International Journal of Management Science
Vol 17, No 1, May 2011

Priority Rule Based Heuristics for the Team Orienteering Problem

Kyoung-Woon Ha
Department of Industrial Engineering, Hanyang University

Jae-Min Yu
Department of Industrial Engineering, Hanyang University

Jong-In Park
Reliability Technology Center Korea, Testing Laboratory

Dong-Ho Lee*
Department of Industrial Engineering,
Graduate School of Technology and Innovation Management, Hanyang University

(Received: December 17, 2010 / Revised: May 2, 2011 / Accepted: May 2, 2011)

ABSTRACT

Team orienteering, an extension of single-competitor orienteering, is the problem of determining multiple paths from a starting node to a finishing node for a given allowed time or distance limit fixed for each of the paths with the objective of maximizing the total collected score. Each path is through a subset of nodes, each of which has an associated score. The team orienteering problem has many applications such as home fuel delivery, college football players recruiting, service technicians scheduling, military operations, etc. Unlike existing optimal and heuristic algorithms often leading to heavy computation, this paper suggests two types of priority rule based heuristics-serial and parallel ones-that are especially suitable for practically large-sized problems. In the proposed heuristics, all nodes are listed in an order using a priority rule and then the paths are constructed according to this order. To show the performances of the heuristics, computational experiments were done on the small-to-medium sized benchmark instances and randomly generated large sized test instances, and the results show that some of the heuristics give reasonable quality solutions within very short computation time.

Keywords: Logistics, Team Orienteering, Heuristics, Priority Rules

* Corresponding author, E- mail: leman@hanyang.ac.kr

1. Introduction

Orienteering is an outdoor sport played in a mountainous or heavily forested area where players use a map and a compass to navigate from a starting node to a finishing node in diverse and usually unfamiliar terrain. Each location between the starting node and the finishing node has an associated score (reward) and the player collecting the highest scores during his travel wins the game. Unlike ordinary vehicle routing problems, a time or distance limit is given to the players in this game and hence the players may not visit all locations. The orienteering problem is to determine a single path that maximizes the total collected score within the time or distance limit. Similar researches in the literature are the selective traveling salesman problem of Laporte and Martello [12] and the maximum collection problem of Butt and Cavalier [4].

As an extension of the orienteering problem, the team orienteering problem determines multiple paths at the same time, each of which has a time or distance limit. A team consisting of several players starts at a starting node. Each team member tries to visit as many nodes as possible within the time limit and then ends at a finishing node. Once a team member visits a node, no other team members can visit the same node, and hence each member of the team has to select a subset of nodes to be visited in order to maximize the total team score. Many applications for the team orienteering problem can be found in the literature: out-door sport game in Chao *et al*. [6], multi-vehicle version of the home fuel delivery problem in Golden *et al*. [9], recruiting of college football players from high schools in Butt and Cavalier [4], service scheduling of routing technicians in Souffriau *et al*. [15], and tour trip design in Tang and Miller-Hooks [16], etc.

In general, there are four basic decision problems associated with orienteering: orienteering, team orienteering, maximum collection, and multiple tour maximum collection. Figure 1 describes the differences among the four decision problems. Note that the orienteering (team orienteering) problem is different from the maximum collection (multiple tour maximum collection) problem in that the former has distinct starting and finishing nodes while the latter has the same starting and finishing node.

Various exact and heuristic algorithms have been suggested for the orienteering (maximum collection) and the team orienteering (multiple tour maximum collection) problems. Since the orienteering (maximum collection) problem has been proved be

NP-hard by Golden *et al*. [9], most of the existing solution algorithms are heuristics. Tsiligirides [17] proposes deterministic and stochastic heuristics using the Monte Carlo technique, and Golden *et al*. [9] suggest a heuristic algorithm using an insertion method and a center of gravity improvement. See Ramesh and Brown [14] and Chao *et al*. [5] for other heuristics on the orienteering problem. Also, there are several optimal solution algorithms using dynamic programming [10], branch and bound [12, 13], and branch-and-cut [7, 8].

