Proceedings of the Twelfth International
Symposium on Combinatorial Search (SoCS 2019)

A Profit Guided Coordination Heuristic for Travelling Thief Problems

Majid Namazi,[1,2] **M. A. Hakim Newton,**[1] **Abdul Sattar,**[1] **Conrad Sanderson**[2,3]
[1]Griffith University, Australia
[2]Data61 / CSIRO, Australia
[3]University of Queensland, Australia

Abstract

The travelling thief problem (TTP) is a combination of two interdependent NP-hard components: travelling salesman problem (TSP) and knapsack problem (KP). Existing approaches for TTP typically solve the TSP and KP components in an interleaved fashion, where the solution to one component is held fixed while the other component is changed. This indicates poor coordination between solving the two components and may lead to poor quality TTP solutions. For solving the TSP component, the 2-OPT segment reversing heuristic is often used for modifying the tour. We propose an extended and modified form of the reversing heuristic in order to concurrently consider both the TSP and KP components. Items deemed as less profitable and picked in cities earlier in the reversed segment are replaced by items that tend to be equally or more profitable and not picked in the later cities. Comparative evaluations on a broad range of benchmark TTP instances indicate that the proposed approach outperforms existing state-of-the-art TTP solvers.

Introduction

Many real-world constraint optimisation problems (Rossi, Van Beek, and Walsh 2006), such as supply chain management, comprise multiple interdependent components (Bonyadi, Michalewicz, and Barone 2013). This interdependency makes solving these problems very challenging: finding an optimal solution to each component separately does not guarantee finding an optimal solution to the whole problem (Michalewicz 2012).

The travelling thief problem (TTP) (Bonyadi, Michalewicz, and Barone 2013; Polyakovskiy et al. 2014) combines two interdependent NP-hard components: the travelling salesman problem (TSP) (Gutin and Punnen 2006) and the knapsack problem (KP) (Kellerer, Pferschy, and Pisinger 2004). In TTP, a thief makes a *cyclic tour* through each given city and using a *picking plan*, picks a subset of available items into a rented knapsack with limited capacity. As items are picked up at each subsequent city, the total profit and weight of the items in the knapsack increases, while the speed of the thief decreases, thereby increasing travelling time and hence the cost of renting the knapsack. The goal in TTP is to simultaneously maximise the total profit of the picked items and minimise the knapsack's renting cost. TTP is a good proxy for many real-world logistics problems (Mei, Li, and Yao 2014).

Existing TTP algorithms typically fall into three main categories (Wagner et al. 2018; Nieto-Fuentes, Segura, and Valdez 2018): **(i)** constructive methods, **(ii)** cooperative methods, and **(iii)** full encoding methods. In constructive methods, such as Insertion (Mei, Li, and Yao 2014) and Packiterative (Faulkner et al. 2015), after finding an initial TSP tour, the picking plan for KP is computed by using scores given to the items based on their profit, weight and position in the TSP tour. These methods are used in restart-based algorithms such as S5 (Faulkner et al. 2015) and in the initialisation phase of more complex methods. In cooperative methods, such as CoSolver (Bonyadi et al. 2014; El Yafrani and Ahiod 2015) and CS2SA* (El Yafrani and Ahiod 2018), the TSP and KP components are solved in an interleaved fashion. Whenever one component is being solved, the other component is considered fixed. In full encoding methods, such as MATLS (Mei, Li, and Yao 2014), MMAS (Wagner 2016), and J2B (El Yafrani and Ahiod 2017), TTP is considered as a whole. However, whenever the TSP component is being solved, the KP picking plan is considered fixed or changed slightly and randomly. This indicates poor coordination between the methods solving the two components and may lead to poor quality solutions.

The 2-OPT segment reversing heuristic (Croes 1958) is often used for modifying the TSP component solution during the search. We propose an extended and modified form of the reversing heuristic in order to explicitly consider both the TSP and KP components at the same time. Items deemed as less profitable and picked in cities earlier in the reversed segment are replaced by items that tend to be equally or more profitable and not picked in the later cities in the segment. Comparative evaluations on a broad range of benchmark TTP instances indicate that the proposed approach outperforms the MATLS, S5 and CS2SA* solvers.

