Reformulation and Metaheuristic for the Team Orienteering Arc Routing Problem

Liangjun Ke(✉) and Weibo Yang

The State Key Laboratory for Manufacturing Systems Engineering,
Xian Jiaotong University, Xi'an 710049, China
keljxjtu@xjtu.edu.cn

Abstract. The team orienteering arc routing problem (TOARP) is a relatively new vehicle routing problem. In this problem, a fleet of vehicles are available to serve two sets of customers, each of which is associated with an arc of a directed graph. The customers of the first set are required to be served whereas the ones of the second set are potential and may be not served. Each potential customer is associated with a profit, and the profit can be gained at most once when it is served. The TOARP aims to maximize the total profit gained by serving the customers while each vehicle must start from and end at a depot within a permitted maximum traveling time. This paper shows that the TOARP can be transformed into a team orienteering problem defined on a directed graph. To solve the TOARP, an iterated local search based algorithm is presented. The effectiveness of the proposed algorithm is studied on the benchmark instances.

Keywords: Vehicle routing problem · Metaheuristic · Team orienteering problem · Arc routing problem · Iterated local search

1 Introduction

Routing problem is an essential problem in real world [7]. In this problem, a fleet of vehicles are used to serve a set of customers distributed in a graph. It aims to find optimal routes for the vehicles under some constraints, e.g., costs, demands, and time windows. Under various objectives and constraints, a large number of routing problems have been considered in the literature [11].

The team orienteering arc routing problem (TOARP) is a routing problem which was first formulated in [3]. The TOARP is defined on a directed graph $G = (V, A)$. In this problem, there are two sets of customers. The first set A_R consists of the customers required to be served. The second one A_P consists of the potential customers which may be not served. Each customer is located at an arc of the directed graph. For each arc, there is a time cost t_{ij}. For each potential customer, a profit $p_{ij} \geq 0$ is assigned to it. There are K vehicles. Each vehicle must start from and end at the depot within the time limit T_{max}. The profit gained by serving each potential customer can be accumulated at most once. The goal of the TOARP is to maximize the total profit collected.

Y. Tan et al. (Eds.): ICSI 2017, Part II, LNCS 10386, pp. 494–501, 2017.
DOI: 10.1007/978-3-319-61833-3_52

The TOARP was first solved by a branch-and-cut [3]. Later, a matheuristic was used to deal with the TOARP [1]. This algorithm uses tabu search to explore the neighborhood of the current solution. The solution found during the tabu search phase will be improved by an intensification phase according to the integer linear programming models. The final step of the tabu search phase uses a diversification mechanism with the aim of searching in a totally new sub-region of the solution space.

In this paper, we show that the TOARP can be transformed into a routing problem defined on a new graph, each node of which corresponds to a potential or required arc of the original problem. We notice that some efforts have been devoted to transforming a (capacitated) arc routing problem into a node routing problem [8]. Here we deduce the transformed problem according to the feature of the TOARP. The transformed problem is defined on a new graph where every node corresponds to every potential arc of primal problem. The distance between two arcs is the minimal shortest path distance calculated by Bellman-Ford algorithm. In addition, we propose an iterated local search based algorithm to deal with the transformed problem. The comparison results on the benchmark instances support the performance of the proposed algorithm is competitive.

The remainder of this paper is organized as follows. Section 2 presents the problem transformed from the TOARP. Our proposed algorithm is described in Sect. 3. Section 4 presents the Simulation study. Section 5 concludes our work.

2 A New Formulation of the TOARP

As for the TOARP, we can define a new directed graph $G' = (V', E')$ as follows:

(1) A dummy node 0: It corresponds to arc $(0, 0) \in A$.