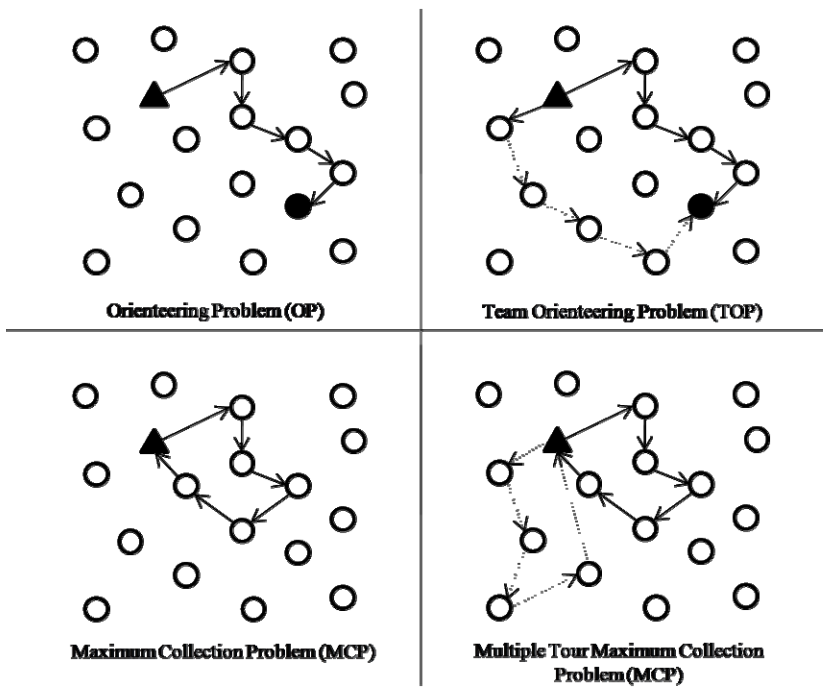

Figure 1. Problems Related with Orienteering

The team orienteering problem (TOP) was firstly studied by Butt and Cavalier [4] where they considered the recruiting problem of college football players while the starting and finishing nodes are located at the same location, i.e., multiple tour maximum collection. In this study, the authors suggest a simple greedy construction heuristic (Note that the team orienteering problem is also NP-hard since its special case, orienteering problem, has been proved to be NP-hard). Later, Chao *et al*. [5] suggest a five-step heuristic with initialization, main movement, clean up, local improvement, and reinitialization. Unlike the construction heuristics, several researchers suggest

various meta-heuristics to obtain better solutions. See Tang and Miller-Hooks [16] and Archetti *et al*. [1] for tabu search, Bouly *et al*. [2] for the memetic algorithm, Ke *et al*. [11] for ant colony optimization, Vansteenwegen *et al*. [18] for guided and iterated local search, and Souffriau *et al*. [15] for path relinking. Finally, unlike the heuristic approach, Boussier *et al*. [3] propose an exact branch-and-price algorithm in order to provide a generic branch-and-price scheme capable of solving different kinds of small-sized instances efficiently.

As explained above, various optimal and heuristic algorithms have been suggested for the TOP that determines multiple paths. However, the existing algorithms may not be suitable for practically large sized instances since they are extremely time consuming to get the final solutions. For example, even the five-step heuristic of Chao *et al*. [6], in our experiments, needed about 14 minutes to give the solutions for the largest test instances with 1000 nodes. Unlike the previous studies on developing optimal algorithms and meta-heuristics, therefore, we suggest fast heuristics that are suitable for practically large-sized instances. For example, the TOP can be applied to real-time decisions that select attack nodes in military operations, especially when making immediate decisions in wartime. In this case, it may not be useful for the exact algorithms and the meta-heuristics to give solutions due to their excessive computation times.

The heuristics suggested in this study are based on priority rules, i.e., all nodes are listed in an order using a priority rule and then the multiple paths are constructed according to this order. Computational experiments were done on the small-to-medium sized benchmark instances and randomly generated large-sized test instantces, and the results are reported. In particular, the heuristics were compared with the five-step heuristic of Chao *et al*. [6].

This paper is organized as follows. The next section describes the problem and presents an integer programming model. Section 3 explains the priority rule based heuristics. Experimental results are reported in Section 4, and the conclusions and further works are discussed in Section 5.

2. Problem Description

Consider a military operation to illustrate the basic concept of the TOP. Figure 2

shows an example of a battle field for a battalion with three companies. As can be seen in the figure, the field has thirty candidate attack nodes (circular nodes) in which each company gets military results (scores) from the starting node (triangular node) to the finishing node (rectangular node). Here, the starting and finishing nodes are called assembly areas or rallying nodes. Also, each candidate attack node can be visited at most once, and there is a time or distance limit for each company. In this military case, the problem is to determine the path for each company in order to maximize the total military result while satisfying the time or distance limit.

