Expert Systems with Applications 40 (2013) 3065–3072

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An augmented large neighborhood search method for solving the team orienteering problem

Byung-In Kim [a,*], Hong Li [a], Andrew L. Johnson [b]

[a] Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Kyungbuk 790-784, Republic of Korea
[b] Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77382, USA

ARTICLE INFO

Keywords:
Team orienteering problem
Large neighborhood search
Heuristics
Vehicle routing problem

ABSTRACT

In the Team Orienteering Problem (TOP), a team of vehicles attempts to collect rewards at a given number of stops within a specified time frame. Once a vehicle visits a stop and collects its reward, no other vehicles can collect the reward again. Typically, a team cannot visit all stops and therefore has to identify the "best" set of stops to visit in order to maximize total rewards. We propose a large neighborhood search method with three improvement algorithms: a local search improvement, a shift and insertion improvement, and replacement improvement. Our proposed approach can find the best known solutions for 386 of the 387 benchmark instances, for the one instance which our solution is not the current best it is only varies by one from the best. Our approach outperforms all the previous approaches in terms of solution quality and computation time.

1. Introduction

The team orienteering problem (TOP) consists of a given number of stops, each with a given reward, and a team of V_{max} vehicles attempts to maximize the total reward collected within a specified timeframe T_{max}. The travel time between stops is known and each vehicle starts and finishes at specified points. Once a vehicle visits a stop and collects the reward, no other vehicles visiting the same stop can collect the reward. In general, not all stops can be visited by the team of vehicles within the time limit. Thus, the team needs to select the "best" stops that can be visited by the set of vehicles. Fig. 1 shows an example TOP problem and a solution. Note that some stops are not visited in the figure. Most vehicle routing problems mandate visits to all stops (Toth & Vigo, 2002), but the TOP does not have this requirement.

The TOP was named by Chao, Golden, and Wasil (1996a), but various applications of TOP appear in the early literature including the home fuel delivery problem of Golden, Levy, and Vohra (1987) and the high school athlete recruitment problem of Butt and Cavalier (1994). Other applications include the sport of orienteering (Chao, Golden, & Wasil, 1996b), the routing problem of technicians to service customers (Tang & Miller-Hooks, 2005), and the personalized mobile tourist guide problem (Vansteenwegen, Souffriau, Berghe, & Van Oudheusdem, 2009).

A special case of TOP, in which there is only one team, becomes the orienteering problem (OP). The OP appears in the literature as the Traveling Salesman Problem (TSP) with profits (Feillet et al., 2005), Selective TSP (Laporte & Martello, 1990), and the maximum collection problem (Butt & Cavalier, 1994). Golden et al. (1987) prove that the OP is NP-hard. TOP is also NP-hard (Chao et al., 1996a), thus research efforts tend to focus on heuristics and metaheuristics; our paper focuses on the latter approach.

We adapt the large neighborhood search (LNS) approach of Shaw (1997) and Ropke and Pisinger (2006) to solve the TOP. In the LNS approach, a current solution is perturbed significantly and large neighborhoods are searched in a single iteration. Within the LNS framework, we apply three improvement algorithms: the local search improvement algorithm, shift and insertion algorithm, and replacement algorithm. Our proposed approach can identify the best known solutions for 386 of 387 benchmark problems. For the remaining problem instance, our approach finds a solution whose reward is 1 less the current best solution. The solution quality equals that of a recently proposed approach by Dang, Guibadj, and Moukrim (2012), which outperforms all the previous approaches, but the computation time of our approach is less.

The remainder of this paper is organized as follows. A brief literature review is presented in Section 2. Our proposed approach and embedded improvement algorithms are explained in Section 3. Experimental results and concluding remarks follow in Sections 4 and 5, respectively.