Background

A TTP instance has a set $\{1, \ldots, n\}$ of n cities and a set $\{1, \ldots, m\}$ of m items. Each *item* i is located at *city* $l_i > 1$. Furthermore, each item has *weight* $w_i > 0$ and associated *profit* $\pi_i > 0$. The *distance* between each pair of cities $j \neq j'$ is $d(j, j') = d(j', j)$. The thief starts a *cyclic tour* from city 1,

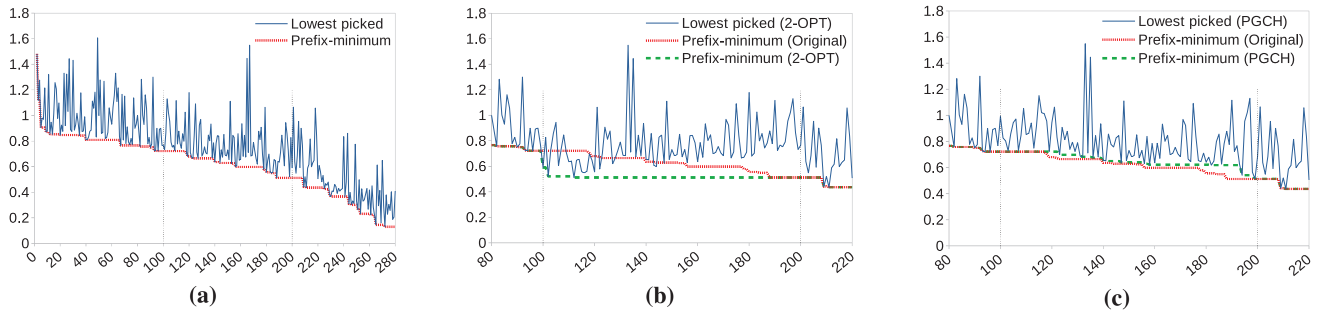

Figure 1: *x*-axis: position in a cyclic tour; *y*-axis: profitability ratio. **(a)**: Lowest picked profitability ratios and corresponding prefix-minimum values in a solution found by MATLS (Mei, Li, and Yao 2014) on instance *a280_n2790_uncorr_10*. **(b)**: Effect of applying 2-OPT, where the segment between positions 100 and 200 is reversed. **(c)**: Effect of applying the proposed PGCH.

visits each city, and finally returns to city 1. We use a permutation c of n cities to represent the cyclic tour. Given a cyclic tour c, let $c_k = j$ denote that the k-th city is j, and $c(j) = k$ denote that the position of city j is k; as such, $c_1 = 1$ and $c(1) = 1$. Let $\ddot{k} = (k \mod n) + 1$ be the next position after k in the cyclic tour. The thief rents a knapsack with a weight capacity W and a rate R of rent per unit time to carry the picked items. Assume that $p_i \in \{1, 0\}$ denotes whether an item i is either picked or not picked under a *picking plan* p. A TTP solution that comprises a cyclic tour c and a picking plan p is denoted as $\langle c, p \rangle$.

We define $W_{c,p}(k) = \sum_{c(l_i) \le k} w_i p_i$ as the total weight of the items picked from the first k cities of the cyclic tour c. The thief travels from city c_k to the next city $c_{\ddot{k}}$ with a speed $v_{c,p}(k)$ that decreases as $W_{c,p}(k)$ increases from one city to the next. We define $v_{c,p}(k) = v_{\max} - W_{c,p}(k) \times (v_{\max} - v_{\min})/W$, where $v_{\max}$ and $v_{\min}$ are the given maximum and minimum speeds, respectively. Given a cyclic tour c and a picking plan p, the total profit is $P(p) = \sum_{i=1}^{m} p_i \pi_i$ and the total travelling time is $T(c, p) = \sum_{k=1}^{n} d(c_k, c_{\ddot{k}})/v_{c,p}(k)$. The goal of TTP is to maximise the *objective function* $G(c, p) = P(p) - R \times T(c, p)$ over any possible c and p. In other words, the goal is to maximise the total profit and minimise the total renting cost of the knapsack.