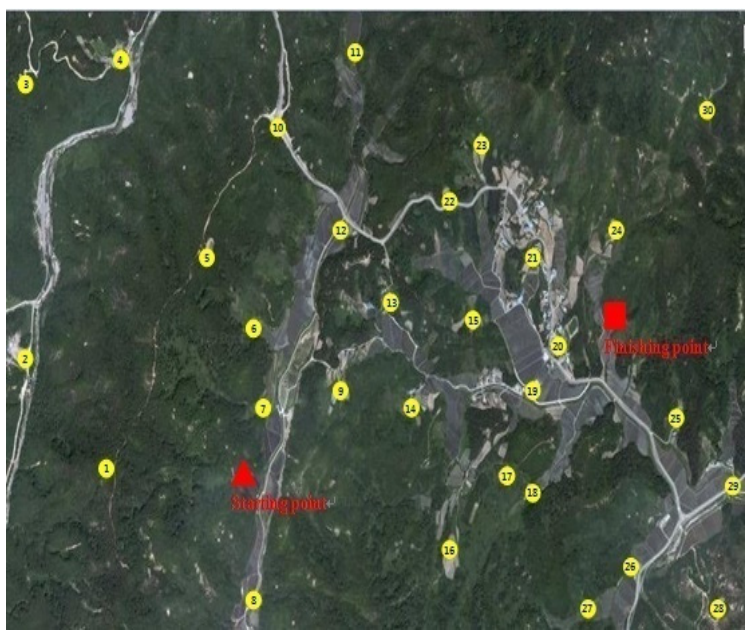

Figure 2. An Example for Team Orienteering: Deployment of Troops

Now, we explain the TOP more formally. Let $G = (N, A)$ be a network, where $N = \{1, 2, \cdots, n\}$ is the set of nodes and $E = \{(i, j): i \neq j\}$ is the set of arcs. Nodes 1 and n denote the starting and the finishing nodes, respectively, whereas the other nodes $V \setminus \{1, n\}$ correspond to the candidate nodes for visiting. Each node, except for the starting and the finishing nodes, is associated with a score and each candidate node is assumed to be visited at most once. A non-negative distance or travel time between two nodes is associated with each arc. We assume that distances or travel times are symmetric and satisfy the triangle inequality. A set of m identical vehicles have a time or

distance limit.

A solution consists of a set of m paths P_1, P_2, $\cdots$ P_m and the problem is to find the solution that gives a maximum score collected. Here, the q_{th} path can be represented as $P_q = (1, i_{q1}, i_{q2}, \cdots, n)$, where i_{qf} denotes the index for the f_{th} node in path q.

Before presenting the integer programming model, the notations used are summarized as follows.

Parameters

c_{ij} travel time (distance) between nodes i and j

s_i score acquired when visiting node i

T_{max} time (distance) limit for each vehicle

U proper subset of nodes in V, i.e., $U \subset V$

Decision variables

y_{ik} = 1 if node i is visited by vehicle k, and 0 otherwise

x_{ijk} = 1 if node j is visited directly after node i by vehicle k, and 0 otherwise

Now, the integer programming model is given below.

Maximize $\sum_{i=2}^{n-1}\sum_{k=1}^{m} s_i \cdot y_{ik}$

subject to

$$\sum_{j=2}^{n}\sum_{k=1}^{m} x_{1jk} = \sum_{i=1}^{n-1}\sum_{k=1}^{m} x_{ink} = m \tag{1}$$

$$\sum_{i<j} x_{ijk} + \sum_{i>j} x_{jik} = 2 \cdot y_{jk} \qquad \text{for all } j\ (\neq 1, n) \text{ and } k \tag{2}$$

$$\sum_{k=1}^{m} y_{ik} \leq 1 \qquad \text{for all } j\ (\neq 1, n) \tag{3}$$

$$\sum_{i=1}^{n-1}\sum_{i<j} c_{ij} x_{ijk} \leq T_{\max} \qquad \text{for all } k \tag{4}$$

$$\sum_{i,j\in U, i<j} x_{ijk} \leq |U| - 1 \qquad \text{for all } U \subset V \setminus \{1, n\}\ (2 \leq |U| \leq n - 2) \text{ and } k \tag{5}$$

$$x_{ijk} \in \{0, 1\} \qquad \text{for all } 1 \leq i < j \leq n \text{ and } k \tag{6}$$

$$y_{ik} \in \{0, 1\} \qquad \text{for all } i\ (\neq 1, n) \text{ and } k \tag{7}$$

$$y_{1k} = y_{nk} = 1 \qquad \text{for all } k. \tag{8}$$

The objective function represents maximizing the total score collected by all vehicles. Constraint (1) ensures that each vehicle starts at node 1 and finishes at node n, and constraint (2) relates to the connectivity of each path. Constraint (3) ensures that all nodes except for nodes 1 and n can be visited at most once. The time (distance) restriction of each vehicle is represented by constraint (4), and sub-paths are prohibited by constraint (5). Finally, constraints (6), (7) and (8) specify the conditions of decision variables.

