Improving Efficiency of Heuristics for the Large Scale Traveling Thief Problem

Yi Mei[1], *Member, IEEE*, Xiaodong Li[1], *Senior Member, IEEE*, and Xin Yao[2], *Fellow, IEEE*

[1] School of Computer Science and Information Technology, RMIT University, Australia (yi.mei@rmit.edu.au, xiaodong.li@rmit.edu.au)
[2] School of Computer Science, University of Birmingham, UK (x.yao@cs.bham.ac.uk)

Abstract. The Traveling Thief Problem (TTP) is a novel problem that combines the well-known Traveling Salesman Problem (TSP) and Knapsack Problem (KP). In this paper, the complexity of the local-search-based heuristics for solving TTP is analyzed, and complexity reduction strategies for TTP are proposed to speed up the heuristics. Then, a two-stage local search process with fitness approximation schemes is designed to further improve the efficiency of heuristics. Finally, an efficient Memetic Algorithm (MA) with the two-stage local search is proposed to solve the large scale TTP. The experimental results on the tested large scale TTP benchmark instances showed that the proposed MA can obtain competitive results within a very short time frame for the large scale TTP. This suggests the potential benefits of designing intelligent divide-and-conquer strategies that solves the sub-problems separately while taking the interdependence between them into account.

1 Introduction

The Traveling Thief Problem (TTP) is a novel composite problem proposed by Bonyadi *et al.* [1] in order to investigate the effect of interdependence between sub-problems in a complicated problem. TTP is a combination of two well-known combinatorial optimization problems, i.e., the Travelling Salesman Problem (TSP) and Knapsack Problem (KP). Specifically, a thief is to visit a set of cities and pick some items from the cities to put in a rented knapsack. Each item has a value and a weight. The knapsack has a limited capacity that cannot be exceeded by the total weight of the picked items. In the end, the thief has to pay the rent for the knapsack, which depends on the travel time. TTP aims to find a tour for the thief to visit all the cities exactly once, pick some items along the way and finally return to the starting city, so that the profit of the visit, which is the total value of the picked items minus the rent of the knapsack, is maximized. Since the TSP and KP have been intensively investigated, the TTP facilitates studies on analysing the interdependence between sub-problems.

A potential real-world applications of TTP is the capacitated arc routing problem with service profit, where each customer has a demand and a service

profit, and the travel cost (e.g., the petrol consumption) depends on the load of the vehicle. This novel form of problem is more practical and closer to reality than the models that have been studied intensively [2] [3] [4] [5].

In this paper, the challenges caused by the interdependence between the TSP and KP components are discussed. In particular, the computational complexities of the search-based heuristics are analyzed in the context of TTP and compared with that for TSP and KP, respectively. It is found that the combination of TSP and KP in TTP can lead to a much higher computational complexity of the search. Such issue becomes much more severe for large scale instances with thousands or tens of thousands of cities and items.

To address the computational complexity issue and to speed up the algorithm, a two-stage local search with novel fitness approximation schemes for TTP is proposed, and embedded into a Memetic Algorithm (MA) framework to solve the large scale TTP. The resultant MA is tested on the CEC'2014 TTP competition benchmarks and the results showed that it managed to achieve competitive solution quality with a limited computational budget.

The rest of the paper is as follows: In Section 2, TTP is described in details. In Section 3, the computational complexities of the commonly used heuristics such as 2-opt operator for TSP and single flip for KP are analyzed in the context of TTP. Then, the complexity reduction strategies for TTP are derived from that for TSP and KP in Section 4. The proposed two-stage local search that further improves the efficiency is described in Section 5.

2 Travelling Thief Problem

The TTP is a combination of TSP and KP. In TSP, n cities with the distance matrix of $D_{n\times n}$ are given, where $d(i,j)$ is the distance from city i to j. In KP, there are m items. Each item i has a weight w_i, a value b_i and an available city a_i. A thief aims to visit all the cities exactly once, pick items on the way and finally come back to the starting city. The thief rents a knapsack to carry the items, which has a capacity of Q. The rent of the knapsack is R per time unit. The speed of the thief decreases linearly with the increase of the total weight of carried items, and is computed by the following formula:

$$v = v_{\max} - \nu\bar{w}, \tag{1}$$

$$\nu = \frac{v_{\max} - v_{\min}}{Q}, \tag{2}$$

where $0 \leq \bar{w} \leq Q$ is the current total weight of the picked items. When the knapsack is empty ($\bar{w} = 0$), the speed is maximized ($v = v_{\max}$). When the knapsack is full ($\bar{w} = Q$), the speed is minimized ($v = v_{\min}$). Then, the profit gained by the thief is defined as the total value of the picked items minus the rent of the knapsack. Let a TTP solution be represented by a tour $\mathbf{x}$ and a picking plan $\mathbf{z}$, where $\mathbf{x} = (x_1, \ldots, x_n)$ is a permutation of the cities and $\mathbf{z} = (z_1, \ldots, z_m)$

is a 0-1 vector of the items. z_i takes 1 if item i is picked, and 0 otherwise. Then, the TTP problem can be stated as follows:

$$\max \ \sum_{j=1}^{m} b_j z_j - R \cdot \left(\sum_{i=1}^{n-1} \frac{d(x_i, x_{i+1})}{v(i)} + \frac{d(x_n, x_1)}{v(n)} \right), \tag{3}$$

$$s.t.: \ v(i) = v_{\max} - \nu \bar{w}(i), \ \ 1 \le i \le n, \tag{4}$$

$$\bar{w}(i) = \bar{w}(i-1) + cw(x_i), \ \ 1 \le i \le n, \tag{5}$$

$$cw(i) = \sum_{j=1}^{m} w_j z_j [a_j = i], \ \ 1 \le i \le n, \tag{6}$$

$$x_i \ne x_j, \ \ 1 \le i \ne j \le n, \tag{7}$$

$$\sum_{j=1}^{m} w_j z_j \le Q, \tag{8}$$

$$x_i \in \{1, \ldots, n\}, z_j \in \{0, 1\}. \tag{9}$$

Eq. (3) is the objective function, which is to maximize the profit. $v(i)$ is the velocity at x_i, and is calculated by Eq. (4). $\bar{w}(i)$ is the accumulated weight from the beginning of the tour to x_i. It is computed by Eq. (5), where $cw(x_i)$ indicates the total weight of the items picked at x_i, and is obtained by Eq. (6). $[a_j = i]$ takes 1 if $a_j = i$, and 0 otherwise. Eqs. (7) and (9) ensures that $\mathbf{x}$ is a valid tour, i.e., each city appears exactly once in the permutation. Eq. (8) indicates that the total weight of the picked items cannot exceed the capacity.

Fig. 1 illustrates an example of a TTP solution that travels through the path A-B-C-D-A, picking items 1 and 2 at cities A and C, respectively. The weights and values of the items are $w_1 = b_1 = 2$ and $w_2 = b_2 = 1$. The numbers associated with the arcs indicate the distances between the cities. The total value of the picked items is $b_1 + b_2 = 3$. The travel speeds between each pair of cities are $v_{AB} = v_{BC} = 4 - 3 \cdot 2/3 = 2$ and $v_{CD} = v_{DA} = 1$. Then, the total travel time is $(2 + 1.5)/2 + (1 + 1.5)/1 = 4.25$. Finally, the profit of the travel is $3 - 1 \cdot 4.25 = -1.25$ (a loss of 1.25).

3 Complexity of Heuristics in TTP

It is obvious that TTP is NP-hard, as it can be reduced to TSP when there are no items. Therefore, the exact methods are not applicable in practice, where the problem has a large size. The heuristics are commonly used for they can provide near-optimal solutions in a short time.

The heuristics can be categorized into two types. The former includes the so-called constructive heuristics that construct a solution based on domain knowledge. For example, the Christofides's heuristic [6] is the best-so-far TSP heuristic that can construct tours whose lengths are no larger than 1.5 times of the optimum. The latter includes the search-based algorithms that start from one or more initial solutions, and then modify it/them by some operators to search

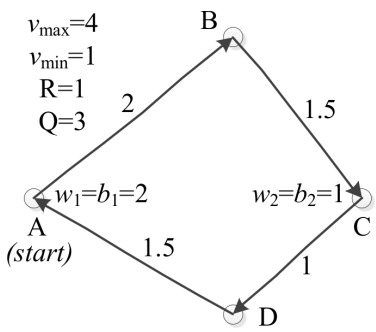

Fig. 1. An example of a TTP solution.