* Corresponding author. Tel.: +82 54 279 2371; fax: +82 54 279 2870.
E-mail address: bkim@postech.ac.kr (B.-I. Kim).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.12.022

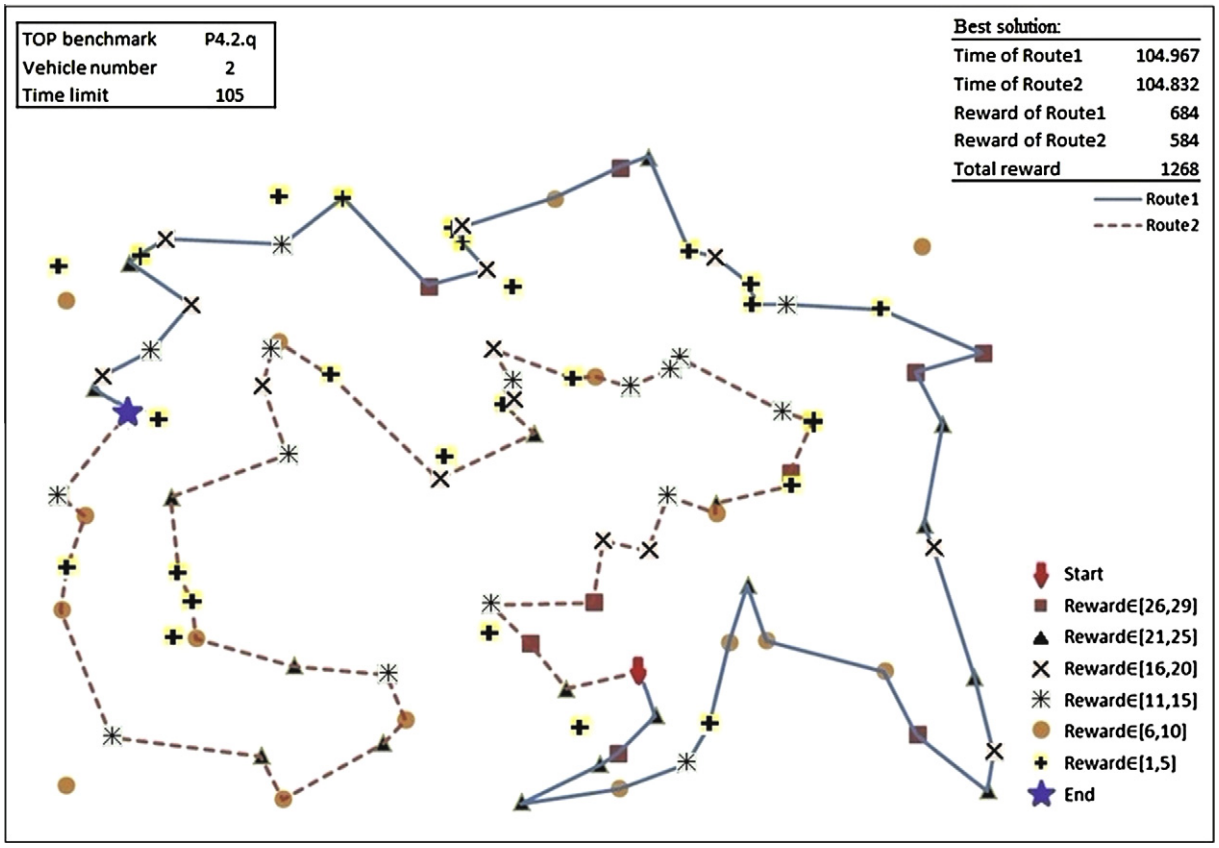

Fig. 1. A TOP problem (instance p4.2.q and a solution with total reward of 1268).

2. Literature review

This section provides a brief review of the TOP literature focusing on heuristic approaches, and the LNS algorithm. Butt and Cavalier (1994) address a high school athlete recruiting problem and call it the Multiple Tour Maximum Collection Problem. This problem was later renamed the TOP. They present a mixed integer programming (MIP) model and a simple heuristic algorithm, in which a node is assigned one at a time to a route based on the ratio of its reward value over the distance increment needed. Butt and Ryan (1999) propose a set partitioning based MIP model. They solve the problem using a column generation method and claim that their solution approach works well for problems in which the number of nodes visited in a tour is relatively small.

Chao et al. (1996a) develop a heuristic approach for the TOP and benchmark data sets. In their approach, the stops farthest from the start and finish points are selected as seeds for the team members and all possible remaining points are inserted into the routes using the cheapest insertion rule. If unassigned points remain, new team routes are constructed. The initial solution is then improved using 1-point movement, 2-point exchange, and 2-opt operator with the record-to-record framework, which is a deterministic version of the simulated annealing algorithm. Removal and reinsertion of points are also applied for improvement.

