Heuristic Evolution with Genetic Programming for Traveling Thief Problem

Yi Mei*, Xiaodong Li*, Flora Salim*, Xin Yao†
* School of Computer Science and Information Technology, RMIT University
Melbourne, Victoria 3000, Australia
{yi.mei, xiaodong.li, flora.salim}@rmit.edu.au
† CERCIA, School of Computer Science, University of Birmingham
B15 2TT Birmingham, UK
x.yao@cs.bham.ac.uk

***Abstract*—In many real-world applications, one needs to deal with a large multi-silo problem with interdependent silos. In order to investigate the interdependency between silos (sub-problems), the Traveling Thief Problem (TTP) was designed as a benchmark problem. TTP is a combination of two well-known sub-problems, Traveling Salesman Problem (TSP) and Knapsack Problem (KP). Although each sub-problem has been intensively investigated, the interdependent combination has been demonstrated to be challenging, and cannot be solved by simply solving the sub-problems separately. The Two-Stage Memetic Algorithm (TSMA) is an effective approach that has decent solution quality and scalability, which consists of a tour improvement stage and an item picking stage. Unlike the traditional TSP local search operators adopted in the former stage, the heuristic for the latter stage is rather intuitive. To further investigate the effect of item picking heuristic, Genetic Programming (GP) is employed to evolve a gain function and a picking function, respectively. The resultant two heuristics were tested on some representative TTP instances, and showed competitive performance, which indicates the potential of evolving more promising heuristics for solving TTP more systematically by GP.**

***Index Terms*—Traveling thief problem, memetic algorithm, genetic programming, interdependent optimization**

I. Introduction

As highlighted in [1], in practice, one often encounters complex problems which have multiple interdependent modules, each of which can be called a *silo*. For example, in the mine supply chain, one needs to solve the storing, production scheduling and delivery silos simultaneously, which are interdependent with each other. A complex problem with interdependent vehicle routing problem and location allocation problem was proposed in [2], and a stochastic scheduling problem with interdependent modules was proposed in [3]. In these cases, solving each silo separately can by no means lead to reasonable results due to the neglect of the interdependency. In other words, when solving multi-silo problems, how to deal with the interdependency between the silos largely determines the goodness of the final solution to the entire problem, as well as the single-silo solution components.

In order to investigate the effect of interdependence between sub-problems in a complicated problem, Bonyadi *et al.* [4] proposed an artificial benchmark problem called the Traveling Thief Problem (TTP). TTP is composed of two sub-problems, i.e., the Travelling Salesman Problem (TSP) and Knapsack Problem (KP), both of which are well-known in operations research and combinatorial optimisation. In TTP, a thief starts from a specific starting city to visit a predefined set of cities exactly once, pick certain items along the way, and finally returns to the starting city. Each item has a value and a weight. To keep the picked items during the trip, the thief rents a knapsack, whose capacity is limited and cannot be exceeded by the total weight of the picked items. The rent of the knapsack is proportional to the travel time. Then, the problem aims to find a tour and a picking plan for the thief to maximize the total profit of the trip, which can be calculated as the total value of the picked items minus the rent of the knapsack. TSP and KP have been intensively investigated and there are already a variety of effective heuristics for solving each of them alone, however there are few studies on examining them as subcomponents of one holistic problem. Investigation on TTP facilitates studies on analysing the interdependence between sub-problems.

Although TTP was artificially designed, one can still find relevant real-world applications. One example 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. It is more practical and closer to reality than the models that have been studied intensively [5] [6] [7] [8].

Since its proposal, there have been several algorithms [9] designed for solving TTP. Among them, the Two-Stage Memetic Algorithm (TSMA) [10] is one of the state-of-the-art algorithms that can produce reasonably good solutions within a short time budget. Its efficacy has been demonstrated by its performance on the CEC'2014 TTP competition benchmark instances, and its scalability is attributed to the separation of the tour improvement stage and the item picking stage. Unlike the traditional local search operators (e.g., the 2-opt operator) adopted in the former stage, the design of the item picking heuristic in the latter stage is rather intuitive. The aim of this paper is to conduct a more systematic investigation on the item picking heuristic in TSMA, i.e., searching for simpler and better item picking heuristics for TSMA.

In this paper, Genetic Programming (GP) is employed to search in two different heuristic spaces, and thus obtain two different item picking heuristics. The first one is called the *gain sorting* heuristic, and the second one is called the *picking function* heuristic. The heuristics were trained on three small representative TTP instances and promising tours obtained by the chained Lin-Kernighan heuristic [11], and tested on a set of large scale instances. The results show that both heuristics have decent generalizability from the small scale training set to the large scale testing set. This shows the efficacy of GP for evolving promising heuristics when combined with some well-structured framework such as TSMA.

The rest of the paper is as follows: In Section II, TTP is described in details. Then, TSMA is briefly reviewed in Section III. Afterwards, the evolution of the gain sorting and picking function heuristics using GP is described in IV. The experimental studies are carried out in Section V. Finally, the conclusion and future work is given in Section VI.

