6-2021

Set Team Orienteering Problem with Time Windows

Aldy GUNAWAN
Singapore Management University, aldygunawan@smu.edu.sg

Vincent F. YU
National Taiwan University of Science and Technology

Andros Nicas SUTANTO
National Taiwan University of Science and Technology

Panca JODIAWAN
National Taiwan University of Science and Technology

Set Team Orienteering Problem with Time Windows

Aldy Gunawan[1], Vincent F. Yu[2,3], Andro Nicus Sutanto[2], and Panca Jodiawan[2]

[1] School of Computing and Information Systems, Singapore Management University, 80 Stamford Road, Singapore aldygunawan@smu.edu.sg
[2] Department of Industrial Management, National Taiwan University of Science and Technology, 43, Sec. 4, Keelung Road, Taipei 106, Taiwan
vincent@mail.ntust.edu.tw,{andronicus.sutanto,pancajodiawan}@gmail.com
[3] Center for Cyber-Physical System Innovation, National Taiwan University of Science and Technology, 43, Sec. 4, Keelung Road, Taipei 106, Taiwan

Abstract. This research introduces an extension of the Orienteering Problem (OP), known as Set Team Orienteering Problem with Time Windows (STOPTW), in which customers are first grouped into clusters. Each cluster is associated with a profit that will be collected if at least one customer within the cluster is visited. The objective is to find the best route that maximizes the total collected profit without violating time windows and time budget constraints. We propose an adaptive large neighborhood search algorithm to solve newly introduced benchmark instances. The preliminary results show the capability of the proposed algorithm to obtain good solutions within reasonable computational times compared to commercial solver CPLEX.

Keywords: orienteering problem · time windows · adaptive large neighborhood search.

1 Introduction

The Orienteering Problem (OP) was first introduced by [9] for which a set of nodes is given, each with a score. The objective is to determine a path, limited in length or travel time, that visits a subset of nodes and maximizes the sum of the collected scores. The Orienteering Problem (OP) has received a lot of attentions since many researchers have worked on it as well as its applications and extensions [2], such as the inventory problem [10], Capacitated Team OP [8], and Set OP [1].

The Set OP (SOP) was presented by [1]. The main difference lies on grouping nodes into clusters and each cluster is associated with a profit. This profit is collected by visiting at least one node in the respective cluster. Due to the time budget constraint, only a subset of clusters can be visited on a path. Various applications of the SOP can be found in mass distributions, yet it may benefit less to visit all customers within a particular district. Therefore, only delivering

products to one customer and letting other customers within the same district to collect from the visited customer will actually help the distributor in terms of travelling time or travelled distance. [1] proposed a matheuristic algorithm and applied it in the context of the mass distribution problem. [7] introduced the applications of the SOP in the travel guide problems. Variable Neighborhood Search (VNS) is proposed to solve SOP. The Team Orienteering Problem with Time Windows (TOPTW) is an extension of the Team OP [5] where the visit on each node is constrained by a given time window. This TOPTW has been studied in the past few years. For more details, please refer to [2].

Our work introduces another extension of the SOP and TOPTW - namely, the Set Team OP with Time Windows (STOPTW). STOPTW considers both multiple paths and time windows. The visits to nodes are also bounded by the given time windows. We thus propose an adaptive large neighborhood search algorithm (ALNS) to solve newly introduced benchmark instances. The preliminary results of our experiments show the capability of the proposed algorithm to generate good solutions within reasonable computational times.

2 Problem Description

Given a complete directed graph $G = (N, A)$ where N represents a set of nodes, $N = \{n_0, n_1, \ldots, n_{|N|}\}$, A represents a set of arcs $A = \{a_{ij}\}$, and n_0 and $n_{|N|}$ are the start and end nodes, respectively. Given a pair of nodes n_i and n_j, there exists an arc a_{ij} with cost c_{ij}. In SOP, all nodes are grouped as clusters as disjoint sets $s_0, s_1, \ldots, s_m$, with $S = \{s_0, s_1, \ldots, s_m\}$, $s_i \cap s_j = \emptyset$ for $i \neq j, 0 \leq i, j \leq |N|$, and each node n_i is associated with exactly one particular set in S. All disjoint sets $s_0, s_1, \ldots, s_m$ have associated profits $p_0, p_1, \ldots, p_m$ for visiting at least one node within the set. We note that s_0 and s_m represent the starting and ending sets with $p_0 = p_m = 0$, respectively. The basic mathematical model of the SOP is presented in [7] while the TOPTW mathematical model can be referred to [2].