(2) The set of nodes V': $V' = \{0, 1, \cdots, m\}$ where m is the number of the required and potential arcs. Each node except node 0 corresponds to a required or potential arc of G. Suppose that node $i \in V' \backslash \{0\}$ corresponds to arc $(n_{is}, n_{ie}) \in A_P \cup A_R$. We can assign a service time s_i and a profit q_i to node i. The service time s_i is defined as

$$s_i = t_{n_{is}, n_{ie}}. \tag{1}$$

The profit q_i is defined as

$$q_i = \begin{cases} p_{n_{is}, n_{ie}} & \text{if } (n_{is}, n_{ie}) \in A_P \\ M & \text{if } (n_{is}, n_{ie}) \in A_R, \end{cases} \tag{2}$$

where M is a sufficient large positive number, say $M = \sum_{b \in A_P} p_b$ (that is, the total profit of all potential arcs in A_P). Accordingly, the nodes in G' corresponding to the required arcs are called the required nodes. Let A'_R be the set of the required nodes.

(3) The set of arcs E': Each arc connects every two nodes in V'. Let us consider two nodes i and $j \in V'$. They correspond to (n_{is}, n_{ie}) and $(n_{js}, n_{je}) \in A_P \cup A_R$

respectively. Let $\mathfrak{P}$ is the shortest path through a subset of arcs in A starting from n_{ie} and ending at n_{js}. The traveling time of arc $(i, j) \in E'$, denoted by d_{ij}, is defined as the distance of the shortest path $\mathfrak{P}$. To calculate the distance of path $\mathfrak{P}$, we implement a label based Bellman-Ford algorithm [5] which is a dynamic programming approach.

In this way, we can define a new routing problem: Given a directed graph G' mentioned above, the goal is to determine K routes, each of which is limited by the time budget T_{max} and starts from and ends at node 0 and visits nodes in V', such that the total collected profit is maximized. It is assumed that the profit of a visited node can be collected at most once. According to the definition of the TOP, one can notice that the routing problem is a TOP defined on a directed graph. In the following, the transformed problem is denoted as DG-TOP.

3 The Proposed Algorithm

In this paper, we present an iterated local search [9] based algorithm to solve the DG-TOP. Iterated local search (ILS) is a singleton search technique, which evolves an incumbent solution over time. Let $\mathbf{s}_b$ and $\mathbf{s}_c$ be the best-so-far and incumbent solution respectively. ILS works as follows: At first, $\mathbf{s}_c$ is initialized and improved by local search, and let $\mathbf{s}_b = \mathbf{s}_c$. After that, four steps are repeated until a termination condition is satisfied. At the first step, a new incumbent solution $\mathbf{s}_c$ is obtained by perturbation. At the second step, $\mathbf{s}_c$ is improved by local search. At the third step, the best-so-far solution $\mathbf{s}_b$ will be replaced by $\mathbf{s}_c$ if $F(\mathbf{s}_c) \geq F(\mathbf{s}_b)$ (i.e., $\mathbf{s}_c$ is better than $\mathbf{s}_b$) where F is the objective value. At the fourth step, the incumbent solution $\mathbf{s}_c$ will be updated according to the solution acceptation criterion. The main procedure of ILS is described in Algorithm 1.

Algorithm 1. The main procedure of ILS

Input: a DG-TOP instance to be solved
Output: the best-so-far solution $\mathbf{s}_b$
1: initialize the incumbent solution $\mathbf{s}_c$ and improve it by local search
2: set $\mathbf{s}_c$ to $\mathbf{s}_b$
3: **while** the stopping condition is not reached **do**
4: obtain a new solution $\mathbf{s}$ by performing a perturbation operator on $\mathbf{s}_c$
5: perform local search on $\mathbf{s}$
6: replace $\mathbf{s}_b$ by $\mathbf{s}$ if $F(\mathbf{s}) \geq F(\mathbf{s}_b)$
7: update $\mathbf{s}_c$ according to the solution acceptation criterion
8: **end while**

3.1 Solution Initialization

The incumbent solution is initialized by two steps. The first step only considers the required nodes. The second step tries to insert some non-required nodes.

The first step constructs a solution as follows: Each route starts from node 0. Afterwards, they are probed in ascending order of their travel time. For a route, suppose its last node is i, an unvisited node is said to be feasible if it obeys the inequality $T(i) + d_{ij} + d_{j0} \leq T_{max}$ where $T(i)$ is the total travel time of the partial route from node 0 to i. Let C be the set of the feasible required nodes. If C is empty, then this route is finished and the next route is tried. Otherwise, a node is chosen from C according to the following probability

$$P(j) = \frac{\frac{1}{d_{ij}}}{\sum_{k \in C} \frac{1}{d_{ik}}}, \quad \forall j \in C \tag{3}$$

If there are still some unvisited required nodes after using the above procedure, the following procedure will be adopted. Firstly, γ required nodes are removed from the routes generated by the above procedure. Then, all unvisited required nodes are inserted based on regret value (that is, the nodes are inserted one by one. At each step, the node with the largest regret value is inserted). The concept of regret value has been adopted in [10]. This procedure is repeated until all required nodes are visited.