for improvements. The well-known search-based heuristics for TSP include the chained Lin-Kernighan (LK) heuristic [7], the efficient LK Heuristic [8], and Ant Colony Optimization [9]. Obviously, the search-based heuristics can perform no worse than the constructive heuristics, since they can take the solutions generated by the constructive heuristics as the initial solutions. Thus, we focus on the search-based heuristics in this paper. Specifically, we focus on the local-search-based heuristics, which iteratively move the current solution to one of its neighbors (e.g., the LK heuristic for TSP). To facilitate the analysis, the simplest 2-opt operator for the tour and flip operator for the picking plan is chosen as an example. The 2-opt operator selects two cities and reverse the sub-tour between them. The flip operator selects an item and flip its status (from picked to unpicked, or from unpicked to picked).

Assuming that the local search has T steps, and enumerates all the neighbors at each step. The neighborhood consists of the TSP neighborhood generated by the 2-opt operator and the KP neighborhood generated by the flip operator. Then, the TSP neighborhood size is $n(n-1)/2$ and the KP neighborhood size is m, where n and m are the number of cities and items, respectively. The fitness evaluation of a TTP solution is given in Algorithm 1. Without any efficiency optimization, the complexity of the evaluation of a neighbor is $O(n+m)$. The complexity of moving the current solution to the next one (modification) can be ignored compared with the enumeration of the neighborhood. Therefore, the complexity of the entire local search is:

$$O(LS) = T\left(\left(\frac{n(n-1)}{2} + m\right) O(m+n)\right) \tag{10}$$

$$= O(T(n^2+m)(n+m)) \tag{11}$$

$$= O(T(n^3 + n^2 m + m^2)). \tag{12}$$

Although the complexity is polynomial, the preliminary studies showed that it still makes the algorithm slow in practice, especially when n and m are large.

Algorithm 1 The evaluation of a TTP solution

1: **for** $i = 1 \rightarrow n$ **do**
2: $cw(i) = 0$, $cb(i) = 0$;
3: **end for**
4: **for** $i = 1 \rightarrow m$ **do**
5: **if** item i is picked in city a_i **then**
6: $cw(a_i) \leftarrow cw(a_i) + w_i$;
7: $cb(a_i) \leftarrow cb(a_i) + b_i$;
8: **end if**
9: **end for**
10: Set $\bar{w} = 0$, $\bar{t} = 0$, $\bar{b} = 0$;
11: **for** $i = 1 \rightarrow n - 1$ **do**
12: $\bar{w} \leftarrow \bar{w} + cw(tour(i))$;
13: $\bar{b} \leftarrow \bar{b} + cb(tour(i))$;
14: $\bar{t} \leftarrow \bar{t} + \frac{d(tour(i),tour(i+1))}{v_{\max} - \nu\bar{w}}$;
15: **end for**
16: $\bar{t} \leftarrow \bar{t} + \frac{d(tour(n),tour(1))}{v_{\max} - \nu\bar{w}}$;
17: **return** $\bar{b} - R \cdot \bar{t}$;

4 Complexity Reduction Strategies for TTP

The following strategies are commonly used to reduce the complexity of the local search:

Neighborhood size: It is obvious that there are a large number of poor neighbors which are not worth evaluating during the local search. For example, it is highly unlikely to link the cities that are far away from each other in the tour. Substantial efforts have been done to reduce the neighborhood size (e.g., the kNN strategy, K-d tree [10] [11], Delaunay triangulation [12] and Delaunay candidate set [13]). There are a number of works on dynamic neighborhood structures to adaptively choose the neighborhood that is the most suitable for the current region [14] [15] [16] [17]. In this paper, the Delaunay triangulation is chosen to reduce the TSP neighborhood, since it has been empirically shown that the edges in the optimal TSP tour are highly likely to be in the Delaunay triangulation [18]. In the Delaunay triangulation, the number of edges is reduced from $n(n-1)/2$ to at most $3n - 6$, and each city has on average 6 surrounding triangles. The Delaunay triangulation is obtained by an efficient 2-D sweepline algorithm [19], which has a complexity of $O(n \log n)$.