In the context of the STOPTW, each node has a non-negative service time s_i and time window $[E_i, F_i]$, where E_i and F_i are the earliest and the latest start times of service at node i, respectively. A visit beyond the time window F_i is not allowed while an early visit is possible with additional waiting times before entering at the earliest start time E_i. Each node i can only be visited once and must be visited within its respective time window. Each set s_j can only be visited at most once as well. Given h paths, each path must start its visit from the start node and also return to the end node. The objective of the STOPTW is to determine h paths, limited by a given time budget T_{max} and time windows, such that each path visits a subset of S and maximizes the total collected profit.

3 Proposed Algorithm

The initial solution is generated based on the nearest distance criterion. Each path h starts from node n_0 and follows up by visiting the nearest unvisited

nodes. This is done until it reaches T_{max} or no more nodes can be visited. The time window constraint has to be considered, and the path ends at node n_0.

The initial solution is further improved by the proposed Adaptive Large Neighborhood Search (ALNS) (Algorithm 1). This proposed algorithm is adopted from a similar algorithm for solving another combinatorial optimization problem, namely the vehicle routing problem [3].

The main idea is to use a set of DESTROY operators for removing nodes from the current solution and to use a set of REPAIR operators for reinserting them into more profitable positions. A particular score is assigned to each selected operator in order to assess its performance upon generating a new neighborhood solution. The better the new generated solution is, the higher is the score given to the corresponding operator.

Let Sol_0, Sol^*, and Sol' be the current solution, the best found solution so far, and the starting solution at each iteration, respectively, we first set Sol_0, Sol^*, and Sol' to be the same as the generated initial solution. The current temperature ($Temp$) is set to the initial temperature (T_0) and will decrease by α after η_{SA} iterations. The number of iterations $Iter$ is set to zero. Let $R = \{R_r | r = 1, 2, \ldots, |R|\}$ be a set of DESTROY operators and $I = \{I_i | i = 1, 2, \ldots, |I|\}$ be a set of REPAIR operators. All operators $j (j \in R \cup I)$ initially have the same weight w_j and probability p_j to be selected, based on:

$$p_j = \begin{cases} \frac{w_j}{\sum_{k \in R} w_k} & \forall j \in R \\ \frac{w_j}{\sum_{k \in I} w_k} & \forall j \in I \end{cases} \tag{1}$$

ALNS adopts the Simulated Annealing (SA) acceptance criteria, under which a worse solution may be accepted with a certain probability [6]. Therefore, each of the operator's score s_j is adjusted by:

$$s_j = \begin{cases} s_j + \delta_1, & \text{if the new solution is the} \\ & \text{best found solution so far} \\ s_j + \delta_2, & \text{if the new solution improves} \\ & \text{the current solution} \\ s_j + \delta_3, & \text{if the new solution does not} \\ & \text{improve the current solution,} \\ & \text{but it is accepted} \end{cases} \quad \forall j \in R \cup I \tag{2}$$

with $\delta_1 > \delta_2 > \delta_3$. The operator's weight w_j is then adjusted by following:

$$w_j = \begin{cases} (1-\gamma) w_j + \gamma \frac{s_j}{\chi_j}, & \text{if } \chi_j > 0 \\ (1-\gamma) w_j, & \text{if } \chi_j = 0 \end{cases} \quad \forall j \in R \cup I \tag{3}$$

where γ refers to the reaction factor ($0 < \gamma < 1$) to control the influence of the recent success of an operator on its weight, and χ_j is the frequency of using operator j.

At each iteration, a certain number of nodes are removed from Sol_0 by using a selected DESTROY operator. The removed nodes are then reinserted into Sol_0 by applying another selected REPAIR operator. Sol_0 is directly accepted if its objective function value is better than Sol^* or Sol'; otherwise, it will only be accepted with probability $e^{\frac{-(Sol_0 - Sol')}{Temp}}$. Each operator's score s_j is then updated according to (2). After η_{ALNS} iterations, each operator's weight w_j is updated by (3), and its probability p_j is updated according to (1). ALNS is terminated when there is no solution improvement after θ successive temperature reductions.