3. Solution Algorithms

This section gives the details of the two types of priority rule based heuristics-serial and parallel types-proposed in this study.

3.1 Serial Type Heuristics

Serial type heuristics construct multiple paths in a consecutive fashion, i.e., each path is generated one after another. To construct a complete path, a candidate node selected based on a certain priority rule is added to the current path constructed so far while checking if the time (distance) limit is violated. As in Chao *et al*. [6], the candidate node is selected among those within the ellipse constructed by using the start and finishing nodes as the two foci of the ellipse since any path that contains a node outside the ellipse violates the time (distance) limit T_{max}. See Chao *et al*. [6] for more details.

The overall procedure of the serial type heuristic is given below.

Procedure 1: (Serial type heuristic)

Step 1: Set $k = 1$ (k denotes the index for paths).

Step 2: Construct the k_{th} path as follows.

(a) Initialize the path (only the starting and finishing nodes).

(b) Select a candidate node among those unconsidered so far using a priority rule.

(c) If the node can be added to the current path without violating the time (distance) limit, add it to the current path and go to step 2(b). Otherwise, go to Step 3.

Step 3: Set $k = k + 1$. If $k > m$, stop the algorithm. Otherwise, go to Step 2.

In Step 2(b), we propose the following four priority rules.

LS select a node with the largest score value, i.e., max s_j (ties broken by selecting the node with the shortest distance from the last node of the current path).

SD select a node with the smallest increment in distance (ties broken by selecting the node with the largest score).

SS/D select a node with the smallest ratio of the score to the distance, i.e. min s_j / l_k, where l_k is the length of path k after adding the node (ties broken by selecting the node with the largest score).

LS/D select a node with the largest ratio of the score to the distance, i.e. max s_j / l_k (ties broken by selecting the node with the shortest path length)

For the ways the candidate nodes are added to the current path, two types of serial heuristics are employed: nearest neighbor and nearest insertion methods, both are well-known construction heuristics found in the literature for the traveling salesman problem. The nearest neighbor method (S1) adds a selected candidate node to the end of the current path while the nearest insertion method (S2) adds a selected node to the position that gives the minimum increase in the path length. Hence, eight combinations (S1-LS, S1-SD, S1-SS/D, S1-LS/D, S2-LS, S2-SD, S2-SS/D, and S2-LS/D) are considered for the serial type heuristics.

3.2 Parallel Type Heuristics

Unlike serial type heuristics, parallel type heuristics construct multiple paths at the same time by adding a candidate node to each of the m paths one by one while checking the time (distance) limit. The overall procedure for the parallel type heuristic is given below.

Procedure 2: (Parallel type heuristic)

Step 1: Initialize the set K of paths, i.e., $K = \{1, 2, \cdots, m\}$, each of which has only the starting and finishing nodes.

Step 2: Construct the multiple paths as follows.

(a) Set $k = 1$ (k denotes the index for paths).
(b) For path k, select a candidate node among those unconsidered so far using a priority rule.
(c) If the node can be added to path k without violating the time (distance) limit, add it to path k and go to Step 2(d). Otherwise, $K = K \setminus \{k\}$.
(d) If $K = \varnothing$, stop the algorithm. Otherwise, go to Step 2(e).
(e) Select another path k' ($k' \neq k$) with the minimum length (ties broken arbitrary) and go to Step 2(b) after setting $k = k'$.

As in the serial type heuristics, we developed eight parallel type heuristics: P1-LS, P1-SD, P1-SS/D, and P1-LS/D for the nearest neighbor method; and P2-LS, P2-SD, P2-SS/D, and P2-LS/D for the nearest insertion method.

4. Computational Experiments

To test the performances of the proposed priority rule based heuristics, computational experiments were done on a number of test instances and the results are reported in this section. All the heuristics were coded in C, and the experiments were done on a personal computer with an Intel Core2 Duo processor operating at 3.0 GHz clock speed.