We define the *profitability ratio* for each item i as $r_i = \pi_i / w_i$. We define relations such as an item i is *more profitable* than item i', if $r_i > r_{i'}$ or if $r_i = r_{i'}$ and $\pi_i > \pi_{i'}$. The profitability relations 'more or equal', 'less', 'less or equal', and 'least' are defined analogously.

In many approaches for solving TTP, a segment reversing heuristic known as the 2-OPT move (Croes 1958) is often used for modifying the TSP component solution. We define the corresponding **2-OPT**(c, k', k'') function as follows. Given a cyclic tour c as well as two positions k' and k'' (with $1 < k' < k'' \le n$), the order of the cities in between the two positions is reversed to obtain a cyclic tour c'. So $c'_{k'+k} = c_{k''-k}$ is obtained for $0 \le k \le k'' - k'$.

Proposed Coordination Heuristic

In the above definition of TTP, an item picking plan is required. A greedy constructive heuristic in KP picks items in a non-increasing order of their profitability ratios (Dantzig 1957). However, the items are dispersed over the cities and their picking order is restricted by the order of the cities in the TSP tour. This suggests that no monotonous item picking ordering should be expected in TTP. Constructive methods such as Insertion (Mei, Li, and Yao 2014) and Packiterative (Faulkner et al. 2015) combine profitability ratios of the items with the distances of the respective cities from the end of the TSP tour.

Considering the above, let us observe the picking orders of items in the solutions returned by the state-of-the-art MATLS solver (Mei, Li, and Yao 2014). For a given solution $\langle c, p \rangle$, we examine the least profitable item $\widehat{i}$ picked at each city c_k and plot its profitability ratio $R_{c,p}(k) = r_{\widehat{i}}$. If no item is picked at a given city c_k, we use $R_{c,p}(k) = 1 + \max_i r_i$, where $\max_i r_i$ is the maximum r among all the items, as the default maximum value.

Fig. 1(a) shows the lowest picked profitability ratios for the *a280_n2790_uncorr_10* benchmark instance (see the Experiments section for details on benchmark instances). From the start to the end of the cyclic tour, the overall trend is a decrease in the lowest picked profitability ratios.

To capture the declining trend of the profitability of the least profitable picked items, we use a prefix-minimum function (Yu, Lin, and Wang 2008) that returns the profitability ratio of the least profitable item picked so far in the tour. For a given TTP solution $\langle c, p \rangle$, a *prefix-minimum function* is defined as $\Pi_{c,p}(k) = \min(\Pi_{c,p}(k-1), R_{c,p}(k))$, with $\Pi_{c,p}(1) = 1 + \max_i r_i$. Fig. 1(a) shows the prefix-minimum values corresponding to the lowest picked profitability ratios.

Suppose that 2-OPT is applied on a segment between positions 100 and 200 of the TSP tour shown in Fig. 1(a). The relevant part of the resultant tour is shown in Fig. 1(b), where the cities between the two positions are reversed. This results in lower prefix-minimum values than the original values at most of the positions in the segment. As such, the overall trend of the lowest picked profitability ratios in the reversed segment is rising, in contrast to the original segment and the whole tour. This reversal in trend can make such a 2-OPT move counterproductive with respect to the TTP objective function. As the picking plan is not changed, the trade-off between profit and renting cost is poor for the low profitable items picked from the start of the reversed segment. In comparison to the original segment, the poor trade-off can result

