Iterated Local Search Algorithm for the Capacitated Team Orienteering Problem

Aldy Gunawan · Kien Ming Ng · Vincent F. Yu · Gordy Adiprasetyo · Hoong Chuin Lau

1 Introduction

Orienteering Problem (OP) is an NP-hard routing problem in which the objective is to determine a subset of nodes to be visited by a vehicle, and in which order, so that the total collected score from the visited nodes is maximized with respect to the time budget constraint [9]. Numerous extended variants of OP have been studied in the past few years [4]. This work focuses on a variant of OP called the Capacitated Team Orienteering Problem (CTOP).

In CTOP, each node is associated with a demand and a profit. Given a set of homogeneous fleet of vehicles, the main objective is to determine a route for each available vehicle that maximizes the total score or profit with respect to capacity and time budget constraints [2]. Each node can only be visited once by one vehicle. All vehicles start and end at the same node (e.g. depot). In the context of the classical OP, each vehicle refers to a particular path. Team OP is the extension of the OP by considering multiple vehicles or paths.

Archetti et al. [1] propose a branch-and-price algorithm, while Tarantilis et al. [8] propose a hierarchical bi-level search framework to solve the problem. We introduce an algorithm which involves modification of the Iterated Local Search (ILS) metaheuristic in our previous work [5], including applying certain local search operators and using different criteria for generating the ranks of nodes.

A. Gunawan and H.C. Lau
Singapore Management University
E-mail: aldygunawan, hclau@smu.edu.sg

K.M. Ng and G. Adiprasetyo
National University of Singapore
E-mail: isenkm@nus.edu.sg, gordy@u.nus.edu

V.F. Yu
National Taiwan University of Science and Technology
E-mail: vincent@mail.ntust.edu.tw

Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2018), Vienna, Austria, August 28–31, 2018

Algorithm 1 ITERATED LOCAL SEARCH

```
S0 ← INITIAL SOLUTION
S0 ← LOCALSEARCH1
S* ← S0
NOIMPR ← 0
ITER ← 0
while ITER < MAXITER do
    S0 ← PERTURBATION
    if ITER ≤ LOWERLIMIT then
        S0 ← LOCALSEARCH1
    else
        S0 ← LOCALSEARCH2
    end if
    if S0 better than S* then
        S* ← S0
        NOIMPR ← 0
    else
        NOIMPR ← NOIMPR + 1
    end if
    if (NOIMPR = MAXNOIMPROV) then
        S0 ← S*
        NOIMPR ← 0
    end if
    ITER ← ITER + 1
end while
return S*
```

2 Proposed Algorithm and Preliminary Results

We adopt a simple construction heuristic for generating an initial solution [6]. Nodes are sorted based on ratio values (eq. (1) and (2)) and will be inserted one by one into the available paths/vehicles until no further insertion is feasible.

$$ratio_i = \left(\frac{score_i}{demand_i}\right) \tag{1}$$

$$ratio_i = \left(\frac{(score_i)^2}{d_{i0} + ST_i}\right) \tag{2}$$

Here, $ratio_i$, $score_i, demand_i, d_{i0}$ and ST_i represent the ratio of node i, score of node i, demand of node i, distance from node 0 to node i and service time of node i, respectively. In order to improve the initial solution S_0, we implement a metaheuristic based on Iterated Local Search (ILS) that explores the solution space by generating and evaluating the neighbors of S_0. There are two different LOCALSEARCH subroutines in the ILS: LOCALSEARCH1 and LOCALSEARCH2. LOCALSEARCH1 is first applied to S_0 as many times as possible, limited by the parameter LOWERLIMIT, and this is followed by applying LOCALSEARCH2 for (MAXITER−LOWERLIMIT) times. The outline of the ILS algorithm is presented in Algorithm 1.

LOCALSEARCH1 and LOCALSEARCH2 use a set of operators which are run consecutively: SWAP1 (exchange two nodes within one vehicle), SWAP2 (exchange two nodes within two vehicles), 2-OPT (reverse the sequence of certain nodes within one vehicle), MOVE (move one node from one vehicle to another vehicle), INSERT (insert nodes into a vehicle), and REPLACE (replace one scheduled node with one unscheduled node). The main difference between both LOCALSEARCH1 and LOCALSEARCH2 lies in applying SWAP2 and INSERT operators. In LOCALSEARCH1,

both operators will be accepted if they do not violate the constraints and are able to improve the quality of solutions. In LOCALSEARCH2, as long as there is no constraint violation, both operators can be accepted.

We implement an acceptance criterion that ensures a good balance between diversification and intensification of the search. Search would continue from the current found solution until a certain number of non-improving solutions, MAXNOIMPROV, is reached, hence leading to more diversification. After that, only solutions that are better than the best found solution are kept, hence leading to intensification.