Two classes of instances were tested: (a) small-to-medium sized benchmark instances; and (b) randomly generated large sized instances. The benchmark instances were obtained from Tsiligrides [17] and Chao *et al*. [6], which are six data sets with 2, 3 and 4 vehicles, and 21, 32, 33, 64, 66, and 100 nodes (The instances with 102 nodes were excluded since they have the same starting and finishing nodes). The large sized instances consist of 288 larger instances, i.e., 18 instances for each of 16 combinations of four levels of the number of nodes (300, 500, 700, and 1000) and four levels of the number of vehicles (3, 9, 27, and 81), which were generated for military applications in Korea. The locations were randomly generated on a square-shaped area. Also, the scores were generated from $DU(0, 20)$, where $DU(l, u)$ denotes the discrete uniform distribution with range $[l, u]$. The time (distance) limit T_{max} was generated by $u \cdot D_{max}$, where u is a parameter (3, 6, or 9) and D_{max} is the maximum among the distances be-

tween two nodes.

The performance measures used are: (a) percentage gaps from the best existing solution values for the benchmark instances; (b) percentage gaps from the solution values obtained from the five-step heuristic of Chao *et al*. [6] for the larger instances; and (c) CPU seconds. More formally, the gap of heuristic h for a problem is defined as

$$(Z_B - Z_h)/Z_B,$$

where Z_h is the objective value obtained from heuristic h and Z_B is the best existing solution value (of a benchmark instance) or the solution value of the five-step heuristic (of a larger instance). Here, the best existing solutions of benchmark instances were obtained from the relevant literatures [18]. Since the optimal solutions could not be obtained for the larger instances, we instead employed the five-step heuristic to compare the proposed methods. Note that the five-step heuristic is a simple one, but may give better solutions due to its sophisticated improvement step.

Table 1 shows the percentage gaps from the best existing solution values for small-to-medium sized benchmark instances. No significant differences were found between the result of nearest neighbor and nearest insertion and the LS/D, which selects a candidate node with the largest ratio of the score to the distance, performed better than the other priority rules. Note that the S1-LS/D gave the best results in overall among the sixteen heuristics. In fact, the overall average gap of S1-LS/D was 16.6% from the best existing solution values. Finally, the overall average gap of the BEST, i.e., the best solution among the sixteen heuristics, was 8.5%. Since it takes only a few seconds to implement all the sixteen heuristics, the BEST would be a good alternative of the optimal or time-consuming heuristic algorithms for practical applications. Although the gaps seem to be large, the priority rule based heuristics have a certain merit in that it can be used in real-time. For example, in military operations, they may be more useful than other time-consuming algorithms in wartime where a commander is forced to make immediate decisions.

Results for large sized test instances (with 300, 500, 700 and 1000 nodes) are summarized in Table 2(a), which shows the percentage gaps of the sixteen priority rule based heuristics with respect to the solution values obtained from the five-step heuristic. The results are similar to those for the benchmark instances except that S2-SD gave slightly better solutions than the others. As shown in Table 2, the serial type