Algorithm 1 Coordinated TTP (CTTP) solver using the proposed PGCH. In TSPSolver(), n is the number of cities in the tour, while k' and k'' indicate the starting and ending point, respectively, of the segment to be reversed. Each segment must have at least two cities. The position of city 1 is fixed as the first position in the tour.

```
proc CTTPSolver()
$\langle c_*, p_* \rangle \leftarrow \emptyset$   {best solution}
while not global-timeout do
  $c \leftarrow$ ChainedLKTour()
  $p \leftarrow$ InitPickingPlan($c$)
  $\langle c, p \rangle \leftarrow$ TSPSolver($c, p$)
  $p \leftarrow$ KPSolver($c, p$)
  if $G(c, p) > G(c_*, p_*)$
    $\langle c_*, p_* \rangle \leftarrow \langle c, p \rangle$
end while
return $\langle c_*, p_* \rangle$
```

```
proc TSPSolver($c, p$)
$\langle c_*, p_* \rangle \leftarrow \langle c, p \rangle$ {best solution}
repeat
  for $k' \leftarrow 2$ to $n - 1$ do
    foreach $c_{k''} \in$ DelaTriNeighb[$c_{k'}$] with $k' < k'' \leq n$
      $\langle c', p' \rangle \leftarrow$ PGCH($c, p, k', k''$)
      if $G(c', p') - G(c_*, p_*) \geq \alpha \cdot |G(c_*, p_*)|$
        $\langle c_*, p_* \rangle \leftarrow \langle c', p' \rangle$ {new best solution}
  $\langle c, p \rangle \leftarrow \langle c_*, p_* \rangle$
while $\langle c_*, p_* \rangle$ has changed
return $\langle c_*, p_* \rangle$
```

```
proc KPSolver($c, p$)
while not local-timeout do
  $p' \leftarrow$ RandFlipOneItem($p$)
  if $G(c, p') \geq G(c, p)$
    $p \leftarrow p'$
end while
return $p$
{KPSolver() runs for the same length of time as the immediately preceding invocation of TSPSolver()}
```

in a lower objective value for the reversed segment, which in turn can cause the move to be rejected by a solver.

We propose to minimise the renting cost of the knapsack for picking less profitable items and maximise the profit of the picked items by raising the prefix-minimum values of the lowest picked profitability ratios (a max-min strategy). To accomplish this, we propose an extended and modified form of the 2-OPT move, denoted as Profit Guided Coordination Heuristic (PGCH). In addition to reversing the tour segment, the picking plan is changed by aiming to raise or restore the original prefix-minimum values. Items deemed as less profitable and picked in cities earlier in the reversed segment are replaced by items that tend to be equally or more profitable and not picked in the later cities in the segment. The original prefix-minimum values at the given tour positions act as the reference to decide which items are unpicked and picked. Applying PGCH on the reversed segment in Fig. 1(b) results in improved prefix-minimum values shown in Fig. 1(c).

We formally define **PGCH**(c, p, k', k'') as follows. Given a TTP solution $\langle c, p \rangle$ as well as positions k' and k'' (where $1 < k' < k'' \leq n$), a modified solution denoted as $\langle c', p' \rangle$ is obtained. Initially, p' is set to p and the cyclic tour c' is obtained such that $c'_{k'+k} = c_{k''-k}$ for $0 \leq k \leq k'' - k'$. Then, for each $k' \leq k \leq k''$, each item $i : p'_i = 1$ from city $c'_k = l_i$ is unpicked (ie. p'_i is set to 0) if $r_i < \Pi_{c,p}(k)$. Furthermore, for each $k'' \geq k \geq k'$, each item $i : p'_i = 0$ from city $c'_k = l_i$ is picked (ie. p'_i is set to 1) if $r_i \geq \Pi_{c,p}(k)$, provided that the total weight of the newly picked items is not larger than the total weight of the unpicked items in the reversed segment.

Note that if there is a locally increasing trend of the lowest picked profitability ratios in the chosen tour segment, then the prefix-minimum values by definition stay the same over the segment. In such a case, no item is unpicked or picked by PGCH and hence it acts like a typical 2-OPT move.