Tang and Miller-Hooks (2005) present a MIP model and a tabu search with an adaptive memory procedure that alternates between small and large neighborhood exploration. For the neighborhood search, they use random and greedy procedures. Their work generates better solutions abet using more computation time than Chao et al. (1996a) for many problem instances. Archetti, Hertz, and Speranza (2007) propose a tabu search algorithm allowing feasible move, a tabu search algorithm allowing infeasible move with penalty, and a variable neighborhood search algorithm allowing feasible move for the TOP. The last approach outperforms Chao et al. (1996a) and Tang and Miller-Hooks (2005).

Ke, Archetti, and Feng (2008), who present an MIP model and an ant colony optimization (ACO) approach, propose the sequential, deterministic-concurrent, random-concurrent, and simultaneous methods as the construction algorithms. The performance of the ACO approach with the sequential method is comparable to Archetti et al. (2007). Ke et al. identify the best-so-far solutions for 347 instances and new best solutions for 12 instances out of 387 benchmark instances.

Vansteenwegen et al. (2009) present an MIP model and propose an algorithm that combines a guided local search method and a diversifying mechanism called disturb. Although the computation time of their approach is shorter, the solution quality is no better than the existing approaches.

Bouly, Dang, and Moukrim (2010) and Dang, Guibadj, and Moukrim (2011) present a particle swarm optimization-based memetic algorithm (PSOMA) in which tour-splitting techniques and genetic crossover improvement are developed. They attained a gap of 31 rewards for the total benchmark problems. In other words, the sum of benchmark solutions rewards that collected by PSOMA is 31 less than the sum of the known current best solutions that attained by various approaches in the literature. Dang et al. (2012) extend their previous research and propose a PSO-inspired algorithm (PSOiA). They updated a best known solution for one instance and found the best known solutions for all the benchmark

instance but one. Our proposed approach achieves the same set of results as PSOiA, which is currently the state-of-the-art algorithm, but requires less computation time.

Local search heuristics attempt to improve solutions via incremental adjustments to current solutions. In the LNS algorithm (Shaw, 1997), the current solution is perturbed significantly and large neighborhoods are explored in a single iteration. Although the computation time is generally longer than local search heuristics, it performs well for various vehicle routing problems (Jun & Kim, 2012; Pisinger & Ropke, 2007; Ropke & Pisinger, 2006). Schrimpf, Schneider, Stamm-Wilbrandt, and Dueck (2000) propose a similar approach called the ruin and recreate heuristic. Pisinger and Ropke (2007) present a detailed history of the LNS and its similarity to the very large scale neighborhood search (VLSN) of Ahuja, Ergun, Orlin, and Punnen (2002) and variable neighborhood search (VNS) of Hansen and Mladenovic (1997).

3. Proposed approach

This paper proposes an augmented LNS (ALNS) with various improvement algorithms. The pseudo-code of the proposed ALNS is shown in Algorithm 1. The number of iterations (I_{max}) and the maximum number of solutions (N) in the solution pool S are user-given parameters. In line 2 an initial solution is generated by a construction method, which will be described later, and in line 3 the solution is improved by local search methods. In line 5 the improved solution is added to the solution pool. In line 8 a solution is selected randomly from the solution pool and in line 9 a random number (k) between 1 and 3 quarters of the number of *routed* stops in the selected solution is generated. The number of routed stops defines the scope of the search.

In line 10, k stops are removed from the selected solution according to the remove criterion. Three remove criterions are designed for solution diversification: random remove, biggest remove and smallest remove. In the random remove, k stops will be randomly selected and removed from the current solution. The biggest remove will remove the k stops which have the biggest rewards and the k smallest rewards stops will be removed under the smallest remove criterion similarly. The three remove criterions are randomly selected in each iteration.

In lines 12 through 15 the solution with stops removed is repaired via two improvement methods until there are no more improvements (the improvement algorithms are explained below). In line 16 some of the *unrouted* stops are inserted via an insertion method. In the insertion method, each *unrouted* stop is tested to be inserted into a route and the *unrouted* feasible stop that has the smallest distance increase due to the stop insertion is iteratively selected and inserted. The insertion procedure is repeated until no additional stops can be inserted to any of the routes.

