, and even the profits and weights of the items are strongly correlated, as listed in Table 1. The presented results demonstrate that

### TABLE 1. Results for category A.

| Instance | MATLS | SS | CS2SA* | RWS |
|----------|-------|----|--------|-----|
| elf76    | 22185(1) | 0 | 0 | 0 |
| kroA100  | 42535(1) | 1.45 | 3940(3) | 0 |
| ch130    | 61028(1) | 0.12 | 58685(2) | 1.21 |
| u159     | 58289(2) | 1.06 | 57618(4) | 0 |
| a280     | 110132(1) | 2.16 | 109921(3) | 0.4 |
| a574     | 254770(1) | 0.76 | 251775(2) | 0.02 |
| u724     | 303435(4) | 1.17 | 305977(2) | 0.32 |
| d1000    | 340317(2) | 1.55 | 342189(1) | 0.59 |
| r1304    | 572766(4) | 1.2 | 575102(3) | 0.85 |
| f1577    | 609283(3) | 1.77 | 607247(4) | 1.62 |
| d2103    | 849625(2) | 1.35 | 853857(1) | 1.2 |
| pch3038  | 116810(4) | 0.82 | 117951(2) | 0.16 |
| fn4461   | 1617401(4) | 0.3 | 1625856(2) | 0.16 |
| pla7397  | 4178551(2) | 3.25 | 4371433(1) | 0.82 |
| r11849   | 4587812(4) | 0.48 | 4630753(3) | 0.29 |
| usa13509 | 7767305(4) | 2.1 | 781185(3) | 0.86 |
| bnd14051 | 6492925(4) | 1.25 | 6552568(3) | 0.58 |
| d2112    | 682852(4) | 2.3 | 6991416(3) | 1.21 |
| d13512   | 7164979(4) | 1.25 | 7257669(3) | 0.81 |
| pla53810 | 15232942(4) | 1.5 | 15454559(3) | 0.74 |

### TABLE 2. Results for category B.

| Instance | MATLS | SS | CS2SA* | RWS |
|----------|-------|----|--------|-----|
| elf76    | 3705(3) | 1.35 | 3742(2) | 0 |
| kroA100  | 4660(1) | 1.36 | 42834(1) | 0 |
| ch130    | 8876(4) | 0.79 | 9250(1) | 0 |
| u159     | 8403(4) | 1.40 | 86031(1) | 0 |
| a280     | 17678(4) | 0.54 | 180461(1) | 0.01 |
| a574     | 26212(3) | 2.30 | 26957(1) | 0.10 |
| v274     | 489503(2) | 1.25 | 50313(1) | 0.12 |
| d11000   | 143699(2) | 0 | 137653(4) | 0.16 |
| r11304   | 75800(3) | 1.26 | 80067(1) | 0.86 |
| f11577   | 88375(3) | 0.41 | 92328(1) | 1.25 |
| d2103    | 113005(4) | 0.45 | 120482(1) | 0.2 |
| pch3038  | 148265(1) | 1.18 | 16006(1) | 0.15 |
| fn4461   | 247553(2) | 0.40 | 262237(1) | 0.11 |
| pla7397  | 365613(2) | 1.32 | 395156(1) | 0.56 |
| r11849   | 661392(2) | 0.29 | 707183(1) | 0.24 |
| usa13509 | 747885(2) | 0.53 | 809623(1) | 0.35 |
| brd14051 | 815602(2) | 0.36 | 875908(1) | 0.25 |
| d15112   | 871313(2) | 0.52 | 939726(1) | 0.48 |
| d18512   | 996582(2) | 0.84 | 1072308(1) | 0.21 |
| pla53810 | 1730352(4) | 0.92 | 187030(1) | 0.62 |

Average ranking 2.8 2.5 2.45 2.25
S5 surpasses the other algorithms in most instances. Category A has the smallest knapsack capacity and only one item in each city. We argue that the greedy approach adopted by S5 is beneficial in solving this type of KP component of the TTP.