Algorithm 1: ALNS pseudocode

```
Sol_0, Sol*, Sol' ← Initial Solution
Temp ← T_0
Iter ← 0
FoundBestSol ← False
Set s_j and w_j such that p_j is equally likely
while NoImpr < θ do
    RemovedNodes ← 0
    while RemovedNodes < π do
        Sol_0 ← Destroy(R_r)
        UpdateRemovedNodes(RemovedNodes, R_r)
    end
    while RemovedNodes > 0 do
        Sol_0 ← Repair(I_i)
        UpdateRemovedNodes(RemovedNodes, I_i)
    end
    AcceptanceCriteria(Sol_0, Sol*, Sol', Temp)
    Update s_j
    if Iter mod η_ALNS = 0 then
        Update w_j and p_j
    end
    if Iter mod η_SA = 0 then
        if FoundBestSol = False then
            NoImpr ← NoImpr + 1
        end
        else
            NoImpr ← 0
        end
        FoundBestSol ← False
        Temp ← Temp × α
    end
    Iter ← Iter + 1
end
Return Sol*
```

Four DESTROY and six REPAIR operators used in the proposed ALNS are:

Random removal (R_1): select q nodes randomly and remove them from the current solution. REMOVEDNODES is increased by q.

Worst removal (R_2): remove the node with the smallest removal profit. The removal profit is defined as the difference in objective function values between including and excluding a particular node.

Shaw removal (R_3): remove a node that is highly related with other removed nodes in a predefined way. In other words, it tries to remove some similar nodes, such that it is easier to replace the positions of one another during the repair process. The last removed node is denoted as node i, while the next candidate

of the removed node is denoted as node j. The relatedness value (φ_j) of node j to node i is calculated by:

$$\varphi_j = \begin{cases} \phi_1 c_{ij}^{'} + \phi_2 t_{ij}^{'} + \phi_3 l_{ij} + \phi_4 |P_i - P_j|, & \text{if } i \in S \\ \phi_1 c_{ij}^{''} + \phi_2 t_{ij}^{''} + \phi_3 l_{ij} + \phi_4 |D_i - D_j|, & \text{if } i \in C \end{cases} \quad (4)$$

Unvisited removal (R_4): this operator removes selected nodes that are not visited due to the time windows violation. When selecting nodes, random numbers are generated to determine whether they will be removed or not.
Greedy insertion (I_1): insert a removed node to a position resulting in the highest insertion profit (i.e., the difference in objective function value after and before inserting a node to a particular position).
Regret insertion (I_2): the regret value is calculated by the difference in Total Profit when node j is inserted in the best position and in the second best position. The idea is to select a node that leads to the largest regret value if it is not inserted into its best position. In other words, this operator tries to insert the node that one will regret the most if it is not inserted now.
Greedy visit insertion (I_3): the insertion is decided by the changes in the number of visited nodes for every inserted node. Since we consider time windows, after inserting a particular node, there will be some nodes that cannot be visited again. Here, we try to find an insertion with the highest number of visited nodes.
Random insertion (I_4): the insertion is decided by choosing a random position in the current solution and trying to insert any removed nodes into that position.
First feasible position insertion (I_5): this operator is adopted from [4]. Every removed node is inserted into the first position that makes the solution feasible, one at a time.
Last feasible position insertion (I_6): it works similarly to the previous operator. The main difference lies on the position of inserting it. It should start from last node of the feasible solution.

4 Computational Results

We first modified a set of TOPTW instances that are taken from Solomon's dataset - namely, Set A. There are 29 instances ($c100, r100$, and $rc100$) where each instance contains 100 nodes. We group nodes into clusters using a method proposed by [1]. The number of clusters is set to 20% of the total number of nodes. After randomly inserting nodes into each cluster, the cluster profit is calculated by adding all profits from all respective nodes in a particular cluster. Another set of larger instances, Set B, is introduced by modifying the above-mentioned instances. A hundred more nodes are added with respective parameters, such as service times, locations, profits, etc. This experiment is performed on a Windows 7 professional computer with Intel core i7-4790 CPU @3.60 GHz processor with 16.00 GB RAM. AMPL is utilized to run the mathematical programming using CPLEX, while Microsoft Visual C++ 2019 is used to code our ALNS algorithm. The obtained results are compared to those of the commercial software CPLEX (Table 4). The profit and computational (CPU) times are

based on 10 runs of ALNS. We also calculate the gap (%) between CPLEX and ALNS results.