II. Traveling Thief Problem

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. According to [10], the TTP problem can be stated as follows:

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

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

$$\bar{w}(i) = \sum_{k=1}^{i-1} \bar{w}(k) + cw(x_i), \;\; 1 \leq i \leq n, \tag{5}$$

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

$$x_i \neq x_j, \;\; 1 \leq i \neq j \leq n, \tag{7}$$

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

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

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.

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.

III. Two-Stage Memetic Algorithm Revisited

The Two-Stage Memetic Algorithm (TSMA) takes a traditional memetic algorithm framework, which can be seen as the combination of genetic algorithm and local search. To be more specific, the mutation operator in genetic algorithm is replaced by local search. Whenever an individual needs to mutate, it undergoes a local search process. This framework has been demonstrated to be effective for solving combinatorial optimisation problems [5] [12] [13] [6] [8].

In TSMA, an individual is represented by a TSP tour and a KP picking plan. First, a population of individuals is initialized by the chained Lin-Kernighan (LK) heuristc [11] and the minimum spanning tree constructive heuristic. At each generation, two parents are selected from the current population, and traditional TSP crossover operator (i.e., the ordered crossover operator [14]) is applied to their tours. Then, the generated offspring undergoes the two-stage local search, which consists of a TSP search and a KP search. First, the tour of the offspring is improved by the TSP search, which generates the neighbours by the 2-opt operator. After that, the picking plan of the offspring based on the improved tour is generated by a sophisticate item picking heuristic [10], following by a simple KP search, which examines the neighbours generated by the single flip operator. Finally, the offspring is added into the population to replace the worst individual. The pseudo code of TSMA is described in Algorithm 1.

Several complexity reduction strategies have been designed to improve the efficiency of the algorithm. For example, the Delaunay Triangulation [15] [16] is adopted to restrict the number of neighbours examined during the TSP search, and the incremental evaluation technique is developed to prevent redundant calculations in individual re-evaluation during the local search and thus reduce the computational complexity of the fitness evaluation of neighbours during the local search process. More details of the complexity reduction strategies employed in TSMA are referred to [10].

Algorithm 1 The Two-Stage Memetic Algorithm

```
1:  for $i = 1 \to popsize$ do
2:      Initialize $pop[i].tour$ with the chained LK heuristic;
3:      Apply the item selection heuristic to obtain $pop[i].pickplan$;
4:      Evaluate $pop[i]$;
5:  end for
6:  Sort pop in terms of the profit of the individuals;
7:  while Stopping criterion is not met do
8:      Randomly select two parents $par_1$ and $par_2$ from pop;
9:      Apply the ordered crossover [14] on $par_1.tour$ and $par_2.tour$ to generate the tour of the offspring $o.tour$;
10:     Conduct TSP search on $o.tour$ to improve it;  ▷ the first stage
11:     Generate $o.pickplan$ by the item selection heuristic [10];
12:     Conduct KP search on $o.pickplan$ to improve it;  ▷ the second stage
13:     Evaluate the offspring $o$, and add it into pop;
14:     Sort pop by the decreasing order of profit, and remove the worst individual;
15: end while
```

In TSMA, the item selection heuristic plays an important role in improving the performance of the algorithm. Given a local optimal tour obtained by the TSP search, the item selection heuristic provides a promising starting point of the subsequent KP search, which thus largely increases its computational effort. In practice, the picking plan obtained by the heuristic is already a local optimum in most cases (the KP search can no longer improve it any more). In [10], the item selection heuristic is designed intuitively based on the understandings of the authors about the problem itself. Concretely, three approximated fitness measures of each item are defined to reflect its relative priority to be selected based on the current tour. The first measure is named the *gain in empty tour* (denoted as Δf_1), which means the gain (difference in profit) when picking an item under an empty tour without picking any other items. Given a tour $\mathbf{x}$, for the j^{th} item, $\Delta f_1(j, \mathbf{x})$ is calculated as follows:

$$\Delta f_1(j, \mathbf{x}) = b_j - R \cdot \Delta t_1(j, \mathbf{x}), \tag{10}$$

where

$$\Delta t_1(j, \mathbf{x}) = \frac{L(\mathbf{x}, loc(a_j))}{v_{\max} - \nu w_j} - \frac{L(\mathbf{x}, loc(a_j))}{v_{\max}} \tag{11}$$

is the increased travel time caused by selection item j, and

$$L(\mathbf{x}, loc(a_j)) = \sum_{i=loc(a_j)}^{n-1} d(x_i, x_{i+1}) + d(x_n, x_1) \tag{12}$$