For the instances with a higher knapsack capacity in Category B (five items in each city, where KP is uncorrelated with similar weights), the comparative results suggest that the CS2SA* and S5 are still competitive in this type of category. However, Table 2 indicates that the RWS outperforms other heuristics for most instances, such as u159, rl1304, rl11849, brd14051, and pla33810. Moreover, MATLS and CS2SA* also perform better in some instances, which are illustrated in table 2.

Table 3 reveals the comparative results for Category C (10 items per city, uncorrelated). This category has the largest knapsack capacity. The CS2SA* performs better in many instances compared to other algorithms in the table. The RWS outperforms other heuristics in most instances for high knapsack capacities such as kroA100, u159, u724, and pla33810. Especially in pla33810, the result value (58818293) obtained by the RWS algorithm is significantly better than others.

Table 4 was applied to determine the differences between the groups when the dependent variable is ordinal and obtain a better performance analysis of all algorithms. The Nemenyi post-hoc test was employed after the Friedman test was applied. The test ranking of all the algorithms is

**TABLE 3. Results for category C.**

| Instance | MATLS | SS | CS2SA* | RWS |
|----------|-------|----|--------|-----|
| Mean     | RSD   | Mean | RSD   | Mean | RSD   |
| eil76    | 881151 | 0.32 | 856644 | 0    | 877773 | 0.27 |
| kroA100  | 155492 | 0.01 | 155503 | 0    | 155585 | 0.48 |
| ch130    | 203468 | 2.13 | 201085 | 0.82 | 197555 | 0.27 |
| u159     | 242558 | 0.45 | 242483 | 0.31 | 242204 | 0.52 |
| a280     | 426293 | 0.2   | 429000 | 0    | 421713 | 0.14 |
| u574     | 966207 | 0.24 | 966344 | 0.11 | 953979 | 0.16 |
| u724     | 1188670 | 0.45 | 118364 | 0.08 | 1191819 | 0.32 |
| df1300   | 1472612 | 1.2   | 1479605 | 0.24 | 1468858 | 0.05 |
| r1304    | 2178475 | 0.21 | 2184853 | 0.33 | 2198943 | 0.16 |
| fl1577   | 2466353 | 0.26 | 2470917 | 0.21 | 2505291 | 0.25 |
| d2103    | 3392866 | 0.32 | 3392172 | 0.26 | 3373781 | 0.93 |
| pcb3038  | 4564228 | 0.22 | 4573748 | 0.15 | 4612956 | 0.01 |
| fn14461  | 6534424 | 0.17  | 6544971 | 0.26 | 6545335 | 0.16 |
| pla7039  | 13785791 | 1.55 | 14239606 | 1.2 | 13977364 | 0.35 |
| r11849   | 18275216 | 0.23 | 18316650 | 0.12 | 18502520 | 0.99 |
| uasa15509 | 25878184 | 0.44 | 25918971 | 0.55 | 26437362 | 0.11 |
| brd14051 | 23672405 | 0.62 | 23826398 | 0.51 | 23908540 | 0.01 |
| d15112   | 25942410 | 1.52 | 26211252 | 1.04 | 27182609 | 0.13 |
| d18512   | 27164388 | 1.25 | 27427144 | 0.32 | 27849746 | 0.21 |
| pla33810 | 58003895 | 0.5  | 57967586 | 0.42 | 58107003 | 0.21 |