Incremental evaluation: In Algorithm 1, lines 1–9 compute the total weight and value of the items picked in each city, and lines 10–16 computes the total travel time of the tour. It is clear that the complexity of the evaluation is $O(m + n)$. However, when evaluating neighbors during the local search, most information required for the evaluation is maintained, and thus does not have to be re-calculated. The incremental evaluation is thus designed to prevent such redundant computations. It is obvious that the incremental evaluation for TSP and KP both reduces the complexity to $O(1)$. In TTP, all the $cw(a_i)$'s and $cb(a_i)$'s are the same for the neighbors generated by the 2-opt operators (and

all the operators that only modify the tour), and only the $cw(a_i)$ and $cb(a_i)$ of the city of the flipped items (and all the items whose status are changed) need to be updated. Therefore, it is easy to reduce the complexity of the lines 1–9 to $O(1)$, and thus reduce the complexity of the evaluation to $O(n)$. However, the complexity of computing the travel time can hardly be reduced, since one needs to recalculate $\bar{w}$, $\bar{b}$ and $\bar{t}$ for each city behind the cities moved in the tour. To the best of our knowledge, one can only keep the information of the unchanged subtour, which does not reduce the complexity. In summary, the incremental evaluation of a TTP solution has a complexity of $O(n)$.

Efficient solution modification: The modification of the tour has a worst-case complexity of $O(n)$. However, when changing the data structure of the tour, the complexity can be reduced. For example, the splay tree [20] reaches the amortized time of $O(\log n)$.

When applying all the above strategies to TTP, the complexity of the local search is reduced to:

$$O(LS) = T(O(S_1 + S_2)O(n) + O(\log n)) \tag{13}$$
$$= O(T(S_1 + S_2)n), \tag{14}$$

where S_1 and S_2 are the size of the reduced neighborhood of the 2-opt and flip operators, respectively.

Without any neighborhood reduction for the flip operator, it is known that $S_2 = O(m)$, where m is the number of items. After reducing the TSP neighborhood with the Delaunay triangulation, S_1 is reduced to $O(n)$. Therefore, we have

$$O(LS) = O(T(m + n)n) = O(T(n^2 + mn)) \tag{15}$$

It can be seen that the reduced complexity is much lower than the original one ($O(T(n^3 + n^2m + m^2)$ shown in Eq. (12)).

5 A More Efficient Two-Stage Local Search

The reduced complexity of the local-search-based heuristics for TTP shown in Eq. (15) is still high for the large scale problems encountered in practice. It is no less than n times of the complexity of the TSP and KP heuristics due to the incremental evaluation complexity of $O(n)$ ($O(1)$ for the TSP and KP heuristics). This will lead to a much longer computational time. For example, assuming that the LK heuristic takes 30 seconds to obtain a near-optimal for a $30,000$-city large scale TSP instance. With the TTP incremental evaluation, the computational time will become roughly $30,000 \times 30 = 900,000$ seconds, i.e., 250 hours. Therefore, the complexity needs to be further improved. To this end, a two-stage local search is proposed. It simply divides the entire search into two stages. The first stage is the TSP search for improving the TSP tour, and the second stage is the KP search for finding the improved picking plan under the given TSP tour.

To further reduce the complexity, the approximated fitness evaluations are designed for the two stages, respectively. During the first stage, the objective of maximizing the profit is approximated by minimizing the tour length. The approximation is motivated by maximizing the profit under the condition that no item is picked, i.e., $z_j = 0, \forall j = 1, \ldots, m$. In this situation, Eqs. (3)–(9) can be simplified as follows:

$$\max \quad -R \cdot \frac{\sum_{i=1}^{n-1} d(x_i, x_{i+1}) + d(x_n, x_1)}{v_{\max}}, \tag{16}$$

$$s.t.: \ x_i \neq x_j, \ \ 1 \leq i \neq j \leq n, \tag{17}$$

$$x_i \in \{1, \ldots, n\}. \tag{18}$$

It is obvious that the objective Eq. (16) is equivalent to minimizing the tour length $\sum_{i=1}^{n-1} d(x_i, x_{i+1}) + d(x_n, x_1)$.