In lines 17 through 22 the updated solution is further improved not only by the previous two improvement methods, but also with two replacement methods explained below. In lines 23 and 24 the algorithm terminates if the improved solution is the upper bound, where the upper bound terminate condition is applied for those instances in which the exact optimal solutions are reported by Boussier, Feillet, and Gendreau (2007). For those instances in which the exact optimal solutions are not known, lines 23 and 24 are skipped. Note that Dang et al. (2011) also used the same upper bound terminate condition. In lines 25 and 26 the best solution is updated if the improved solution is better than the current best solution (S_{best}). In lines 27 and 28 the new solution replaces the worst solution if: (1) the solution pool is full, (2) the current solution is not present in the pool and (3) the pool's worst solution is worse than the current solution. In lines 29 and 30 the new solution is inserted into the solution pool if the solution pool is not yet full and does not contain the same solution. Lines 7 through 30 are repeated until the number of iterations is reached at which point the best solution is returned.

Algorithm 1. ALNS basic framework

1 Input: I_{max} = number of iterations, N = number of maximum solutions in the solution pool
2 Generate an initial solution s_0 by the construction heuristic of Algorithm 2
3 Improve s_0 by local search methods
4 $S_{best} = s_0$
5 Insert s_0 into solution pool S
6 Repeat until the number of iterations reaches I_{max}
7 Set all the stops *unrouted*
8 s_1 = random selection of a solution from S
9 k = random number between 1 and 0.75∗number of *routed* stops in s_1
10 Remove k stops from s_1 and set them *unrouted*
11 Set remained stops in s_1 *routed*
12 Repeat while there is improvement
13 Improve s_1 by the local search method of Algorithm 3
14 Improve s_1 by the shifting method of Algorithm 4
15 End repeat
16 Reinsert as many as possible *unrouted* stops to s_1 by the insertion method
17 Repeat while there is improvement
18 Improve s_1 by the local search method of Algorithm 3
19 Improve s_1 by the replacement method (random) of Algorithm 5-1
20 Improve s_1 by the shifting method of Algorithm 4
21 Improve s_1 by the replacement method (all) of Algorithm 5-2
22 End repeat
23 If ($f(s_1)$ = *Upper Bound*) then
24 Terminate algorithm and return s_1 (line 32)
25 If ($f(s_1) > f(S_{best})$) then
26 $S_{best} = s_1$
27 If ($s_1 \notin S$ and $n(S) = N$ and $f(s_1) > f(S_{worst})$) then
28 Replace S_{worst} of S with s_1
29 If ($s_1 \notin S$ and $n(S) < N$) then
30 Insert s_1 into solution pool S
31 End repeat
32 Return S_{best}

Our construction algorithm is a greedy heuristic, the pseudo-code for which appears in Algorithm 2. From the current stop, c_L, the feasible stop that has the smallest distance divided by reward is selected as the next stop. The procedure repeats until no more stops can be added to the route. In line 11 the route is improved by the intra-route improvement algorithm (explained in Algorithm 3). When there is improvement, i.e., the total route travel distance is reduced, additional stops are inserted. In line 12 the algorithm attempts to add as many *unrouted* stops as possible to the route; which is similar to line 16 of Algorithm 1, but with a route.

Algorithm 2. Construction algorithm

1 Input: V_{max} = number of teams, T_{max} = maximum allowable time
2 Set all the stops *unrouted*
3 For each team in V_{max},
4 c_L = the team start stop
5 Repeat while there is feasible c_N stop

(*continued on next page*)

```
6       c_N = the unrouted stop that can be serviced within
        T_max and have the least value of distance from c_L/the
        reward of the stop
7       If there is feasible c_N,
8         Insert c_N to the route of the team and set c_N routed
9         c_L = c_N
10    End repeat
11    Improve the route with the intra-route improvement of
      Algorithm 3
12    Insert as many as possible unrouted stops into the route
      by the insertion method
13    Insert the generated team route into solution s_0
14  End For
15  Return s_0
```

The pseudo-code for our local search improvement algorithm appears in Algorithm 3. It consists of popular inter-route and intra-route improvement methods. In the 1-1 improvement method, a stop from a route is exchanged with a stop from another route. The exchange is accepted if it is feasible for both routes and the total travel distance of the two routes is reduced. In the 1-0 improvement method, a stop from a route is moved into another route and improvement is tested. Similarly, the 2-1 improvement method exchanges two stops from a route with one stop from another route. The intra-route improvement algorithm consists of well-known 2-opt edge exchange method and Or-opt improvement method (Or, 1976). The improvement methods repeat until no further improvement is possible.