**TABLE 4. Performance comparison of two solvers on three categories of TTP instances.**

| Instance | Category A | Category B | Category C |
|----------|------------|------------|------------|
|          | Solver1 | Solver2 | Solver1 | Solver2 | Solver1 | Solver2 |
| eil76    | 3765 | 3670 | 21620 | 20192 | 87664 | 87599 |
| kroA100  | 4445 | 4424 | 41238 | 41353 | 155947 | 155669 |
| ch130    | 9013 | 8963 | 57964 | 58792 | 202348 | 202182 |
| u159     | 8627 | 8566 | 58966 | 58955 | 244770 | 244228 |
| a280     | 17723 | 17723 | 107874 | 108378 | 426736 | 424358 |
| u574     | 26366 | 26266 | 247992 | 249638 | 955745 | 953998 |
| u724     | 48794 | 49588 | 304420 | 309750 | 1193604 | 1191819 |
| df1300   | 141117 | 140620 | 359557 | 338661 | 1460926 | 1468839 |
| r1304    | 75206 | 76435 | 585103 | 585600 | 2198947 | 2198942 |
| fl1577   | 85923 | 82848 | 635112 | 636424 | 2655297 | 2655294 |
| d2103    | 118338 | 118652 | 842596 | 845222 | 3410978 | 3393849 |
| pcb3038  | 149337 | 146115 | 1176520 | 1197373 | 4612966 | 4612957 |
| fn14461  | 241291 | 240822 | 1624685 | 1628417 | 6545620 | 6545346 |
| pla7039  | 315386 | 310473 | 3751665 | 3731312 | 13444071 | 13197751 |
| r11849   | 653857 | 659283 | 4729274 | 4710149 | 18422410 | 18504597 |
| uasa15509 | 679783 | 682230 | 8022398 | 8115207 | 26552971 | 26552972 |
| brd14051 | 798787 | 801888 | 6778329 | 6654177 | 23809751 | 23907953 |
| d15112   | 686019 | 686998 | 7060136 | 7060876 | 27184251 | 27182054 |
| d18512   | 962781 | 964518 | 7507146 | 7580272 | 27980877 | 27861162 |
| pla33810 | 1781384 | 1777592 | 15821323 | 15745060 | 58818293 | 58542757 |
presented in Fig. 3, and the Friedman test rankings of the four approaches in Categories A, B, and C are presented. The MATLS approach performs poorly primarily because population-based heuristics for the TTP are not efficient for handling large-scale instances. Most single solution heuristic over an evolutionary algorithm for the TTP is that needs to initialize the tour only once. In large-scale instances (in Category C), our proposed algorithm performs better than the other algorithms. Our algorithm prioritizes the selection of city items near the end of the tour and considers the accumulated weight of the items in the backpack during initialization, avoiding selecting items with a small profit. Thus, the final obtain profit is higher. The premature consideration of item weight’s cumulative effect and the reverse selection of items reduce the TTP solution’s search space, resulting in our algorithm not being prominent in small-scale examples.

In addition, to verify our supposition in Section 3, the item weight has a greater effect on the final profit than the other item attributes (value, location, etc.) due to the cumulative effect of the selected item weight. The results of the experiments we performed are listed in Table 4. The value of exponent $\alpha$ is used to manage the effect of the item weight on the final profit. Solver 1 outperforms the Solver 2 in many instances (especially in Category C). A representative excerpt of the results is presented in Fig. 4, 5, and 6. We rescaled the achieved objective values into the range $[0, 1]$ using the normalization method. The normalization function is $x_{\text{normalization}} = \frac{x - \text{Min}}{\text{Max} - \text{Min}}$, where Max and Min denotes the maximum and minimum values in the vector, respectively. The box diagram on the right side of the figure displays the algorithm performance with the different values of parameter $\alpha$. The results indicate that the performance of Solver 1 has no obvious superiority in small-scale (Category A) and medium-scale (Category B) instances. However, in large-scale instances (Category C), the performance of the Solver 1 algorithm has greater advantages. In Fig. 7 and 8,
we draw the convergence diagrams of the four methods in some instances of Category B and Category C, respectively. Our method has high convergence in experiments. Some of our algorithms in the convergence graph will further increase in a midway position, which shows that our algorithm can jump out of the local optimal solution for exploration. For further verification, we compared two large-scale instances of experiments in Categories C (item factor: 10) and D (item factor: 9) in Fig. 9. This experimental result also demonstrates the superior performance of Solver 1. The results of this investigation verify our supposition that: the item weight has a greater influence on the final profit in large-scale instances.