Coordinated Solver

The proposed PGCH move is employed in the Coordinated TTP (CTTP) solver shown in Algorithm 1. In the function CTTPSolver(), an initial TSP tour is found using the well-known Chained Lin-Kernighan (CLK) heuristic (Applegate, Cook, and Rohe 2003) via the ChainedLKTour() function. For the KP component, the InitPickingPlan() function returns an initial picking plan, which is the best plan (ie. obtaining the largest TTP objective value) out of the plans provided by the Insertion (Mei, Li, and Yao 2014) and Packiterative (Faulkner et al. 2015) methods. The initial solution $\langle c, p \rangle$ is then refined through the functions TSPSolver() and KPSolver(). The refined solution replaces the best known solution if it has a larger objective value. The entire process is iteratively restarted until a global time limit is reached.

The TSPSolver() function behaves as follows. Given a TTP solution $\langle c, p \rangle$, for each position k', the precomputed Delaunay triangulation (Delaunay 1934) neighbourhood for city $c_{k'}$ is considered, as done by other TSP algorithms (El Yafrani and Ahiod 2018). For each city $c_{k''} : k' < k'' \leq n$ in the neighbourhood specified by DelaTriNeighb[$c_{k'}$], the proposed PGCH move is then applied to obtain a new candidate solution $\langle c', p' \rangle$. If the objective value of the new candidate solution is sufficiently larger (with a margin empirically quantified as $\alpha = 0.01\%$ of the best solution), it is accepted as the new best solution. This approach reduces the search space to more promising solutions, and hence reduces the amount of time taken by TSPSolver(). After checking all positions in the tour, the current solution $\langle c, p \rangle$ is replaced by the best solution $\langle c_*, p_* \rangle$. As such, only the best PGCH move takes effect and changes both c and p. If the best solution has changed, the check of all positions is repeated with the new current solution. Otherwise, the best solution is returned.

In the KPSolver() function, as done by Faulkner et al. (2015), the picking state p_i is flipped for a randomly selected item i via the RandFlipOneItem() function. The change is accepted if it results in a larger objective value. The KPSolver() function is run for the same length of time as the preceding invocation of the TSPSolver() function. This allows equal opportunities for TSPSolver() and KPSolver() to improve the objective value. Instead of spending more time in KPSolver(), the whole search is restarted to try more cyclic tours. This is preferable if a limited amount of time is available to solve a given TTP (Faulkner et al. 2015).

Experiments

Two sets of experiments were performed. In the first set, we compare the CTTP solver (employing the proposed PGCH move) against the following solvers: MATLS (Mei, Li, and Yao 2014), S5 (Faulkner et al. 2015) and CS2SA* (El Yafrani and Ahiod 2018). The MATLS and S5

Table 1: Performance comparison of the CTTP solver against MATLS, S5 and CS2SA*. Performance is reported in terms of the relative deviation index (RDI), expressed as percentage, on 3 categories of TTP instances. Category A: 1 item in each city; profits and weights of items are strongly correlated; knapsack capacity is relatively small. Category B: 5 items in each city; profits and weights of items are uncorrelated; weights of items are similar to each other; knapsack capacity is moderate. Category C: 10 items in each city; profits and weights of items are uncorrelated; knapsack capacity is high.