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

The second measure is the so-called the *worst-case gain under the total weight* (denoted as Δf_2). Given a tour $\mathbf{x}$ and a total weight W, $\Delta f_2(j, \mathbf{x}, W)$ of the j^{th} item is calculated as follows:

$$\Delta f_2(j, \mathbf{x}, W) = b_j - R \cdot \Delta t_2(j, \mathbf{x}, W), \tag{13}$$

$$\Delta t_2(j, \mathbf{x}, W) = \frac{L(\mathbf{x}, loc(a_j))}{v_{\max} - \nu(W + w_j)} - \frac{L(\mathbf{x}, loc(a_j))}{v_{\max} - \nu W}. \tag{14}$$

The third measure is called the *expected gain under the total weight* (denoted as Δf_3), which can be calculated as follows:

$$\Delta f_3(j, \mathbf{x}, W) = b_j - R \cdot \Delta t_3(j, \mathbf{x}, W), \tag{15}$$

$$\Delta t_3(j, \mathbf{x}, W) = \frac{L(\mathbf{x})}{g} \left(\log \frac{g + h_2}{h_2} - \log \frac{g + h_1}{h_1} \right) \tag{16}$$

where $g = \nu(W_1 - W)$, $h_1 = v_{\max} - \nu W_1$, $h_2 = v_{\max} - \nu W_2$, $W_1 = \frac{1 - L(\mathbf{x}, loc(a_j))}{L(\mathbf{x})} W$, and $W_2 = W_1 + w_j$.

In the item selection heuristic, the items are first sorted in the decreasing order of the ratio $\Delta f_1 / w_j$, and examined one by one during the selection process. When considering each item, Δf_2 is first calculated under the current weight. If it is positive, then the item is selected. Otherwise, Δf_3 is calculated and the item is selected if $\Delta f_3 > 0$.

IV. Evolving Item Selection Heuristics with Genetic Programming

From the description in Section III, it is obvious that the item selection heuristic is manually designed based on the understanding of the problem, i.e., the potential factors affecting the solution quality. However, this is rather arbitrary and thus it is still unknown whether such heuristic is really promising or not, since there is no existing competitive heuristics to be compared with. In this paper, such issue is addressed by evolving the item selection heuristics by Genetic Programming (GP). By searching within the heuristic space, one can have a more comprehensive understanding about the item selection heuristics, e.g., whether the selected heuristic is promising, or how much difference are there between different heuristics.

GP [17] [18] has been successfully applied to evolve competitive heuristics for a wide range of complex combinatorial optimisation problems [19] [20] [21] [22], and has shown advantages in scalability, flexibility and adaptability in a dynamic environment. In this paper, GP is adopted to search within the space of the item selection heuristics, which will be later embedded into the TSMA framework. More specifically, two different GPs are developed to search for two different heuristics that will be embedded into different levels of the TSMA framework. The first heuristic is called the *item selection by gain* (GAIN) heuristic, and the second one is called the *item selection by picking function* (PICKFUNC) heuristic.

TABLE I: The terminal set in the GAIN heuristic.

Terminal	Description
profit	Profit of the item
weight	Weight of the item
bdist	Distance of the location of the item to the end of the tour
Q	The capacity of the knapsack
L	The total length of the tour
R	The rent ratio of the knapsack
ν	The coefficient defined by Eq. (2)
$v_{\max}$	The maximal speed

A. Item Selection By Gain Heuristic

As the name implies, this heuristic selects the items based on some gain function, which is evolved by GP. Specifically, during the local search of TSMA, given the TSP tour improved by the TSP search, the GAIN heuristic evaluates each item by the real-valued gain function produced by GP. Then, the items are sorted in the decreasing order of their gain function values, given the assumption that a more desirable item has a higher gain function value. Then, the sorted items are examined one by one and are selected to be picked until there is no more item that can be selected without violating the capacity constraint.

1) Individual Representation: A standard tree-based GP representation is adopted here. The terminal set is given in Table I. The first three are item-related terminals (the profit, weight and distance to the end of the tour). It is obvious that they are closely related to the desirability of an item, since an item is intuitively more promising if it has a higher profit, lower weight and is closer to the end of the tour, and thus has less impact on the travel time increase. The latter five terminals are problem properties, which are used to give references to the item-related terminals. For example, they can transform *bdist* into the same scale of *profit* (e.g., Eqs. (10) and (11)).

In addition to the above terminals, random real-valued numbers are sampled from the uniform distribution between 0 and 1 to add into the terminal set during the GP search process.

The standard arithmetic operators "$+$", "$-$", "$\times$" and "$/$" are adopted here to keep a simple and small search space of the heuristic.

2) Fitness Evaluation: During the GP process, each individual is a tree representing a gain function that can be applied to each item. Then, each individual f is evaluated by two steps. First, the picking plan under the current tour is obtained by the corresponding item selection heuristic based on the gain function f. Then, the entire TTP solution consisting of both the tour and the picking plan are evaluated, and its profit is set as the fitness value of f. The details of such fitness evaluation is described in Algorithm 2.