Therefore, from the result, we conclude that our proposed algorithm performs better than the other state-of-the-art algorithms in most instances, especially for Category B and C. The proposed algorithm adopts a reverse-order picking plan, based on sorting the items according to the proposed formula. It expands the search space and is likely to determine potential solutions for solving the TTP.
V. CONCLUSION

In real-world optimization problems, combinatorial optimization problems with two or more interdependent components have significant role. Due to interdependency, an optimal solution to one of the components does not guarantee an overall optimal solution to the whole problem. The TTP can be thought of as a combination of two well-known interdependent problems: the TSP and the KP, which were introduced to represent real-world applications. The interaction and interdependence between the subproblems indicate the complexity of the whole problem.

In this paper, an approach for solving the TTP based on item selection weight and reverse-order allocation is proposed. This paper introduces an EM formula to calculate the score value of each item for item selection, which expands effect of the item’s weight. The proposed approach also adopts the reverse-order allocation, which selects items in an inverse order according to the traveling route. Due to the cumulative effect of the weight of the picked item, we suppose that the item weight has a greater influence on the final profit than other item attributes (value, location, etc.). To address the issue, we proposed a new heuristic for the TTP based on managing the effect of the item weight on the final profits. Moreover, we believe that high-value, low-weight items near the end of the travel route should be selected. Under the condition that the total selected item weight is not heavier than the knapsack capacity, the items are chosen from back to front according to the route. Thus, we proposed a method of choosing items in reverse order. The results reveal that our approach is competitive for many instances of various sizes and properties compared to other heuristics.

Most real-world combinatorial optimization problems have more than two components. Further research will be conducted to investigate the bi-Objective TTP and other problems with more than two components to obtain internal dependencies in the future. Besides, the method we proposed can be further improved in space exploration and can be adopted in problems with many interacting components with immense potential in real-world applications. Furthermore, some of the representative computational intelligence algorithms will be used to solve the TTP in future work, like monarch butterfly optimization (MBO), earthworm optimization algorithm (EWA), elephant herding optimization (EHO), moth search (MS) algorithm, Slime mould algorithm (SMA), and Harris hawks optimization (HHO).

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their helpful comments and suggestions that helped improve the presentation.

REFERENCES

[1] S. T. Alharbi, “The design and development of a modified artificial bee colony approach for the traveling thief problem,” Int. J. Appl. Evol. Comput., vol. 9, no. 3, pp. 32–47, Jul. 2018.

[2] M. R. Bonyadi, Z. Michalewicz, and L. Barone, “The travelling thief problem: The first step in the transition from theoretical problems to realistic problems,” in Proc. IEEE Conge. Evol. Comput., Jun. 2013, pp. 1037–1044.

[3] M. R. Bonyadi, Z. Michalewicz, M. Wagner, and F. Neumann, “Evolutionary computation for multicomponent problems: Opportunities and future directions,” in Optimization in Industry. Cham, Switzerland: Springer, 2019, pp. 13–30.

[4] M. R. Bonyadi, Z. Michalewicz, M. R. Przybylak, and A. Wierzbicki, “Socially inspired algorithms for the travelling thief problem,” in Proc. Annu. Conf. Genetic Evol. Comput., Jul. 2014, pp. 421–428.

[5] A. Cuevas, M. Febbrero, and R. Fraiman, “An anova test for functional data,” Comput. Statist. Data Anal., vol. 47, no. 1, pp. 111–122, Aug. 2004.

[6] B. Delaunay, “Sur la sphere vide,” Izv. Akad. Nauk, Otdelenie Matematicheski Estestvennyska Nauk, vol. 7, nos. 1–2, pp. 793–800, 1934.

[7] M. El Yafrani and B. Ahiod, “Population-based vs. single-solution heuristics for the travelling thief problem,” in Proc. Genetic Evol. Comput. Conf., Jul. 2016, pp. 317–324.

[8] M. El Yafrani and B. Ahiod, “A local search based approach for solving the travelling thief problem: The pros and cons,” Appl. Soft Comput., vol. 52, pp. 795–804, Mar. 2017.