Instance	**Category A**				**Category B**				**Category C**			
	MATLS	**S5**	**CS2SA***	**CTTP**	**MATLS**	**S5**	**CS2SA***	**CTTP**	**MATLS**	**S5**	**CS2SA***	**CTTP**
eil76	72.5	74.3	36.4	**89.3**	**96.1**	70.9	40.9	89.2	**99.3**	77.4	66.8	94.6
kroA100	**92.9**	0.0	18.3	46.9	**85.6**	11.1	4.7	72.3	98.6	99.2	50.0	**100.0**
ch130	49.7	77.8	43.1	**90.3**	**95.3**	76.2	21.8	**95.3**	85.0	54.3	16.8	**90.7**
u159	61.5	80.1	49.4	**100.0**	86.9	79.6	67.3	**94.1**	32.2	30.3	23.1	**86.8**
a280	72.4	94.9	43.4	**96.2**	73.5	62.0	30.5	**97.9**	90.4	**99.6**	37.5	99.4
u574	68.1	91.2	26.3	**94.8**	85.4	84.0	33.1	**99.8**	**94.2**	89.5	39.8	91.8
u724	44.7	85.2	17.6	**94.8**	59.5	62.8	44.1	**82.8**	70.0	70.1	27.0	**91.2**
dsj1000	92.4	2.6	**100.0**	**100.0**	55.0	62.8	21.1	**89.8**	66.7	94.2	49.3	**98.7**
rl1304	51.8	83.1	45.5	**94.7**	72.4	70.0	35.0	**96.9**	80.2	80.4	45.4	**97.7**
fl1577	54.3	84.3	15.6	**90.6**	85.9	90.9	57.0	**98.4**	94.9	92.4	46.3	**97.0**
d2103	1.1	85.6	69.8	**98.5**	33.0	37.9	27.6	**96.3**	28.8	21.7	24.7	**95.0**
pcb3038	41.3	90.9	19.3	**96.7**	47.8	68.2	34.0	**93.4**	72.3	79.6	42.4	**90.6**
fnl4461	34.6	94.5	6.0	**96.3**	85.0	89.6	58.3	**97.4**	94.5	91.9	27.7	**97.4**
pla7397	70.1	95.8	39.1	**96.7**	79.2	86.8	49.7	**94.8**	79.6	74.9	56.0	**97.5**
rl11849	29.5	93.6	9.4	**94.7**	59.1	67.7	35.2	**86.3**	55.5	51.1	23.5	**82.0**
usa13509	42.2	95.9	38.4	**96.3**	68.6	70.6	34.4	**90.4**	45.7	34.6	62.5	**83.1**
brd14051	44.0	**96.5**	20.9	95.3	75.4	79.3	53.3	**95.1**	61.3	65.7	57.5	**88.5**
d15112	3.0	73.0	22.5	**89.8**	21.5	32.4	58.0	**87.5**	33.8	18.1	62.9	**84.4**
d18512	53.9	**97.3**	21.6	95.6	82.5	84.1	57.8	**93.5**	69.4	60.7	53.0	**87.2**
pla33810	25.6	86.3	28.0	**92.7**	59.9	48.1	25.2	**88.3**	74.1	46.6	28.3	**88.4**
Average	50.3	79.2	33.5	**92.5**	70.4	66.7	39.5	**92.0**	71.3	66.6	42.0	**92.1**

solvers were selected based on their notable performance reported by Wagner et al. (2018), while CS2SA* was selected due to its recency.

We use a broad subset of benchmark instances introduced by Polyakovskiy et al. (2014), which are placed into 3 categories as per El Yafrani and Ahiod (2018). Each category has 20 instances with a range of 76 to 33810 cities. In category A, there is only one item in each city; the profits and weights of the items are strongly correlated; knapsack capacity is relatively small. In category B, there are 5 items in each city; the profits and weights of the items are uncorrelated; the weights of the items are similar to each other; knapsack capacity is moderate. In category C, there are 10 items in each city; the profits and weights of the items are uncorrelated; knapsack capacity is high.

The source codes for MATLS, S5 and CS2SA* solvers were obtained from the respective authors. The same experiment setup was used for all solvers. All solvers were independently run on each TTP instance 10 times. Each run had a standard 10-minute timeout. For each run, we ensured that each solver computes new initial cyclic tours via the CLK heuristic whenever required.

For each algorithm on each TTP instance, we use the relative deviation index (Kim and Kim 1996), defined as RDI $= (G_{\text{mean}} - G_{\text{min}}) \times 100/(G_{\text{max}} - G_{\text{min}})$, where G_{mean} is the mean of the $G(c, p)$ objective values of the 10 runs of the algorithm on an instance, while G_{min} and G_{max} are the minimum and the maximum $G(c, p)$ values, respectively, obtained by any run of any algorithm on the same instance.