B. Item Selection By Picking Function

The PICKFUNC heuristic is somehow of lower-level than the GAIN heuristic, as it employs more of the knowledge about the problem. Specifically, the GAIN heuristic replaces the whole item selection heuristic in TSMA, while the PICKFUNC heuristic does not change the order of the items sorted by Δf_1 over the item weight, but replaces the inner decision process based on Δf_2 and Δf_3 by a so-called picking function f_p. That is, when determining whether each item should be selected or not, one examines whether the f_p provided by GP is positive or not instead of checking the signs of Δf_2 and Δf_3.

Algorithm 2 Fitness evaluation of a gain function f

```
1:  Initialize the picking plan $\mathbf{z} = \mathbf{0}_{m\times 1}$;
2:  for $i = 1 \to m$ do
3:      Calculate $f(i)$ for item $i$;
4:  end for
5:  Sort the items in the decreasing order of $f(i)$ to obtain
    the sorted list $(s_1, \dots, s_m)$ $(f(s_i) > f(s_j), \forall i < j)$;
6:  $W = 0$;
7:  for $i = 1 \to m$ do
8:      if $w(s_i) \le Q - W$ then
9:          $\mathbf{z}(s_i) = a_{s_i}$ (Pick item $s_i$);
10:         $W = W + w(s_i)$;
11:     end if
12: end for
13: Evaluate the profit $profit(\mathbf{x}, \mathbf{z})$ of the TTP solution $(\mathbf{x}, \mathbf{z})$;
14: return $fitness(f) = profit(\mathbf{x}, \mathbf{z})$
```

TABLE II: The terminal set in the PICKFUNC heuristic.

Terminal	Description
profit	Profit of the item
weight	Weight of the item
bdist	Distance of the location of the item to the end of the tour
Q	The capacity of the knapsack
L	The total length of the tour
R	The rent ratio of the knapsack
ν	The coefficient defined by Eq. (2)
$v_{\max}$	The maximal speed
W	The total weight of the items selected so far

1) Individual Representation: The terminal set is given in Table II. All the terminals except the last one are the same as those used in the GAIN heuristic. Then newly introduced terminal W is the current total weight of the tour, which influences the temporary desirability of the unselected items. For example, ceteris paribus, a larger W leads to a larger increase of the travel time when selecting a new item. Therefore, the temporal gain of an item that is more distant from the end of the tour will deteriorate more under a larger W.

During the GP search process, random real-valued numbers are sampled from the uniform distribution between 0 and 1 to add into the terminal set. The standard arithmetic operators "$+$", "$-$", "$\times$" and "$/$" are used in the PICKFUNC heuristic, as they are in the GAIN heuristic.

2) Fitness Evaluation: As mentioned before, the picking function f_p is used to replace the decision process based on Δf_2 and Δf_3 in the original item selection heuristic. Therefore, the fitness of f_p is defined as the profit of the TTP solution obtained by the corresponding item selection heuristic. The details of the fitness evaluation in the PICKFUNC heuristic is given in Algorithm 2. The main differ-

Algorithm 3 Fitness evaluation of a picking function f_p

```
1:  Initialize the picking plan $\mathbf{z} = \mathbf{0}_{m\times 1}$;
2:  for $i = 1 \to m$ do
3:      Calculate $\Delta f_1(i)$ for item $i$ by Eq. (10);
4:  end for
5:  Sort the items in the decreasing order of $\Delta f_1(i)/w_i$
    to obtain the sorted list $(s_1, \dots, s_m)$ $(\Delta f_1(i)/w_i >
    \Delta f_1(j)/w_j, \forall i < j)$;
6:  $W = 0$;
7:  for $i = 1 \to m$ do
8:      if $w(s_i) > Q - W$ then
9:          continue;
10:     end if
11:     if $f_p(s_i) > 0$ then
12:         $\mathbf{z}(s_i) = a_{s_i}$ (Pick item $s_i$);
13:         $W = W + w(s_i)$;
14:     end if
15: end for
16: Evaluate the profit $profit(\mathbf{x}, \mathbf{z})$ of the TTP solution $(\mathbf{x}, \mathbf{z})$;
17: return $fitness(f)$ = $profit(\mathbf{x}, \mathbf{z})$
```

ence between the PICKFUNC heuristic and the original item selection heuristic is on line 11, where the condition of $\Delta f_2 > 0 \vee (\Delta f_2 \leq 0 \wedge \Delta f_3 > 0)$ is replaced by $f_p > 0$.

C. *Training and Testing Processes*

Both the GAIN and PICKFUNC heuristics described above are used to obtain the picking plan given a TTP instance and a TSP tour. It is impractical to evolve a specific heuristic for each given instance and TSP tour. In fact, the ultimate goal of GP is to evolve heuristics that can generalize well on the unseen cases, i.e., testing instances. Therefore, one needs to learn the heuristics on certain training instances by GP, and then apply them directly to all the testing instances.

When selecting the training instances, the representativeness and scalability are two of the important characteristics to be considered. First, representativeness determines the distribution of the variables in the training and test sets. If the distributions are close to each other, then the learned heuristics are more likely to generalize well to the testing cases. Second, the scalability influences the efficiency of the training process. Ideally, a scalable training process is expected to learn the heuristics on small sized training instances which can generalize well on large scale testing instances.