The second step repeats the following procedure until no non-required node can be inserted: Let C_{nr} be the set of the remaining non-required nodes which is feasible to be inserted. For each node $k \in C_{nr}$, we define a preference value as $\varphi_k = q_k \delta_k$ where δ_k is the regret value of node k. That is, a node with larger profit and regret value is more desirable. A node $j \in C_{nr}$ is selected according to the probability

$$P(j) = \frac{\varphi_k}{\sum_{k \in C_{nr}} \varphi_k}, \quad \forall j \in C_{nr} \tag{4}$$

3.2 Perturbation

Perturbation aims to make the algorithm search in a new area of the solution space. A remove-insert-based and exchange-based operators are used to disturb the incumbent solution. The remove-insert-based removes γ_1 nodes and inserts the removed required back one by one based on regret value. The exchange-based operator only works on the visited nodes. It repeats the following procedure γ_2 times: randomly choose two nodes from different routes and exchange their positions. If the resulting solution is infeasible, some visited non-required nodes will be removed based on preference value until the solution becomes feasible.

3.3 Local Search

Local search tries to find a better solution in the neighborhood of a starting solution. However, searching in a larger neighborhood is usually time-consuming. To accelerate local search, the mechanism of "do not look bits" [4] is adopted.

We use a so-called *active list*, denoted by AL, to record those active nodes. The active list initially consists of all visited nodes. The active nodes are checked in a random order. Given an active node i, suppose its candidate list is CL. For a node j in CL, three possible cases may occur:

(1) both i and j belong to the same route: a 2-opt move will be used. The resulting solution will be accepted once the total travel time is shortened.
(2) i and j belongs to two different routes: the best move between an exchange move and a relocation move is used. The resulting solution will be accepted once the total travel time is shortened.
(3) j is an unvisited node: at first, a relocation move is used, the resulting solution will be accepted if it is feasible. In this case, the total profit will be increased. Otherwise, i and j are tried to exchange. Let the forward node and backward node of i be i_f and i_b respectively, then the resulting solution will be accepted if it is feasible and if $\frac{p_j}{d_{i_f,j}+d_{j,i_b}-d_{i_f,i_b}} \geq \frac{p_i}{d_{i_f,i}+d_{i,i_b}-d_{i_f,i_b}}$.

If a better solution is found, node i, node j, and their forward and backward nodes in their corresponding routes are added into an auxiliary list. If all active nodes have been tried, the active node list is replaced by the auxiliary list. The procedure is repeated until the auxiliary list is empty.

3.4 Solution Acceptation Criterion

A simulated annealing like criterion [9] is used. It permits some inferior solutions to replace the old incumbent solution with a small probability, which is beneficial to preserve diversity. Formally, let the new solution be $\mathbf{s}$, then

$$\mathbf{s}_c = \begin{cases} \mathbf{s} & \text{if } F(\mathbf{s}) \geq F(\mathbf{s}_b) \\ \mathbf{s} & \text{if } \exp(\frac{F(\mathbf{s})-F(\mathbf{s}_b)}{\theta F(\mathbf{s}_b)}) \geq \varepsilon \\ \mathbf{s}_b & \text{otherwise} \end{cases} \tag{5}$$

where ε is a random number generated from [0, 1]. θ is a parameter.

4 Simulation Study

To study the performance of the proposed algorithm (ILS), we implemented it in C++ and tested it on a PC equipped with Pentium 4, 2.4 GHz CPU, and 4 GB RAM. For each instance, ILS was stopped when one of the following conditions is satisfied: (1) the number of iterations reaches 40000, (2) the running time reaches 200 s. Parameters γ, γ_1, and γ_2 are an integer randomly sampled from interval $[5, 15]$ at each time. Parameter $\theta = 0.1$. These parameters are determined according to extensive test. We compare ILS with the matheuristic algorithm proposed in [1], denoted by MAT. MAT was performed on a PC with Athlon 64 X2 Dual Core Processor 5600 + 2.89 GHz CPU, and 3.37 GB RAM. The stopping criterion of MAT was set to 30 min.