The TSP search provides a TTP solution with a promising empty tour (without picking any item), which is a lower bound of the optimal TTP solution. Then, during the second stage, the items are inserted into the empty tour to improve the profit. The insertion of the items in TTP is different from that is in KP, because the profit of the insertion depends not only on the value of the inserted item, but also on the location of its inserted city in the tour. Specifically, given a tour $\mathbf{x}$ and the current picking plan $\mathbf{z}$, the change of profit caused by inserting an item j is given as follow:

$$\Delta p(j, \mathbf{x}, \mathbf{z}) = b_j - R \cdot \Delta t(j, \mathbf{x}, \mathbf{z}), \tag{19}$$

$$\Delta t(j, \mathbf{x}, \mathbf{z}) = \sum_{i=loc(a_j)}^{n-1} \left(\frac{d(x_i, x_{i+1})}{v'(i)} - \frac{d(x_i, x_{i+1})}{v(i)} \right) + \frac{d(x_n, x_1)}{v'(n)} - \frac{d(x_n, x_1)}{v(n)}, \tag{20}$$

$$v'(i) = v_{\max} - \nu(\bar{w}(i) + w_j) = v(i) - \nu w_j, \ \ 1 \leq i \leq n, \tag{21}$$

where $loc(a_j)$ is the location index of the city a_j in the tour. From Eq. (19), one can see that the insertion of an item j leads to a profit gain of its value b_j and a profit loss of $R \cdot \Delta t(j, \mathbf{x}, \mathbf{z})$ due to the increased travel time. $\Delta t(j, \mathbf{x}, \mathbf{z})$ depends on the $\bar{w}(i)$'s along the subtour after $loc(a_j)$, i.e., all the items picked in the subtour. Therefore, it is impossible to precisely evaluate each item separately, as it is in KP.

To address the above issue, three approximations to $\Delta t(j, \mathbf{x}, \mathbf{z})$ are proposed to separately estimate the priority of the items to be inserted into the tour. The first one is the called the empty-tour increased time, which is the increased time when inserting the item into an empty tour. It can be calculated as follows:

$$\Delta t_1(j, \mathbf{x}) = L(\mathbf{x}, loc(a_j)) \left(\frac{1}{v_{\max} - \nu w_j} - \frac{1}{v_{\max}} \right), \tag{22}$$

where

$$L(\mathbf{x}, loc(a_j)) = \sum_{i=loc(a_j)}^{n-1} d(x_i, x_{i+1}) + d(x_n, x_1) \tag{23}$$

is the length of the subtour of $\mathbf{x}$ from $loc(a_j)$ to the end.

The second approximation is called the worst-case increased time under a given total weight. It is known that given a total weight W, the worst-case increased time by inserting the item j is when all the weights are picked before its picked city a_j. Therefore, the worst-case increased time under a given total weight W is calculated as follows:

$$\Delta t_2(j, \mathbf{x}, W) = L(\mathbf{x}, loc(a_j)) \left(\frac{1}{v_{\max} - \nu(W + w_j)} - \frac{1}{v_{\max} - \nu W} \right). \tag{24}$$

When inserting an item into the tour $\mathbf{x}$ with the total weight of W, it is clear that $\Delta t_1(j, \mathbf{x}) \leq \Delta t(j, \mathbf{x}, \mathbf{z}) \leq \Delta t_2(j, \mathbf{x}, W)$. In other words, $\Delta t_1(j, \mathbf{x})$ and $\Delta t_2(j, \mathbf{x}, W)$ provide the lower and upper bounds of $\Delta t(j, \mathbf{x}, \mathbf{z})$ for each item.

The last approximation is called the expected increased time under a given total weight. This approximation is based on the assumption that the total weight W is uniformly distributed along the tour. Under this assumption, given a location l, where l indicates the distance from the location to the end of the tour, then the accumulated weight $\bar{w}(l)$ at location l can be calculated as follows:

$$\bar{w}(l) = \left(1 - \frac{l}{L(\mathbf{x})}\right) W, \tag{25}$$

where $L(\mathbf{x})$ is the length of the tour $\mathbf{x}$. Therefore, the expected increased time under the given total weight of W is calculated as follows:

$$\Delta t_3(j, \mathbf{x}, W) = \int_{L(\mathbf{x}) - L(\mathbf{x}, loc(a_j))}^{L(\mathbf{x})} \left(\frac{1}{v_{\max} - \nu(\bar{w}(l) + w_j)} - \frac{1}{v_{\max} - \nu \bar{w}(l)} \right) dl \tag{26}$$

$$= \frac{1}{a} \ln \left(\frac{(aL(\mathbf{x}) + b_1) \cdot (a(L(\mathbf{x}) - L(\mathbf{x}, loc(a_j))) + b_2)}{(a(L(\mathbf{x}) - L(\mathbf{x}, loc(a_j))) + b_1) \cdot (aL(\mathbf{x}) + b_2)} \right), \tag{27}$$

where

$$a = \frac{W}{L(\mathbf{x})}, b_1 = v_{\max} - \gamma(W + w_j), b_2 = v_{\max} - \gamma W. \tag{28}$$

Based on the above three approximations, a heuristic of inserting the items is designed and described in Algorithm 2.