In the case of TTP benchmark instances [9], it is known that there are three categories of instances: bounded strongly correlated, uncorrelated and uncorrelated with similar weights. Thus, one should select at least one training instance from each of the three categories. On the other hand, by taking the scalability into account, the size of the training instances should be as small as possible. Keeping this in mind, we selected the three instances "eil51_n50_bounded-strongly-corr_01", "eil51_n50_uncorr_01" and "eil51_n50_uncorr-similar-weights_01" from the TTP benchmark instances as our training instances. All the three training instances have 51 cities and 50 items, which are among the smallest instances in the benchmark set. Note that for evolving the heuristics, a tour has to be given as well as the TTP instance. To this end, for each instance, the chained LK heuristic is run once, and the resultant TSP tour is selected as the input tour for evolving the item selection heuristic.

During the training process, the fitness evaluations of the heuristics need to be extended from single instance to multiple instances. Originally, the fitness of a gain function or a picking function is defined for a given TTP instance and a tour. When given multiple instances and tours, the problem becomes a multi-objective one, which aims to optimize the fitness of the heuristics on multiple scenarios simultaneously. Here, to simplify the optimization process, the multi-objective optimisation problem is transformed into a single-objective one by the commonly used weighted sum approach. Specifically, given the three training scenarios, the three corresponding fitness values $fitness_1$, $fitness_2$ and $fitness_3$ are transformed into a single-objective function as follows:

$$fitness = \frac{1}{3}\sum_{i=1}^{3} \frac{profit_i^{\dagger} - fitness_i}{profit_i^{\dagger}}, \tag{17}$$

where $profit_i^{\dagger}$ is the profit of the best known solution of the i^{th} TTP instance, which is obtained by running the original TSMA once. In other words, the single-objective function implies the average percentage deviation of the fitness of the evolved heuristic from the best known profit. Note that the quality of such transformation depends on that of the selected best known solutions. However, we expected that the gap between the best known result and the true optimum does not affect the final results much, since the fitness values for each scenario are already of the same scale after the transformation.

V. Experimental Studies

A. *Experimental Settings*

To evaluate the proposed GP and the evolved item selection heuristics, three representative small sized TTP benchmark instances [9], i.e., "eil51_n50_bounded-strongly-corr_01", "eil51_n50_uncorr_01" and "eil51_n50_uncorr-similar-weights_01" are selected as the training instances. All the instances have 51 cities and 50 items. The i^{th} item is located at the $(i+1)^{th}$ city. For each instance, the chained LK heuristic is then run once on it and the resultant TSP tour is taken as the input tour of the corresponding GP.

The GP package in the R software, which is named the RGP package, is employed to evolve the heuristics based on the terminal sets defined in Section IV. The population size is set to 100, and the stopping criterion is set to a maximal running time of two hours. The preliminary studies showed that the stopping criterion is more than sufficient to guarantee the convergence of GP.

During the testing phase, another subset of large scale TTP benchmark instances are adopted to test the generalizability of the heuristics evolved by GP. Concretely, given a test instance, the TSMA is applied with the corresponding component replaced by the GAIN or PICKFUNC heuristic, respectively. The time budget of the algorithm is set to 10 minutes, which is the proposed time budget of CEC2014 and CEC2015 competition on problems with multiple interdependent components. For each test instance, 30 independent runs were conducted for each compared algorithm, and the corresponding results were compared statistically.

B. Training Performance

1) The GAIN heuristic: When evolving the item selection heuristics with GP, an elite set (which is set to one tenth of the population size, i.e., 10 in the experiments) is returned as the potential set of promising heuristics. For the sake of simplicity, the best elitist in terms of the training fitness value was selected for testing. The final gain function is described as follows:

$$f = (\mathit{profit} - \mathit{weight} + L)\frac{(\mathit{profit} + L)(\mathit{profit} \cdot \mathit{profit})}{\mathit{weight} * (LQ + \mathit{weight} \cdot \mathit{bdist})} \quad (18)$$

Its training performance is $\mathit{fitness}_1 = 3003.98$, $\mathit{fitness}_2 = 340.54$ and $\mathit{fitness}_3 = 1552.12$. Given that $\mathit{profit}_1^\dagger = 3948$, $\mathit{profit}_2^\dagger = 2730$ and $\mathit{profit}_3^\dagger = 1447$, the transformed fitness value is calculated as

$$\mathit{fitness} = \frac{0.239 + 0.875 - 0.073}{3} = 0.347. \quad (19)$$

One can see that the evolved GAIN heuristic performed the best on the last scenario (on which the best known solution was even improved), and performed the worst on the second scenario. The reason can be mainly attributed to the differences between the structures of the problems. The second scenario is the most complex one, since the items have uncorrelated profits and weights. The last scenario is the simplest one, as all the items have nearly the same weights. As a result, the weight factor can be almost ignored.