Table 1. Test Results for Benchmark Instances

NP[1]	NV[2]	NI[3]	Serial-type								Parallel-type								BEST[†]
			Nearest Neighbor				Nearest Insertion				Nearest Neighbor				Nearest Insertion				
			S1-LS	S1-SD	S1-SS/D	S1-LS/D	S2-LS	S2-SD	S2-SS/D	S2-LS/D	P1-LS	P1-SD	P1-SS/D	P1-LS/D	P2-LS	P2-SD	P2-SS/D	P2-LS/D	
21	2	11	26.4[*]	16.7	47.2	15.4	23.9	18.3	39.8	**10.0**	24.4	15.9	44.9	21.6	59.9	30.7	60.0	25.4	8.2
	3	11	8.9	14.2	28.1	14.2	13.4	14.2	30.3	**10.0**	8.5	14.2	28.0	19.7	33.3	23.7	45.9	19.8	8.8
	4	11	16.7	**11.1**	13.9	12.7	16.7	**11.1**	24.8	**11.1**	16.7	33.3	25.0	18.2	41.7	15.8	32.5	15.6	8.3
32	2	18	37.0	23.2	66.5	17.5	29.1	**14.4**	54.7	14.9	47.1	19.9	66.3	19.2	66.3	28.6	68.8	30.8	8.9
	3	18	22.8	20.3	59.5	**13.9**	20.0	14.5	42.3	17.5	28.4	19.1	60.3	14.9	55.2	40.8	68.5	35.9	7.2
	4	18	16.9	21.3	44.8	11.3	17.7	11.7	33.9	12.2	18.6	16.8	56.7	**10.3**	51.7	42.7	67.7	34.5	4.4
33	2	20	32.6	17.3	72.8	**8.8**	23.6	19.3	60.9	14.0	30.8	16.4	70.9	15.9	52.2	22.4	69.1	29.6	6.7
	3	20	23.7	19.2	66.7	**11.5**	21.1	16.0	49.2	12.0	22.1	17.0	61.4	14.1	50.3	40.5	68.9	23.9	6.8
	4	20	22.1	15.8	61.6	**8.0**	16.1	13.5	45.3	10.1	20.2	19.6	53.0	18.6	40.9	43.8	65.8	22.7	5.4
64	2	14	38.1	48.2	85.5	29.6	39.6	57.5	80.6	32.3	**29.0**	51.3	88.6	28.9	73.7	26.9	85.4	36.0	18.0
	3	14	31.3	50.8	84.5	30.3	36.0	62.8	79.9	33.1	36.4	55.3	84.8	33.5	79.7	32.8	89.2	38.6	13.5
	4	14	33.3	44.9	82.9	31.7	38.5	59.2	76.1	33.5	37.2	52.7	85.9	40.2	75.5	37.7	90.8	41.8	10.2
66	2	26	26.1	33.3	72.9	**9.1**	33.3	16.2	65.2	22.1	34.7	35.5	74.4	10.4	55.3	51.6	79.1	61.8	7.3
	3	26	18.8	32.1	67.6	**8.2**	29.5	14.8	62.0	20.2	29.0	26.9	66.7	8.6	61.9	69.1	81.1	64.4	5.6
	4	26	20.6	31.3	65.0	**7.4**	28.8	19.0	58.0	17.9	32.9	22.7	68.5	14.1	65.4	72.6	82.7	67.3	5.0
100	2	20	52.2	34.1	91.2	**22.3**	40.1	12.8	78.4	25.6	49.2	39.6	89.6	42.5	75.8	27.5	96.9	38.1	10.0
	3	20	45.4	29.7	87.1	**27.0**	37.1	13.6	76.1	21.4	46.2	47.1	84.6	35.0	77.0	46.8	97.3	40.1	11.3
	4	20	38.9	33.0	82.2	**19.4**	33.2	9.3	66.2	16.2	36.4	52.3	81.4	33.2	73.5	51.3	95.7	38.4	7.5
Average			28.4	27.6	65.6	16.6	27.6	22.1	56.9	18.6	30.4	30.9	66.2	22.2	60.5	39.2	74.7	36.9	8.5

Note) [1] Number of nodes.

[2] Number of vehicles.

[3] Number of instances.

[*] Average percentage gap from the optimal solution values of the test instances.

[†] The best solution among the 16 heuristics.

Table 2. Test Result for Large Sized Instances

(a) Average Percentage Gaps from the Solution Values of the Five-step Heuristic

NP[1]	NV[2]	NI[3]	Serial-type								Parallel-type								BEST[†]
			Nearest Neighbor				Nearest Insertion				Nearest Neighbor				Nearest Insertion				
			S1-LS	S1-SD	S1-SS/D	S1-LS/D	S2-LS	S2-SD	S2-SS/D	S2-LS/D	P1-LS	P1-SD	P1-SS/D	P1-LS/D	P2-LS	P2-SD	P2-SS/D	P2-LS/D	
300	3	18	67.2*	23.1	95.1	35.8	52.7	**15.3**	89.5	20.4	67.2	32.7	95.6	41.1	84.0	44.4	99.6	50.6	12.5
	9	18	64.0	12.6	93.9	30.9	49.3	**9.9**	86.4	20.8	64.7	26.6	93.3	36.6	79.7	32.3	98.6	41.0	9.5
	27	18	57.7	13.5	90.5	28.1	39.7	**13.1**	81.6	19.0	58.9	41.0	89.8	40.7	78.2	42.8	98.2	44.4	11.3
	81	18	50.5	4.8	88.8	19.6	32.1	**5.7**	76.1	11.9	51.6	28.6	88.3	30.5	71.8	42.0	97.3	40.5	3.9
500	3	18	74.8	14.6	97.1	34.8	59.4	**7.9**	91.0	21.2	74.4	21.7	96.8	40.5	85.0	35.7	99.5	48.6	6.9
	9	18	72.5	15.1	96.3	38.5	58.4	**12.1**	90.8	27.3	72.4	32.6	96.0	46.6	84.3	31.0	99.4	47.2	10.2
	27	18	68.1	10.7	94.6	33.9	50.3	**9.5**	87.9	23.5	69.1	33.3	93.9	43.0	84.4	39.6	99.1	47.6	8.5
	81	18	65.2	**8.1**	93.1	28.5	44.7	10.6	84.6	19.7	65.0	28.8	92.7	40.3	81.3	37.9	98.6	41.8	8.4
700	3	18	77.9	22.7	97.0	42.9	63.0	**13.4**	93.8	30.6	78.4	32.0	97.5	47.5	89.2	35.2	99.8	49.5	11.6
	9	18	74.2	16.1	96.7	38.4	60.0	**12.5**	91.6	25.5	74.6	33.7	96.1	45.9	88.2	37.9	99.7	53.2	10.8
	27	18	71.8	18.6	96.2	37.1	55.8	**14.1**	89.6	25.1	71.9	40.0	95.6	48.6	88.4	51.6	99.7	59.2	12.5
	81	18	68.5	8.4	95.2	33.3	54.2	**7.5**	88.0	22.5	70.6	34.7	94.2	45.4	83.7	36.5	99.3	49.0	10.2
1000	3	18	79.7	17.8	98.0	41.6	64.5	**9.2**	94.5	22.4	78.5	24.2	98.0	46.2	89.7	26.1	99.9	42.1	7.7
	9	18	78.0	18.0	97.3	39.0	63.0	**10.0**	93.5	26.8	77.5	27.0	97.1	47.4	90.4	35.2	99.9	52.1	9.3
	27	18	74.9	10.5	96.7	38.0	60.3	**8.2**	92.5	26.6	74.9	32.8	96.5	47.2	90.6	46.7	99.8	55.6	7.4
	81	18	74.4	**5.7**	96.6	36.5	58.6	5.9	92.0	25.5	74.3	28.9	96.3	45.3	87.5	35.7	99.6	48.9	7.8
Average			70.0	13.8	95.2	34.8	54.1	10.3	89.0	23.1	70.3	31.2	94.8	43.3	84.8	38.2	99.2	48.2	9.3