The performance of all solvers in terms of RDI is given in Table 1. In all categories, the CTTP solver outperforms the other solvers in a vast majority of the instances. This indicates that the explicit coordination in solving the TSP and KP components, as provided by the proposed PGCH move, is indeed beneficial.

In the second set of experiments, the effects of the proposed PGCH move are gauged in more detail. Two versions of Algorithm 1 are used: (i) with the PGCH move, and (ii) with the 2-OPT move replacing the PGCH move. 3 instances from each category are used as representatives of small, moderate and large instances.

Table 2 shows the average $G(c, p)$ values obtained by each of the two heuristics over 10 runs, the average size of the moves (length of the reversed segment) that improved the $G(c, p)$ values when applied, and the relative improvement (in %) by TSPSolver() with respect to $G(c, p)$ obtained by InitPickingPlan() in Algorithm 1. In all cases, PGCH outperforms 2-OPT in obtaining higher $G(c, p)$ values.

The improvement is obtained by using larger moves made in the TSP tour which are accepted due to the corresponding adjustment of the picking plan. In contrast, such large moves are typically not accepted when using the 2-OPT move, as they do not improve the objective value. This is consistent with the observations made by El Yafrani and Ahiod (2018), where a TSP solver employing the 2-OPT move resulted in only minor improvements to the TTP objective value.

Table 2: Comparing the effects of PGCH with 2-OPT. $|k''- k'+1|$ is the size of the move (length of the reversed segment). $\Delta = (G_{\text{x}} - G_{\text{init}}) \times 100/G_{\text{init}}$, where G_{x} is $G(c, p)$ obtained by either PGCH or 2-OPT, while G_{init} is $G(c, p)$ obtained by the InitPickingPlan() function in Algorithm 1.

Cat.	Instance	Move	$G(c, p)$	$\lvert k''- k'+1\rvert$	Δ
A	a280	2-OPT	18446	108	1.81
		PGCH	**18452**	**168**	**5.53**
	fnl4461	2-OPT	262488	503	0.20
		PGCH	**262701**	**901**	**0.38**
	pla33810	2-OPT	1886098	1259	1.62
		PGCH	**1892614**	**6414**	**4.43**
B	a280	2-OPT	109823	3	0.01
		PGCH	**115614**	**174**	**3.83**
	fnl4461	2-OPT	1618268	138	0.01
		PGCH	**1639347**	**1337**	**5.13**
	pla33810	2-OPT	15443569	2443	2.20
		PGCH	**16167283**	**8739**	**7.86**
C	a280	2-OPT	428900	4	0.00
		PGCH	**428926**	**262**	**1.81**
	fnl4461	2-OPT	6543739	165	0.00
		PGCH	**6561287**	**1547**	**2.15**
	pla33810	2-OPT	57615238	4069	0.89
		PGCH	**58337984**	**11540**	**3.83**

Conclusion

Many real-world constraint optimisation problems, such as supply chain problems, comprise two or more interdependent components. Compared to solving a single component (Namazi et al. 2018), the interdependency makes finding good overall solutions considerably more challenging, as finding an optimal solution to each component separately does not guarantee finding an optimal overall solution to the whole problem. In TTP there are two interdependent components: the travelling salesman problem and the knapsack problem. In many TTP solvers, the typical approach is to solve the overall problem in an interleaved fashion: when the solution for one component is changed, the other is fixed.

We have proposed a new heuristic for solving TTPs: whenever a segment in the TSP tour is reversed, the prefix-minimum values of the lowest picked profitability ratios are maintained. Items deemed as less profitable and picked in cities earlier in the reversed segment are replaced by items that tend to be equally or more profitable and not picked in the later cities in the segment. Comparative evaluations show that the proposed approach leads to considerably better performance than state-of-the-art solvers on a broad range of TTP benchmark instances.