The idea of Algorithm 2 can be explained as follows: At first, the priorities of the items are determined by $\frac{b_j - R \cdot \Delta t_1(j, \mathbf{x})}{w_j}$, which is the profit change when the tour is empty (Line 1. This approximation is more precise when the total weight of the tour is not large relative to the capacity, and works better in the early stage of the insertion. Then, as more and more items are inserted and the total weight of the tour increases, it becomes more and more important to evaluate the items given the total weight of the tour. First, the worst-case approximation is evaluated (Line 5). If it is positive, then the item is inserted since its insertion must lead to a profit gain. Otherwise, the expected approximation is evaluated (Line 7). If it is positive, then the item should be more likely to have a profit gain

Algorithm 2 The insertion heuristic of the items given a tour

Sort the items by the decreasing order of $\frac{b_j - R \cdot \Delta t_1(j, \mathbf{x})}{w_j}$;
Let the sorted item list be $(s(1), \ldots, s(m))$, and set $W = 0$;
for $j = 1 \to m$ **do**
    if $W + w_{s(j)} \leq Q$ **then**
        if $b_{s(j)} > R \cdot \Delta t_2(s(j), \mathbf{x}, W)$ **then**
            $z_{s(j)} = 1$, $W \leftarrow W + w_{s(j)}$; ▷ Insert this item
        else if $b_{s(j)} > R \cdot \Delta t_3(s(j), \mathbf{x}, W)$ **then**
            $z_{s(j)} = 1$, $W \leftarrow W + w_{s(j)}$; ▷ Insert this item
        end if
    end if
end for

rather than a profit loss. Thus, it is inserted into the tour as well. The capacity constraint is imposed as a global constraint. That is, any insertion that leads to the violation of the capacity constraint will be abandoned.

After Algorithm 2, a subsequent local search is conducted on the generated picking plan by applying the flip operator only to the inserted items. The preliminary studies showed that the local search was very fast in practice because the generated picking plan is usually good enough and there is nearly no space for the improvement by local search.

The complexity of the two-stage local search consists of that of the TSP search and the KP search. The TSP search has a complexity of $T_1 O(n)$, where T_1 is the number of steps of the search, since the evaluation approximation reduces the complexity of evaluation from $O(n)$ to $O(1)$. Then, the complexity of the KP search is $O(m) + T_2 O(nm)$, where the former term is the complexity of Algorithm 2, and the latter one is the complexity of the subsequent local search. Therefore, the complexity of the proposed two-stage local search is as follows:

$$O(TSLS) = T_1 O(n) + T_2 O(nm) \tag{29}$$

Since T_2 is usually small ($T_2 = 1$ for most of the cases in the experimental studies), the practical complexity of the two-stage local search is usually $T_1 O(n) + O(nm)$.

6 A Memetic Algorithm with the Two-Stage Local Search

The proposed Memetic Algorithm with the Two-stage Local Search (MATLS) starts from an initial population consisting of individuals with tours generated by the chained LK heuristic, and picking plans by the insertion heuristic (Algorithm 2). In each generation, two parents are randomly selected and undergo the ordered crossover [21] to obtain the tour of the offspring, which is then improved by the TSP search. After that, the picking plan of the offspring is initialized by Algorithm 2 and improved by the KP search. Finally, the offspring is added into the population to replace the worst individual.

Table 1. The performance of the compared algorithms on the large scale TTP benchmark instances with bounded strongly correlated item weights. The best results are marked in bold.