4.1 Test Instances

We used $D36$, $D64$, and $D100$ in [1] to test the algorithm. Parameter p determines the probability of an arc is declared required when generating instances of

the TOARP. For each instance of RPP in [6], nine TOARP instances were generated by taking p and K from $\{0, 0.25, 0.5\}$ and $\{2, 3, 4\}$, respectively. In Table 1, the fist four columns present the name of a set, the number of instances, the minimum and maximum number of vertices and arcs, respectively. From columns

Table 1. The information of class D

set	*#Inst*	$\|V\|$	$\|A\|$	$p = 0$		$p = 0.25$		$p = 0.5$	
				$\|A_R\|$	$\|A_P\|$	$\|A_R\|$	$\|A_P\|$	$\|A_R\|$	$\|A_P\|$
$D36$	9	17–36	96–270	0	10–38	2–10	6–30	6–20	4–23
$D64$	9	37–62	264–482	0	27–75	4–21	22–54	11–38	15–37
$D100$	9	68–100	544–846	0	50–121	9–28	37–95	26–64	20–70

Table 2. Computational results obtained by MAT and ILS for class D

Instance				MAT			ILS			
K	p	*Set*	*#Solved*	*Opt*	*Av.Gap*	*Max.Gap*	*Opt*	*Av.Gap*	*Max.Gap*	*Time(s)*
2	0	$D36$	9	9	0.00	0.00	9	0.00	0.00	5.38
		$D64$	9	5	0.48	1.79	**7**	**0.10**	**0.54**	42.28
		$D100$	9	0	2.58	5.40	**3**	**0.76**	**2.02**	168.37
	0.25	$D36$	9	9	0.00	0.00	9	0.00	0.00	4.88
		$D64$	9	5	0.26	1.20	**6**	**0.09**	**0.44**	28.4
		$D100$	9	1	2.92	10.68	**4**	**0.57**	**2.08**	1.58
	0.5	$D36$	9	9	0.00	0.00	9	0.00	0.00	4.88
		$D64$	9	4	1.10	5.12	**6**	**0.25**	**1.20**	34.8
		$D100$	9	1	4.71	12.42	**3**	**1.22**	**2.98**	153.71
3	0	$D36$	9	**9**	**0.00**	**0.00**	8	0.02	0.14	2.57
		$D64$	9	**7**	**0.10**	**0.75**	6	0.26	1.63	17.14
		$D100$	4	**2**	3.31	12.50	1	**1.97**	**5.89**	92.44
	0.25	$D36$	9	8	**0.08**	**0.74**	8	0.17	1.05	5.40
		$D64$	9	6	**0.42**	**1.64**	6	0.48	2.89	41.04
		$D100$	5	**3**	4.14	9.34	2	**1.57**	**5.29**	48.8
	0.5	$D36$	9	9	0.00	0.00	9	0.00	0.00	3.04
		$D64$	9	5	2.05	5.78	**6**	**0.99**	**4.83**	15.89
		$D100$	7	1	20.45	20.07	**4**	**1.87**	**8.02**	70.75
4	0	$D36$	9	9	0.00	0.00	9	0.00	0.00	2.02
		$D64$	5	4	1.42	4.05	4	**1.30**	**3.89**	13.4
		$D100$	2	2	4.58	10.98	2	**4.15**	**8.91**	56.66
	0.25	$D36$	9	9	0.00	0.00	9	0.00	0.00	2.14
		$D64$	7	6	0.68	4.57	6	**0.65**	**4.24**	13.28
		$D100$	4	3	3.02	9.27	**4**	**2.32**	**6.47**	69.13
	0.5	$D36$	9	9	0.00	0.00	9	0.00	0.00	2.16
		$D64$	7	**6**	**1.21**	5.36	5	1.25	**5.22**	12.68
		$D100$	5	3	7.53	22.07	**5**	**3.02**	**9.73**	58.32

5 to 10, the minimum and maximum number of required arcs and potential arcs are shown for different values of p.