Algorithm 3. Local search improvement algorithm

```
1   Input: s_1 = current solution
2   Repeat while there is improvement in any inter-route or
    intra-route
3     Repeat while there is any improvement (inter-route
      algorithm)
4       For all the combinations of the routes in s_1
5         Do 1-1 improvement
6         Do 1-0 improvement
7         Do 2-1 improvement
8       End For
9     End repeat
10    Repeat while there is any improvement (intra-route
      algorithm)
11      For each route in the solution s_1
12        Do 2-opt edge exchange improvement
13        Do Or-opt improvement
14      End For
15    End repeat
16  End repeat
17  Return improved s_1
```

The pseudo-code for our shifting and insertion algorithm appears in Algorithm 4. In lines 3 and 4 attempts are made to move stops from a route into other routes to make room for *unrouted* stops. In lines 6 and 7 all possible *unrouted* stops are inserted into a route; observe that line 7 is the same as line 12 of Algorithm 2.

Regarding Algorithm 4, we note that although stops can be exchanged among the routes using Algorithm 3, adding Algorithm 4 improves the solution far more rapidly. Moreover, Algorithm 3 does not allow the movement of stops that results in a worse solution, however, Algorithm 4 does not have such a restriction thus additional *unrouted* stops can be inserted.

Algorithm 4. Shifting and insertion algorithm

```
1   Input: s_1 = current solution
2   For each route r in s_1
3     For each stop in route r
4       If feasible, move the stop into the least distance
        incremental route
5     End repeat
6     If any stops are moved from route r to other routes
7       Insert as many as possible unrouted stops into route r
        by the insertion method
8   End For
9   Return improved s_1
```

The pseudo-code for our random replacement algorithm appears in Algorithm 5-1. The algorithm exchanges *routed* stops with *unrouted* stops to improve the solution's total reward value. In line 5 a random number is generated for each iteration and this number determines the number of stops that will be removed from the route. In line 6 the stops to be deleted are selected randomly and their total rewards are compared with the total rewards of all *unrouted* stops. In lines 7 and 8 if the value of the *unrouted* stops is less than the value of the deleted stops, that iteration will be skipped because there is no potential benefit to the exchange. Otherwise in line 11 the selected stops are deleted from the route and *unrouted* stops are randomly inserted. In this step, each *unrouted* stop is tested to determine if the insertion is feasible. In lines 12 through 14 if the total rewards of the newly inserted stops are larger than the total rewards of the deleted stops, the replacement for the route will be accepted and the next route will be tested. The above procedures repeat for R_{max} iterations for each route.

Algorithm 5-1. Replacement algorithm (random)

```
1   Input: s_1 = current solution
       D_max = maximum stop numbers that can be deleted
       from a route
       R_max = number of max iterations
2   For each route r in s_1
3     n_r = number of stops in route r
4     Repeat R_max iterations
5       d_r = random number between 1 and min(D_max, n_r)
6       Randomly select d_r stops from route r
7       If the total rewards of the unrouted stops are not
        larger than the total rewards of the selected stops
8         Go to next iteration (line 4)
9       Remove the selected stops from route r
10      Generate a random sequence of the unrouted stops
11      Insert as many as possible unrouted stops into route r
        according to the random sequence
12      If the total rewards of the newly inserted stops are
        larger than the total rewards of the deleted stops
13        Accept the change
14        Go to next route (line 2)
15      End repeat
16  End For
17  Return improved s_1
```

The Algorithm 5-2 is a full-enumeration replacement algorithm. Instead of randomly selecting d_r *routed* stops and randomly replacing current stops, the full-enumeration replacement algorithm tests all *routed* stops for a 1-1 replacement and then a 1-2 replacement. The 1-1 replacement exchanges a *routed* stop with an *unrouted* stop and repeats until there is no improvement. Then 1-2 replacement is tested checking the replacement of a *routed* stop with two *unrouted*

Table 1
Benchmark data set.