Name	n	m	RLS	EA	MATLS
brd14051	14051	140500	-5.50e+7(1.22e+6)	-6.30e+7(2.52e+6)	**2.66e+7**(2.07e+5)
d15112	15112	151110	-6.42e+7(2.52e+6)	-7.01e+7(1.63e+6)	**2.85e+7**(5.35e+5)
d18512	18512	185110	-8.18e+7(1.27e+6)	-8.89e+7(3.98e+6)	**3.07e+7**(2.73e+5)
pla33810	33810	338090	-1.71e+8(1.49e+6)	-1.80e+8(1.64e+6)	**6.34e+7**(4.59e+5)
rl11849	11849	118480	-4.38e+7(7.58e+5)	-5.05e+7(2.76e+5)	**1.97e+7**(8.29e+4)
usa13509	13509	135080	-5.45e+7(7.57e+5)	-6.17e+7(2.95e+5)	**2.92e+7**(2.60e+5)

Table 2. The performance of the compared algorithms on the large scale TTP benchmark instances with bounded uncorrelated item weights. The best results are marked in bold.

Name	n	m	RLS	EA	MATLS
brd14051	14051	140500	-4.12e+7(7.68e+5)	-4.73e+7(1.64e+5)	**1.69e+7**(2.09e+5)
d15112	15112	151110	-4.56e+7(7.44e+5)	-5.18e+7(1.08e+5)	**1.83e+7**(2.43e+5)
d18512	18512	185110	-5.98e+7(6.12e+5)	-6.62e+7(8.82e+5)	**1.94e+7**(2.47e+5)
pla33810	33810	338090	-1.22e+8(8.40e+5)	-1.28e+8(1.01e+6)	**4.10e+7**(2.38e+5)
rl11849	11849	118480	-3.23e+7(1.79e+6)	-3.81e+7(2.12e+6)	**1.29e+7**(4.47e+4)
usa13509	13509	135080	-4.00e+7(5.79e+5)	-4.55e+7(1.29e+5)	**1.88e+7**(1.69e+5)

7 Experimental Studies

To evaluate the proposed MATLS, it is tested on a subset of large scale TTP benchmark instances developed by Polyakovskiy *et al.* [22] and compared with the RLS and EA proposed in the same paper. RLS generates a decent tour by the chain LK heuristic, and then does a simple local search on the picking plan with the single flip operator. EA takes the same search process except that the status of each item has a probability of $1/m$ (m is the number of items) to be flipped in each generation. Since the investigation of TTP is still in its infancy, the two compared algorithms are the state-of-the-art published algorithms although they are relatively simple. The time budget of MATLS is set to 10 minutes, the same as that of RLS and EA. The population size is set to 30. The tours of 10 individuals are initialized by the more time-consuming chained LK heuristic, and tours of the remaining 20 individuals are initialized by the minimal spanning tree heuristic.

Tables 1–3 show the average performance (mean and standard deviation) of the compared algorithms on the selected benchmark problems, where n and m stand for the number of cities and items, respectively. It can be seen that the selected benchmark instances all have more than 10,000 cities, and 100,000 items. From the tables, it can be seen that for all the large scale instances, MATLS performed significantly better than RLS and EA. In fact, both RLS had highly negative profits, while MATLS managed to obtain highly positive profits for all the instances. This verifies the advantage of MATLS and thus the efficacy of the

Table 3. The performance of the compared algorithms on the large scale TTP benchmark instances with bounded uncorrelated but similar item weights. The best results are marked in bold.

Name	n	m	RLS	EA	MATLS
brd14051	14051	140500	-3.88e+7(1.86e+6)	-4.32e+7(1.82e+5)	**1.31e+7**(2.28e+5)
d15112	15112	151110	-4.30e+7(1.43e+6)	-4.74e+7(7.51e+5)	**1.38e+7**(2.64e+5)
d18512	18512	185110	-5.50e+7(5.86e+5)	-6.00e+7(1.09e+5)	**1.42e+7**(3.39e+5)
pla33810	33810	338090	-1.10e+8(7.47e+5)	-1.15e+8(6.16e+5)	**3.10e+7**(3.38e+5)
rl11849	11849	118480	-3.10e+7(4.54e+5)	-3.51e+7(1.70e+5)	**9.15e+6**(4.31e+4)
usa13509	13509	135080	-3.77e+7(1.18e+6)	-4.20e+7(6.38e+5)	**1.52e+7**(1.70e+5)

proposed two-stage local search. Since all the compared algorithms generated the tours of the solutions by the chained LK heuristic, the efficacy of the proposed item insertion heuristic is verified by the outperformance of MATLS over RLS and EA.