Note) See the footnotes of Table 1.

* Average percentage gap from the solution values of the five-step heuristic.

Table 2. (Continued)
(b) CPU Seconds

NP[1]	NV[2]	NI[3]	Serial-type								Parallel-type								BEST	Five-step heuristic
			Nearest Neighbor				Nearest Insertion				Nearest Neighbor				Nearest Insertion					
			S1-LS	S1-SD	S1-SS/D	S1-LS/D	S1-LS	S1-SD	S1-SS/D	S1-LS/D	P1-LS	P1-SD	P1-SS/D	P1-LS/D	P2-LS	P2-SD	P2-SS/D	P2-LS/D		
300	3	18	< 0.1*	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.1**	39.6
	9	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.1	48.2
	27	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.1	33.5
	81	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.1	17.9
500	3	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.3	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.1	< 0.1	< 0.1	0.3	122.3
	9	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.7	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.4	< 0.1	< 0.1	1.1	186.4
	27	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.7	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.1	< 0.1	< 0.1	0.8	161.9
	81	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.8	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.8	114.8
700	3	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	1.2	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.6	< 0.1	< 0.1	1.8	268.7
	9	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	1.5	< 0.1	0.2	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.4	< 0.1	< 0.1	2.1	351.4
	27	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	1.7	< 0.1	0.2	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	0.6	< 0.1	< 0.1	2.5	337.1
	81	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	2.4	< 0.1	0.6	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	1.2	< 0.1	0.2	4.3	321.9
1000	3	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	3.1	< 0.1	0.4	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	2.0	< 0.1	< 0.1	5.4	450.1
	9	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	4.7	< 0.1	1.0	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	2.3	< 0.1	0.1	8.1	705.3
	27	18	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	5.2	< 0.1	1.2	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	1.8	< 0.1	0.3	8.5	746.3
	81	18	< 0.1	< 0.1	< 0.1	< 0.1	0.1	7.2	< 0.1	1.9	< 0.1	< 0.1	< 0.1	< 0.1	< 0.1	3.3	< 0.1	0.6	13.1	842.4

Note) See the footnotes of Table 1.
* Average of the CPU seconds out of the test instances.
** Total CPU second of the 16 priority rule based heuristics.

heuristic (S2-SD) was better than the parallel type one (P2-SD) in terms of overall performance. In fact, the overall average gaps of S2-SD and the BEST were 10.3% and 9.3%, respectively. To check the effectiveness of the proposed priority rule based heurisrtfics, an additional experiment was done on the largest test instances, i.e., both serial and the parallel type heuristics select candidate nodes randomly for 30 minutes. The results showed that the serial and the parallel type heuristics algorithms led to 74.0% and 80.4% of average gaps from the solutions obtained from the five-step heuristic.

Finally, computation times in terms of the CPU seconds were summarized in Table 2(b). As one can expect, the sixteen priority rule based heuristics were very fast. In fact, they gave solutions within 1 second for all the test instances. In particular, although it is not reported here, the BEST heuristic required 15 seconds in overall average even for the largest instances with 1000 nodes, which shows the practicality of the proposed heuristics. On the other hand, the existing five-step heuristic required about 14 minutes for the largest instances.