When taking a deeper look into the evolved gain function, it is obvious that *profit* always has a positive sign, and lies in the numerator component, while *weight* and *bdist* either have a negative sign or are ingredients of the denominator. In other words, an item with a larger profit, smaller weight and shorter distance to the end of the tour has a higher gain function value, and thus is considered to be more promising. This is consistent with our intuition.

2) The PICKFUNC Heuristic: The best picking function in the elitist set is described as follows:

$$f_p = \left(0.93 - \left(\frac{\mathit{bdist}}{\mathit{profit}} - \nu\right)\right)\mathit{bdist} - \frac{\nu + R \cdot \mathit{profit} - 0.73 * (\frac{\mathit{weight}}{W} + L)}{\mathit{bdist}}. \quad (20)$$

The training performance of the picking function Eq. (20) is $\mathit{fitness}_1 = 2411.81$, $\mathit{fitness}_2 = 609.41$ and $\mathit{fitness}_3 = 1338.95$, and the transformed fitness value is

$$\mathit{fitness} = \frac{0.389 + 0.777 + 0.075}{3} = 0.414. \quad (21)$$

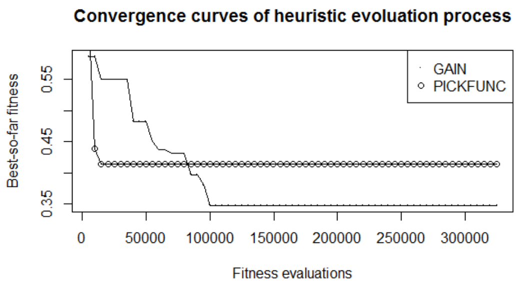

Fig. 1: The convergence curves of the GAIN and PICKFUNC heuristic evolution processes.

It is seen that the picking function performed better than the gain function on the second training instance, which is the most complex one, while was outperformed by the gain function on the first and last training instances. Overall, the picking function showed a worse training performance than the gain function (0.414 versus 0.347).

From Eq. (20), it is quite difficult to identify meaningful information about the desirability of the items. That is, one can hardly tell the reason why a positive value of Eq. (20) indicates a more promising item than a negative value. Therefore, the training performance of Eq. (20) is largely attributed to the well-organized framework of the heuristic, in which the items have already been sorted based on $\Delta f_1(i)/w_i$. It is obvious that an item with a larger Δf_1 and a smaller weight tends to be more promising.

Fig. 1 shows the convergence curves of the GAIN and PICKFUNC heuristic evolution processes by GP. It can be seen that the PICKFUNC heuristic converged very fast, and then was stuck in an local optima. On the contrary, the GAIN heuristic converged more slowly, but managed to achieve a better local optimum than the PICKFUNC heuristic. This implies that the more structured framework of PICKFUNC restricted the search space of the heuristic within a relatively narrow area, and thus increased the premature rate.

Overall, both evolved heuristics converged to some local optima for the given training instances and tours. This is consistent with the discovery found in [4] that a tour with a low total cost does not necessarily lead to a high-quality solution for the entire TTP. However, it is still a good tour for training without any other prior information.

C. Test Performance

To evaluate the test performance, the obtained GAIN and PICKFUNC heuristics were embedded into the original TSMA to replace the corresponding components (according to Algorithms 2 and 3), and the resultant algorithms were applied to a subset of large scale benchmark instances, which have more than 10,000 cities and 100,000 items. There are 6 different graphs, and 3 different instances, each of one category, for each graph. Thus, there are totally 18 instances selected as the test instances.

Tables III–V show the average performance (mean and standard deviation) of the compared algorithms on the selected

TABLE III: The performance of the compared algorithms on the large scale TTP benchmark instances with bounded strongly correlated item weights.

Name	n	m	RLS	EA	TSMA	TSMA-GAIN	TSMA-PICKFUNC
brd14051	14051	140500	-5.50e+7(1.22e+6)	-6.30e+7(2.52e+6)	**2.66e+7(2.07e+5)**	**2.66e+7(2.59e+5)**	**2.66e+7(3.41e+5)**
d15112	15112	151110	-6.42e+7(2.52e+6)	-7.01e+7(1.63e+6)	**2.85e+7(5.35e+5)**	**2.83e+7(3.40e+5)**	**2.84e+7(4.75e+5)**
d18512	18512	185110	-8.18e+7(1.27e+6)	-8.89e+7(3.98e+6)	**3.07e+7(2.73e+5)**	**3.07e+7(3.47e+5)**	**3.06e+7(4.95e+5)**
pla33810	33810	338090	-1.71e+8(1.49e+6)	-1.80e+8(1.64e+6)	**6.34e+7(4.59e+5)**	**6.32e+7(6.87e+5)**	**6.33e+7(5.11e+5)**
rl11849	11849	118480	-4.38e+7(7.58e+5)	-5.05e+7(2.76e+5)	**1.97e+7(8.29e+4)**	**1.97e+7(9.06e+4)**	**1.97e+7(1.04e+5)**
usa13509	13509	135080	-5.45e+7(7.57e+5)	-6.17e+7(2.95e+5)	**2.92e+7(2.60e+5)**	**2.92e+7(2.55e+5)**	**2.92e+7(1.70e+5)**

TABLE IV: The performance of the compared algorithms on the large scale TTP benchmark instances with bounded uncorrelated and dissimilar item weights.