4.2 Results

Table 2 reports the results obtained on class D. By varying K, p, and set, 27 combinations are obtained. The column '$\#solved$' reports the number of instances which can be solved to optimality by the branch-and-cut algorithm in [2]. For each algorithm, we report the following results:

- $\#opt$: the number of instances of which an optimal solution can be found,
- $Av.Gap$ and $Max.Gap$: the average and maximum percentage gap of the solution found with respect to the upper bound found by the branch-and-cut algorithm in [2].

Table 2 presents the results obtained on class D. When $K = 2$, ILS can find better solutions for 6 out of 9 combinations. For the other three combinations, ILS and MAT are even. When $K = 3$, in terms of $\#opt$, MAT works better on 4 combinations and ILS works better on 2 other combinations. In terms of $Av.Gap$ and $Max.Gap$, both MAT and ILS work better on 4 different combinations. When $K = 4$, in terms of $\#opt$, MAT works better on 1 combination and ILS works better on 2 other combinations. In terms of $Av.Gap$, ILS works better on 5 combinations and MAT works better on only one combination. In terms of $Max.Gap$, ILS works better on 6 combinations. Therefore, ILS works slightly better than MAT on class D. The maximum time spent by ILS is less than 169 s.

5 Conclusion

The team orienteering arc routing problem aggregates some features of the team orienteering problem and arc routing problem, which incurs extra requirements to the current solution techniques. This paper first transforms the TOARP to a node routing problem, called DG-TOP. The difference between DG-TOP and the traditional TOP are analyzed. According to the characteristics of the DG-TOP, an ILS-based algorithm is proposed to solve the DG-TOP. Special considerations have been devoted to the required nodes. Moreover, the algorithm uses two operators to perturb the incumbent solution. In addition, a fast local search is presented. Based on the experimental results, the proposed algorithm can find promising solutions for the tested instances within short time.

Acknowledgements. The authors would like to thank the anonymous reviewers for their insightful comments. This work was supported by National Natural Science Foundation of China (No. 61573277, 71471158), the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU 15201414), the Fundamental Research Funds for the Central Universities, the Open Research Fund of the State Key Laboratory of Astronautic Dynamics under Grant 2015ADL-DW403, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State

Education Ministry, Natural Science Basic Research Plan in Shaanxi Province of China (No. 2015JM6316). The authors also would like to thank The Hong Kong Polytechnic University Research Committee for financial and technical support.

References

1. Archetti, C., Corberán, Á., Plana, I., Sanchis, J.M., Speranza, M.G.: A matheuristic for the team orienteering arc routing problem. Eur. J. Oper. Res. **245**(2), 392–401 (2015)
2. Archetti, C., Speranza, M.G.: Arc routing problems with profits. In: Arc Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization, pp. 257–284 (2013)
3. Archetti, C., Speranza, M.G., Corberán, Á., Sanchis, J.M., Plana, I.: The team orienteering arc routing problem. Transp. Sci. **48**(3), 442–457 (2013)
4. Bentley, J.J.: Fast algorithms for geometric traveling salesman problems. ORSA J. Comput. **4**(4), 387–411 (1992)
5. Goldberg, A.V., Radzik, T.: A heuristic improvement of the Bellman-Ford algorithm. Appl. Math. Lett. **6**(3), 3–6 (1993)
6. Hertz, A., Laporte, G., Hugo, P.N.: Improvement procedures for the undirected rural postman problem. INFORMS J. Comput. **11**(1), 53–62 (1999)
7. Laporte, G.: Fifty years of vehicle routing. Transp. Sci. **43**(4), 408–416 (2009)
8. Longo, H., De Aragaao, M.P., Uchoa, E.: Solving capacitated arc routing problems using a transformation to the CVRP. Comput. Oper. Res. **33**(6), 1823–1837 (2006)
9. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: framework and applications. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics, pp. 363–397. Springer, New York (2010)
10. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006)
11. Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods, and Applications, vol. 18. SIAM, Philadelphia (2014)