Table 1. Total profit comparison between ALNS and CPLEX when solving Set A

Instance	CPLEX		ALNS		
	Profit	CPU time	Profit	CPU time	Gap (%)
$c100$	1808.89	753.22	1666.44	109.40	2.79
$r100$	1387.92	3300.81	1223.80	109.87	2.58
$rc100$	1601.50	3600	1389.23	108.31	6.11

For solving Set A instances, our proposed ALNS is comparable to CPLEX. Here, our main purpose is to test the current performance of ALNS and emphasize the CPU time, which is much lower than the one of CPLEX. We note that ALNS outperforms CPLEX for 5 instances - namely, r102, r104, r107, r108, and rc104. The average gap in terms of the solution quality is 3.62%. We report the performance of ALNS in solving Set B. The 29 instances are served by four different numbers of vehicles, from one to four vehicles. CPLEX is also used to solve those instances with the maximum CPU times of 2 hours (7200 seconds). The results are summarized in Table 4. We observe that ALNS outperforms CPLEX for solving larger instances. This can be seen from the calculated average gaps, which are -50.63%, -8.5%, -21.53%, and -48.16% for one, two, three, and four vehicles, respectively. For most instances, CPLEX is unable to obtain the optimal solutions, and therefore we only report the best found solutions within 2 hours of CPU time.

Table 2. Total profit comparison between ALNS and CPLEX when solving Set B

Number of vehicles	Instance	CPLEX		ALNS		
		Profit	CPU time	Profit	CPU time	Gap (%)
1	$c100$	243.33	6405.1	282.56	1233.35	-26.85
	$r100$	133	6602.95	201.25	1657.66	-89.15
	$rc100$	224	7200	219	1708.86	-19.6
2	$c100$	561.11	7200	458	972	15.6
	$r100$	313.5	7200	371.75	1198.46	-29.19
	$rc100$	394.13	7200	383.75	1128.79	-4.58
3	$c100$	687.78	7200	646.67	1046.7	3.36
	$r100$	406.92	6682.44	533.08	980.55	-37.98
	$rc100$	466.38	7200	552.75	1032.34	-24.85
4	$c100$	802.22	7200	833.22	893.25	-8.43
	$r100$	451.33	7200	700.75	829.2	-69.26
	$rc100$	483.5	7200	720.75	860.85	-61.2

5 Conclusion

This research introduces the Set Team Orienteering Problem with Time Windows (STOPTW) as a new extension of TOPTW, where customers are grouped into clusters and a profit is associated with each cluster. The profit collection

will only happen if at least one customer is visited in a particular cluster. The objective of STOPTW is to maximize the total collected cluster profit without violating any time windows. We propose an Adaptive Large Neighborhood Search (ALNS) algorithm to solve STOPTW, while CPLEX is used to obtain optimal solutions for comparison purposes. The computational study shows that our algorithm outperforms CPLEX in solving newly larger introduced instances. More development on the algorithm can be considered as future research. Other destroy and removal operators can also be developed to provide better solutions. Since STOPTW is a new problem, other (meta)heuristics can also be considered.

Acknowledgment

The work of Vincent F. Yu was partially supported by the Ministry of Science and Technology of Taiwan under grant MOST 108-2221-E-011-051-MY3 and the Center for Cyber-Physical System Innovation from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

References

1. Archetti, C., Carrabs, F., Cerulli, R.: The set orienteering problem. European Journal of Operational Research **267**, 264–272 (2018)
2. 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)
3. Gunawan, A., Widjaja, A.T., Vansteenwegen, P., Yu, V.F.: Vehicle routing problem with reverse cross-docking: An adaptive large neighborhood search algorithm. In: Lalla-Ruiz, E., Mes, M., Voß, S. (eds.) Computational Logistics. Lecture Notes in Computer Science, vol. 12433, pp. 167–182. Springer (2020)
4. Hammami, F., Rekik, M., Coelho, L.C.: A hybrid adaptive large neighborhood search heuristic for the team orienteering problem. Computers and Operations Research **123**, 105034 (2020)
5. Labadie, N., Mansini, R., Melechovskỳ, J., Calvo, R.W.: The team orienteering problem with time windows: an lp-based granular variable neighborhood search. European Journal of Operational Research **220**(1), 15–27 (2012)
6. Lutz, R.: Adaptive large neighborhood search (2015)
7. Pěnička, R., Faigl, J., Saska, M.: Variable neighborhood search for the set orienteering problem and its application to other orienteering problem variants. European Journal of Operational Research **276**(3), 816–825 (2019)
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. Tsiligirides, T.: Heuristic methods applied to orienteering. Journal of the Operational Research Society **35**(9), 797–809 (1984)
10. Vansteenwegen, P., Mateo, M.: An iterated local search algorithm for single-vehicle cyclic inventory. European Journal of Operational Research **237**(3), 802–813 (2014)