[9] M. El Yafrani and B. Ahiod, “Efficiently solving the travelling thief problem using hill climbing and simulated annealing,” Int. Sci., vol. 432, pp. 231–244, Mar. 2018.

[10] A. Maity and S. Das, “Efficient hybrid local search heuristics for solving the travelling thief problem,” Appl. Soft Comput., vol. 93, Aug. 2020, Art. no. 106284.

[11] A. Cuevas, M. Febrero, and R. Fraiman, “An anova test for functional data,” Int. J. Appl. Evol. Comput., vol. 9, no. 3, pp. 32–47, Jul. 2018.

[12] M. S. R. Martins, M. El Yafrani, M. R. B. S. Delgado, M. Wagner, B. Ahiod, and R. Laiders, “HSEDA: A heuristic selection approach based on estimation of distribution algorithm for the travelling thief problem,” in Proc. Genetic Evol. Comput. Conf., Jul. 2017, pp. 361–368.

[13] M. El Yafrani and B. Ahiod, “On investigation of interdependence between sub-problems of the travelling thief problem,” Soft Comput., vol. 20, no. 1, pp. 157–172, Jan. 2016.

[14] Z. Michalewicz, “Quo vadis, evolutionary computation?” in Proc. IEEE World Congr. Comput. Intell. Berlin, Germany: Springer, 2012, pp. 98–121.

[15] S. Lin and B. W. Kernighan, “An effective heuristic algorithm for the traveling-salesman problem,” Oper. Res., vol. 21, no. 2, pp. 498–516, Apr. 1973.

[16] Y. Mei, X. Li, and X. Yao, “Improving efficiency of heuristics for the large scale traveling thief problem,” in Proc. Soft Comput., vol. 93, Aug. 2020, Art. no. 106284.

[17] M. El Yafrani and B. Ahiod, “Population-based vs. single-solution heuristics for the travelling thief problem,” in Proc. Asia–Pacific Conf. Simulated Evol. Learn. Cham, Switzerland: Springer, 2014, pp. 631–643.

[18] M. El Yafrani and B. Ahiod, “On investigation of interdependence between sub-problems of the travelling thief problem,” Soft Comput., vol. 20, no. 1, pp. 157–172, Jan. 2016.

[19] Z. Michalewicz, “Quo vadis, evolutionary computation?” in Proc. IEEE World Congr. Comput. Intell. Berlin, Germany: Springer, 2012, pp. 98–121.

[20] M. Namazi, “A profit guided coordination heuristic for travelling thief problems,” in Proc. SOCS, 2019, pp. 140–144.

[21] M. El Yafrani, C. Sanderson, M. A. Hakim Newton, and A. Sattar, “A cooperative coordination solver for travelling thief problems,” 2019, arXiv:1911.03124. [Online]. Available: http://arxiv.org/abs/1911.03124

[22] F. Neumann, “A fully polynomial time approximation scheme for packing while traveling,” in Proc. Int. Symp. Algorithmic Aspects Cloud Comput. Cham, Switzerland: Springer, 2018, pp. 59–72.

[23] S. Polyanovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann, “A comprehensive benchmark set and heuristics for the travelling thief problem,” in Proc. Annu. Conf. Genetic Evol. Comput., Jul. 2014, pp. 477–484.

[24] G. Reintelt, “TSPLIB—A traveling salesman problem library,” ORSA J. Comput., vol. 3, no. 4, pp. 376–384, 1991.

[25] M. Wagner, “Stealing items more efficiently with ants: A swarm intelligence approach to the travelling thief problem,” in Proc. Int. Conf. Swarm Intell. Cham, Switzerland: Springer, 2016, pp. 273–281.

[26] J. Wu, “Exact approaches for the travelling thief problem,” in Proc. Asia–Pacific Conf. Simulated Evol. Learn. Cham, Switzerland: Springer, 2017, pp. 110–121.

[27] D. W. Zimmerman and B. D. Zumbo, “Relative power of the wilcoxon test,” J. Educ. Statist., vol. 62, no. 1, pp. 75–86, Jul. 1993.