References

Applegate, D.; Cook, W.; and Rohe, A. 2003. Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing* 15(1):82–92.

Bonyadi, M. R.; Michalewicz, Z.; Przybylek, M. R.; and Wierzbicki, A. 2014. Socially inspired algorithms for the travelling thief problem. In *Annual Conference on Genetic and Evolutionary Computation*, 421–428.

Bonyadi, M. R.; Michalewicz, Z.; and Barone, L. 2013. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *IEEE Congress on Evolutionary Computation (CEC)*, 1037–1044.

Croes, G. A. 1958. A method for solving traveling-salesman problems. *Operations Research* 6(6):791–812.

Dantzig, G. B. 1957. Discrete-variable extremum problems. *Operations Research* 5(2):266–288.

Delaunay, B. 1934. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk* 7:793–800.

El Yafrani, M., and Ahiod, B. 2015. Cosolver2B: an efficient local search heuristic for the travelling thief problem. In *IEEE/ACS International Conference of Computer Systems and Applications (AICCSA)*, 1–5.

El Yafrani, M., and Ahiod, B. 2017. A local search based approach for solving the Travelling Thief Problem: The pros and cons. *Applied Soft Computing* 52:795–804.

El Yafrani, M., and Ahiod, B. 2018. Efficiently solving the Traveling Thief Problem using hill climbing and simulated annealing. *Information Sciences* 432:231–244.

Faulkner, H.; Polyakovskiy, S.; Schultz, T.; and Wagner, M. 2015. Approximate approaches to the traveling thief problem. In *Annual Conference on Genetic and Evolutionary Computation*, 385–392.

Gutin, G., and Punnen, A. P. 2006. *The Traveling Salesman Problem and Its Variations*. Springer.

Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. Introduction to NP-completeness of knapsack problems. In *Knapsack Problems*. Springer. 483–493.

Kim, J.-U., and Kim, Y.-D. 1996. Simulated annealing and genetic algorithms for scheduling products with multi-level product structure. *Computers & Operations Research* 23(9):857–868.

Mei, Y.; Li, X.; and Yao, X. 2014. Improving efficiency of heuristics for the large scale traveling thief problem. In *Simulated Evolution and Learning, Lecture Notes in Computer Science (LNCS), Vol. 8886*, 631–643.

Michalewicz, Z. 2012. Quo vadis, evolutionary computation? In *Advances in Computational Intelligence, Lecture Notes in Computer Science (LNCS), Vol. 7311*. 98–121.

Namazi, M.; Sanderson, C.; Newton, M. H.; Polash, M.; and Sattar, A. 2018. Diversified late acceptance search. In *Lecture Notes in Computer Science (LNCS), Vol. 11320*, 299–311.

Nieto-Fuentes, R.; Segura, C.; and Valdez, S. I. 2018. A guided local search approach for the travelling thief problem. In *IEEE Congress on Evolutionary Computation (CEC)*, 1–8.

Polyakovskiy, S.; Bonyadi, M. R.; Wagner, M.; Michalewicz, Z.; and Neumann, F. 2014. A comprehensive benchmark set and heuristics for the traveling thief problem. In *Annual Conference on Genetic and Evolutionary Computation*, 477–484.

Rossi, F.; Van Beek, P.; and Walsh, T. 2006. *Handbook of Constraint Programming*. Elsevier.

Wagner, M.; Lindauer, M.; Mısır, M.; Nallaperuma, S.; and Hutter, F. 2018. A case study of algorithm selection for the traveling thief problem. *Journal of Heuristics* 24(3):295–320.

Wagner, M. 2016. Stealing items more efficiently with ants: a swarm intelligence approach to the travelling thief problem. In *Swarm Intelligence, Lecture Notes in Computer Science (LNCS), Vol. 9882*, 273–281.

Yu, H.-I.; Lin, T.-C.; and Wang, B.-F. 2008. Improved algorithms for the minmax-regret 1-center and 1-median problems. *ACM Transactions on Algorithms (TALG)* 4(3):36.