8 Conclusion

In this paper, the complexity of large scale Traveling Thief Problem (TTP) is analyzed, and the complexity reduction strategies are proposed to solve the complex problem within a given limited computational budget. The resultant Memetic Algorithm with Two-stage Local Search (MATLS) is evaluated on the large scale TTP benchmark instances with more than 10,000 cities and 100,000 items, and our results demonstrate the efficacy of MATLS on large scale TTP.

The success of MATLS implies the importance of complexity reduction for solving large scale problems to speed up the search process significantly without losing much of its accuracy, e.g., by means of neighborhood filtering and surrogate models. However, this needs to exploit domain knowledge, and the strategy may vary from problem to problem. In the case of TTP, the efficacy of the two-stage local search gives an insight that an intelligent divide-and-conquer approach taking into account the interdependence between the sub-problems (TSP and KP) can perform better than the extreme approaches, i.e., solving the problem as a whole and solving the sub-problems separately.

References

1. M. Bonyadi, Z. Michalewicz, and L. Barone, "The travelling thief problem: the first step in the transition from theoretical problems to realistic problems," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico*, 2013, pp. 1037–1044.
2. Y. Mei, K. Tang, and X. Yao, "Improved memetic algorithm for capacitated arc routing problem," in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 1699–1706.
3. ——, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 151–165, 2011.

4. ——, "A Memetic Algorithm for Periodic Capacitated Arc Routing Problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 6, pp. 1654–1667, 2011.
5. Y. Mei, X. Li, and X. Yao, "Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 435–449, 2014.
6. N. Christofides, "The optimum traversal of a graph," *Omega*, vol. 1, no. 6, pp. 719–732, 1973.
7. D. Applegate, W. Cook, and A. Rohe, "Chained lin-kernighan for large traveling salesman problems," *INFORMS Journal on Computing*, vol. 15, no. 1, pp. 82–92, 2003.
8. K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.
9. M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
10. J. Bentley, "K-d trees for semidynamic point sets," in *Proceedings of the sixth annual symposium on Computational geometry*. ACM, 1990, pp. 187–197.
11. ——, "Fast algorithms for geometric traveling salesman problems," *ORSA Journal on computing*, vol. 4, no. 4, pp. 387–411, 1992.
12. B. Delaunay, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, pp. 793–800, 1934.
13. G. Reinelt, "Fast heuristics for large geometric traveling salesman problems," *ORSA Journal on computing*, vol. 4, no. 2, pp. 206–217, 1992.
14. X. Yao, "Simulated annealing with extended neighbourhood," *International Journal of Computer Mathematics*, vol. 40, no. 3, pp. 169–189, 1991.
15. ——, "Dynamic neighbourhood size in simulated annealing," in *Proceedings of International Joint Conference on Neural Networks (IJCNN'92)*, vol. 1, 1992, pp. 411–416.
16. Y. Mei, K. Tang, and X. Yao, "A Global Repair Operator for Capacitated Arc Routing Problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 3, pp. 723–734, 2009.
17. K. Tang, Y. Mei, and X. Yao, "Memetic Algorithm with Extended Neighborhood Search for Capacitated Arc Routing Problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, 2009.
18. N. Krasnogor, P. Moscato, and M. Norman, "A new hybrid heuristic for large geometric traveling salesman problems based on the delaunay triangulation," in *Anales del XXVII Simposio Brasileiro de Pesquisa Operacional*. Citeseer, 1995, pp. 6–8.
19. B. Žalik, "An efficient sweep-line delaunay triangulation algorithm," *Computer-Aided Design*, vol. 37, no. 10, pp. 1027–1038, 2005.
20. M. Fredman, D. Johnson, L. McGeoch, and G. Ostheimer, "Data structures for traveling salesmen," *Journal of Algorithms*, vol. 18, no. 3, pp. 432–479, 1995.
21. D. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proceedings of the first international conference on genetic algorithms and their applications*. Lawrence Erlbaum Associates, Publishers, 1985, pp. 154–159.
22. S. Polyakovskiy, M. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann, "A comprehensive benchmark set and heuristics for the traveling thief problem," in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, 2014, pp. 477–484.