Name	n	m	RLS	EA	TSMA	TSMA-GAIN	TSMA-PICKFUNC
brd14051	14051	140500	-4.12e+7(7.68e+5)	-4.73e+7(1.64e+5)	**1.69e+7(2.09e+5)**	**1.70e+7(1.78e+5)**	**1.69e+7(2.30e+5)**
d15112	15112	151110	-4.56e+7(7.44e+5)	-5.18e+7(1.08e+5)	**1.83e+7(2.43e+5)**	**1.83e+7(1.88e+5)**	**1.82e+7(2.31e+5)**
d18512	18512	185110	-5.98e+7(6.12e+5)	-6.62e+7(8.82e+5)	**1.94e+7(2.47e+5)**	**1.94e+7(2.94e+5)**	**1.94e+7(1.84e+5)**
pla33810	33810	338090	-1.22e+8(8.40e+5)	-1.28e+8(1.01e+6)	**4.10e+7(2.38e+5)**	**4.10e+7(3.37e+5)**	**4.10e+7(2.82e+5)**
rl11849	11849	118480	-3.23e+7(1.79e+6)	-3.81e+7(2.12e+6)	**1.29e+7(4.47e+4)**	**1.29e+7(4.73e+4)**	**1.29e+7(4.61e+4)**
usa13509	13509	135080	-4.00e+7(5.79e+5)	-4.55e+7(1.29e+5)	**1.88e+7(1.69e+5)**	**1.87e+7(4.57e+5)**	**1.88e+7(2.94e+5)**

TABLE V: The performance of the compared algorithms on the large scale TTP benchmark instances with bounded uncorrelated but similar item weights.

Name	n	m	RLS	EA	TSMA	TSMA-GAIN	TSMA-PICKFUNC
brd14051	14051	140500	-3.88e+7(1.86e+6)	-4.32e+7(1.82e+5)	**1.31e+7(2.28e+5)**	**1.31e+7(2.10e+5)**	**1.30e+7(2.04e+5)**
d15112	15112	151110	-4.30e+7(1.43e+6)	-4.74e+7(7.51e+5)	**1.38e+7(2.64e+5)**	**1.37e+7(3.29e+5)**	**1.37e+7(3.48e+5)**
d18512	18512	185110	-5.50e+7(5.86e+5)	-6.00e+7(1.09e+5)	**1.42e+7(3.39e+5)**	**1.42e+7(2.03e+5)**	**1.42e+7(3.32e+5)**
pla33810	33810	338090	-1.10e+8(7.47e+5)	-1.15e+8(6.16e+5)	**3.10e+7(3.38e+5)**	**3.10e+7(3.08e+5)**	**3.10e+7(3.97e+5)**
rl11849	11849	118480	-3.10e+7(4.54e+5)	-3.51e+7(1.70e+5)	**9.15e+6(4.31e+4)**	**9.16e+6(4.94e+4)**	**9.17e+6(4.88e+4)**
usa13509	13509	135080	-3.77e+7(1.18e+6)	-4.20e+7(6.38e+5)	**1.52e+7(1.70e+5)**	**1.52e+7(1.87e+5)**	**1.52e+7(1.55e+5)**

benchmark problems, where n and m stand for the number of cities and items, respectively. The results of RLS and EA [9] were included in the tables as references. From the tables, it can be seen that for all the test instances, the original TSMA, the TSMA with GAIN heuristic (TSMA-GAIN) and the TSMA with PICKFUNC heuristic (TSMA-PICKFUNC) performed significantly better than RLS and EA. Meanwhile, the test performance of TSMA-GAIN and TSMA-PICKFUNC are statistically the same as that of the original TSMA on all the test instances. In other words, both the GAIN and PICKFUNC heuristics evolved by GP managed to obtain comparable results as that of the sophisticated designed item selection heuristic in the original TSMA, which are significantly better than the compared RLS and EA. This also demonstrates the generalizability of the trained heuristics on the unseen test instances.