5. Concluding Remarks

In this study, we dealt with the team orienteering problem (TOP) that has many practical applications such as home fuel delivery, football players recruiting, service technicians scheduling, military operations, etc. The problem is to determine multiple paths from a starting node to a finishing node for a given allowed time or distance fixed for each path for the objective of maximizing the total collected score. For practically large sized instances where the existing exact or heuristic algorithms may not be useful, we suggested two types of priority rule based heuristics, serial type and parallel type heuristics, coupled with four priority rules (LS, SD, SS/D, and LS/D) and two different strategies of adding a candidate node (nearest neighbor and nearest insertion). Recall that the serial type heuristics construct multiple paths consecutively, one after another, while the parallel type heuristics construct the paths at the same time.

Computational experiments were done on the small-to-medium sized benchmark instances and randomly generated large sized instances, and the results showed

that the heuristics with the nearest insertion method gave better solutions than the others for the case of large size instances. Also, we observed that the SD rule, which selects a candidate node with the shortest distance, outperformed the other priority rules. Although producing lower performances than the existing algorithm, the proposed heuristics would be promising alternatives for practically large size instances due to their adequate performances and very short computation times.

This research may be extended in several research directions. Case studies should be done to prove the practical effectiveness of the proposed heuristics. The priority rule based heuristics may be used to give initial solutions for meta-heuristics, such as simulated annealing, tabu search and genetic algorithm, for the case where solution quality is more critical than computation time. Then, evaluation of potential benefits obtained from such combinations might be one possible direction for further study.

References

[1] Archetti, C., A. Hertz, and M. Speranza, "Metaheuristics for the team orienteering problem," *Journal of Heuristics* 13 (2007), 49-76.

[2] Bouly, H., D.-C. Dang, and A. Moukrim, "A memetic algorithm for the team orienteering problem," *Lecture Notes in Computer Science* 4974 (2008), 649-658.

[3] Boussier, S., D. Feillet, and M. Gendreau, "An exact algorithm for team orienteering problems," *4OR* 5 (2007), 211-230.

[4] Butt, S. and T. Cavalier, "A heuristic for the multiple tour maximum collection problem," *Computers and Operations Research* 21 (1994), 101-111.

[5] Chao, I., B. Golden, and E. Wasil, "A fast and effective heuristic for the orienteering problem," *European Journal of Operational Research* 88 (1996a), 475-489.

[6] Chao, I., B. Golden, and E. Wasil, "The team orienteering problem," *European Journal of Operational Research* 88 (1996b), 464-474.

[7] Fischetti, M., J. Salazar, and P. Toth, "Solving the orienteering problem through branch-and-cut," *INFORMS Journal on Computing* 10 (1998), 133-148.

[8] Gendreau, M., G. Laporte, and F. Semet, "A branch-and-cut algorithm for the undirected selective travelling salesman problem," *Networks* 32 (1998), 263-273.

[9] Golden, B., L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics* 34 (1987), 307-318.

[10] Hayes, M. and J. M. Norman, "Dynamic programming in orienteering: route choice and the siting of controls," *Journal of the Operational Research Society* 35 (1984), 791-796.

[11] Ke, L., C. Archetti, and Z. Feng, "Ants can solve the team orienteering problem," *Computers and Industrial Engineering* 54 (2008), 648-665.

[12] Laporte, G. and S. Martello, "The selective travelling salesman problem," *Discrete Applied Mathematics* 26 (1990), 193-207.

[13] Ramesh, R. and K. M. Brown, "An efficient four-phase heuristic for the generalized orienteering problem," *Computers and Operations Research* 18 (1991), 151-165.

[14] Ramesh, R., Y. Yoon, and M. Karwan, "An optimal algorithm for the orienteering tour problem," *ORSA Journal of Computing* 4 (1992), 155-165.

[15] Souffriau, W., P. Vansteenwegen, G. V. Berghe, and D. V. Oudheusden, "A path relinking approach for the team orienteering problem," *Computers and Operations Research* 37 (2010), 1853-1859.

[16] Tang, H. and E. Miller-Hooks, "A tabu search heuristic for the team orienteering problem," *Computer and Operations Research* 32 (2005) 1379-1407.

[17] Tsiligirides, T., "Heuristic methods applied to orienteering," *Journal of the Operational Research Society* 35 (1984), 797-809.

[18] Vansteenwegen, P., W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden, "A guided local search metaheuristic for the team orienteering problem," *European Journal of Operational Research* 196 (2009), 118-127.