The algorithm is coded in C++ and all experiments are executed on an Intel Core i7-4790 3.60 GHz processor CPU with 32 GB of RAM running on Microsoft Windows operating system. Two sets of benchmark instances, Archetti et al. instances [2] and Tarantilis et al. instances [8], are used. Archetti et al. instances are further categorized into three subsets. The first subset contains the instances generated from the Capacitated VRP instances [3] that allow all customers or nodes to be served. The second subset modifies vehicle sizes, capacity and route duration limits. The third subset alters the vehicle size. Tarantilis et al. instances, which are adopted from the large scale Period VRP instances of Pirkwieser and Raidl [7], are divided into three subsets with similar characteristics.

The parameter values are defined as follows: MAXITER = 1000, LOWERLIMIT = 250 and MAXNOIMPROV = 10. Tables 1 summarizes results obtained by ILS for both sets of instances. We use eq. (1) and (2) for sorting nodes, denoted as ILS(R1) and ILS(R2) respectively. Each instance is run five times. The gap values are calculated by comparing our results with the best known solutions. It does not improve any current best known solution, partly because most of the solutions are proven to be optimal. The gap values are less than 1%. In Subset 1 of Archetti et al. instances, both can obtain the best known solutions except for one instance. For Subset 2 with 90 instances, ILS(R1) obtains 70 best known solutions, while ILS(R2) obtains 80 best known solutions, though at the expense of computation time. Finally, both ILS(R1) and ILS(R2) can get 24 best known solutions for Subset 3. For the Archetti et al. instances, the performances of ILS(R1) and ILS(R2) are thus promising.

For Tarantilis et al. instances, ILS(R2) also performs better than ILS(R1) at the expense of computation time. But the results are still worse than that of best known solutions, with the gap values being up to 3.09%. However, ILS(R2) is able to improve one best known solution of instance 14 of Subset 2, from 509 to 602. Our ILS has some limitations especially pertaining to the large computation time. We are currently improving the ILS and the results would be presented during the conference.

Table 1: Results of Archetti et al. and Tarantilis et al. instances

Instances	Number of instances	Average Objective Function Value					Average CPU time (seconds)		
		Best Known	ILS(R1)	Gap(%)	ILS(R2)	Gap(%)	Best Known	ILS(R1)	ILS(R2)
Subset 1	10	1814.20	1814.00	0.01%	1814.00	0.01%	0.22	0.10	0.17
Subset 2	90	295.21	293.08	0.76%	294.32	0.30%	19.52	12.23	15.39
Subset 3	30	728.40	727.90	0.08%	728.40	0.00%	3.59	2.21	3.11
Subset 1	10	5647.20	5643.50	0.06%	5643.50	0.06%	63.26	60.53	62.11
Subset 2	90	1268.87	1209.06	4.11%	1229.63	3.09%	3061.30	3000.21	3102.69
Subset 3	30	3032.50	2894.77	4.51%	2985.77	1.54%	10978.65	9542.47	10711.66

References

1. Archetti, C., Bianchessi, N., Speranza, M.G.: Optimal solutions for routing problems with profits. Discrete Applied Mathematics **161**(4–5), 547–557 (2013)
2. Archetti, C., Feillet, D., Hertz, A., Speranza, M.G.: The capacitated team orienteering and profitable tour problems. Journal of the Operational Research Society **60**(6), 831–842 (2009)
3. Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: N. Christofides, A. Mingozzi, P. Toth, C. Sandi (eds.) Combinatorial Optimization, pp. 315–338. Wiley, Chichester (1979)
4. Gunawan, A., Lau, H.C., Vansteenwegen, P.: Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research **255**(2), 315–332 (2016)
5. Gunawan, A., Lau, H.C., Vansteenwegen, P., Lu, K.: Well-tuned algorithms for the team orienteering problem with time windows. Journal of the Operational Research Society **68**(8), 861–876 (2017)
6. Luo, Z., Cheang, B., Lim, A., Zhu, W.: An adaptive ejection pool with toggle-rule diversification approach for the capacitated team orienteering problem. European Journal of Operational Research **229**(3), 673 – 682 (2013)
7. Pirkwieser, S., Raidl, G.R.: Multilevel variable neighborhood search for period routing problem. In: P. Cowling, P. Merz (eds.) Evolutionary Computation in Combinatorial Optimization - EvoCOP, pp. 226–238. Springer (2010)
8. Tarantilis, C.D., Stavropoulou, F., Repoussis, P.P.: The capacitated team orienteering problem: a bi-level filter-and-fan method. European Journal of Operational Research **224**(1), 65–78 (2013)
9. Vansteenwegen, P., Souffriau, W., Oudheusden, D.V.: The orienteering problem: A survey. European Journal of Operational Research **209**(1), 1–10 (2011)
10. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: Iterated local search for the team orienteering problem with time windows. Computers and Operations Research **36**(12), 3281–3290 (2009)