Based on the above observations, we have the following interpretations:

1) For both the GAIN and PICKFUNC heuristics, the generalizability of the evolved solutions from the small-sized training instances to the large scale test instances is verified by their competitive test performance;
2) The gain function evolved by GP looks intuitively reasonable, as it attaches more importance to the items with larger profit, smaller weight and shorter distance to the end of the tour. As a result, the corresponding GAIN heuristic showed a competitive test performance, which is comparable as the performance of the original TSMA (which directly optimizes the given problems);
3) Contrary to the gain function, the evolved picking function brings about less intuitive information, and is thus more difficult to analyze. However, interestingly, it still managed to obtain nearly the same solutions on the test instances as that of both the original TSMA and TSMA-GAIN. There might be two possible explanations to this phenomenon. First, there may be some potential useful information hidden in the picking function that helps to identify the items to be selected, which has not been observed. Second, after sorting the items in the decreasing order of $\Delta f_1(i)/w_i$, the space of the picking plan generated by selecting the items in order is relatively small. In addition, the item selection heuristic is followed by a local search, which pushes the picking plan generated by the heuristic to the nearest local optimum. Then, the picking function may not really affect the final local optimal picking plan, i.e., different picking function may lead to the same final picking plan after the subsequent local search. More investigations are to

be conducted to verify the above hypothesis;

4) The better training performance of the gain function than that of the picking function did not necessarily lead to a better test performance. This indicates that the training fitness measure may not be proper to identify more promising heuristics. For example, only one tour was selected for each training instance, which may be quite different from the tours of the high-quality TTP solutions. As a result, even the best item selection heuristics found by GP still have a poor training performance (e.g., 0.347 for the gain function and 0.414 for the picking function). Therefore, how to select proper tours for the training instances is another issue to be addressed.

VI. Conclusion

This paper carries out a more systematic investigation on the item selection heuristic in TSMA for solving TTP. Contrary to the original manually designed item selection heuristic, GP is adopted to evolve effective item selection heuristics automatically. Two different types of heuristics, namely the GAIN heuristic and the PICKFUNC heuristic are designed to replace different components of the original item selection heuristics. The GAIN heuristic replaces the whole item selection heuristic, while the PICKFUNC heuristic keeps the original main framework and replaces a lower level component, i.e., the conditions to determine whether to select the current item or not.

Both heuristics were trained on small scale representative training instances, and tested on another set of large scale test instances. The experimental results showed that even with a naive GP without much parameter tuning, the obtained heuristics already managed to show comparable performance as the original sophisticated designed heuristic. This demonstrates the efficacy of using GP to find out promising item selection heuristics for solving TTP, and it is expected that a more structured GP such as the strong-typed GP will improve the training and test performance of the evolved heuristics. Additionally, the generalizability of the evolved heuristics verifies the feasibility to learn heuristics on smaller and simpler TTP instances and generalize them on larger and more complex instances to improve the efficiency.

The work in this paper is merely a preliminary work on this topic. In the future, the following work will be done to further understand the problem and improve the performance of the algorithm: (1) Address the representativeness of the tour for each training instance to improve the training performance of the heuristics; (2) Employ more sophisticated GP structures such as the strong-typed GP to restrict the huge search space of the heuristics properly to improve the efficiency of the GP search; and (3) Improve the selection of the final elitist heuristics rather than simply selecting the one with the best training performance.

References

[1] Z. Michalewicz, "Quo vadis, evolutionary computation? on a growing gap between theory and practice," in *Advances in Computational Intelligence, Lecture Notes in Computer Science*. Springer, 2012, vol. 7311, pp. 98–121.

[2] K. Gupta and X. Yao, "Evolutionary approach for vehicle routing problem with time windows and facility location allocation problem," in *Proc. of the 2002 UK Workshop on Computational Intelligence (UKCI'02)*.

[3] J. Xiong, J. Liu, Y. Chen, and H. Abbass, "A knowledge-based evolutionary multiobjective approach for stochastic extended resource investment project scheduling problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 742–763, 2014.

[4] 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.

[5] 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.

[6] ——, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 151–165, 2011.

[7] ——, "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.

[8] 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.

[9] 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.

[10] Y. Mei, X. Li, and X. Yao, "Improving efficiency of heuristics for the large scale traveling thief problem," in *Simulated Evolution and Learning*. Springer, 2014, pp. 631–643.

[11] 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.

[12] M. Tang and X. Yao, "A memetic algorithm for VLSI floorplanning," *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 62–69, 2007.

[13] 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.

[14] 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.

[15] 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.

[16] B. Žalik, "An efficient sweep-line delaunay triangulation algorithm," *Computer-Aided Design*, vol. 37, no. 10, pp. 1027–1038, 2005.

[17] J. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[18] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, *Genetic programming: an introduction*. Morgan Kaufmann San Francisco, 1998, vol. 1.

[19] S. Nguyen, M. Zhang, M. Johnston, and K. Tan, "A coevolution genetic programming method to evolve scheduling policies for dynamic multi-objective job shop scheduling problems," in *2012 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2012, pp. 1–8.

[20] ——, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, 2013.

[21] J. Xie, Y. Mei, A. Ernst, X. Li, and A. Song, "A genetic programming-based hyper-heuristic approach for storage location assignment problem," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 3000–3007.

[22] ——, "Scaling up solutions to storage location assignment problems by genetic programming," in *Simulated Evolution and Learning*. Springer, 2014, pp. 691–702.