Problem set	Number of stops	Number of sub-problems	T_{max}
p1.2	32	18	2.5–42.5
p1.3	32	18	1.7–28.3
p1.4	32	18	1.2–21.2
p2.2	21	11	7.5–22.5
p2.3	21	11	5.0–15.0
p2.4	21	11	3.8–11.2
p3.2	33	20	7.5–55.0
p3.3	33	20	5.0–36.7
p3.4	33	20	3.8–27.5
p4.2	100	20	25.0–120.0
p4.3	100	20	16.7–80.0
p4.4	100	20	12.5–60.0
p5.2	66	26	2.5–65.0
p5.3	66	26	1.7–43.3
p5.4	66	26	1.2–32.5
p6.2	64	14	7.5–40.0
p6.3	64	14	5.0–26.7
p6.4	64	14	3.8–20.0
p7.2	102	20	10.0–200.0
p7.3	102	20	6.7–133.3
p7.4	102	20	5.0–100.0

stops. The replacement is executed when the solution rewards are increased or the route travel time is decreased without rewards loses.

Algorithm 5-2. Replacement algorithm (all)

```
1  Input: s1 = current solution
2  For each route r in s1
3    Do 1-1 replacement while there is improvement
4    Do 1-2 replacement while there is improvement
5  End For
6  Return improved s1
```

4. Experimental results

We programed our approach in C++ and conducted the experiments on an Intel Core i7-2600 CPU with 3.40 GHz, 16 GB RAM running Windows 7. The computational experiments are demonstrated on a set of 387 benchmark instances from Chao et al. (1996a, http://www-c.eco.unibs.it/~archetti/TOP.zip). Table 1 summarizes the characteristics of the seven benchmark problem sets.

Table 2
Best rewards obtained by the algorithms.

Set	TMH	GTP	GTF	FVF	SVF	SEQ	DET	RAN	SIM	GLS	PSOMA (ω = 0.07)	PSOiA	ALNS (I_{max} = 5000)
p.1.2	148.8	**149.1**	**149.1**	**149.1**	**149.1**	**149.1**	**149.1**	**149.1**	**149.1**	–	**149.1**	**149.1**	**149.1**
p.1.3	124.7	**125.0**	**125.0**	**125.0**	**125.0**	**125.0**	**125.0**	**125.0**	**125.0**	–	**125.0**	**125.0**	**125.0**
p.1.4	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**	–	**101.0**	**101.0**	**101.0**
p.2.2	190.0	**190.5**	**190.5**	**190.5**	**190.5**	**190.5**	**190.5**	**190.5**	**190.5**	–	**190.5**	**190.5**	**190.5**
p.2.3	135.9	**136.4**	**136.4**	**136.4**	**136.4**	**136.4**	**136.4**	**136.4**	**136.4**	–	**136.4**	**136.4**	**136.4**
p.2.4	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	–	**94.5**	**94.5**	**94.5**
p.3.2	492.0	494.5	**496.0**	**496.0**	**496.0**	**496.0**	**496.0**	**496.0**	**496.0**	–	**496.0**	**496.0**	**496.0**
p.3.3	408.0	**411.5**	**411.5**	**411.5**	**411.5**	**411.5**	**411.5**	**411.5**	**411.5**	–	**411.5**	**411.5**	**411.5**
p.3.4	335.0	**336.5**	**336.5**	**336.5**	**336.5**	**336.5**	**336.5**	**336.5**	**336.5**	–	**336.5**	**336.5**	**336.5**
p.4.2	895.1	904.9	908.5	914.0	915.9	915.6	908.4	909.5	911.8	887.5	916.9	**917.1**	**917.1**
p.4.3	844.3	845.5	852.5	853.0	855.6	853.8	847.7	848.4	848.2	830.0	856.1	**856.2**	**856.2**
p.4.4	784.6	800.1	802.3	801.7	801.9	798.1	795.9	791.4	795.2	770.9	803.6	**804.1**	**804.1**
p.5.2	886.8	892.6	897.4	895.8	896.8	897.6	896.4	896.2	896.2	882.4	897.6	**897.8**	**897.8**
p.5.3	775.8	781.4	**783.6**	**783.6**	**783.6**	782.8	780.4	781.2	781.0	769.8	**783.6**	**783.6**	**783.6**
p.5.4	699.0	707.5	**708.8**	**708.8**	**708.8**	**708.8**	707.7	706.3	705.6	696.3	708.5	**708.8**	**708.8**
p.6.2	818.2	813.8	818.7	**819.3**	**819.3**	**819.3**	818.7	818.7	**819.3**	804.5	**819.3**	**819.3**	**819.3**
p.6.3	783.0	**792.8**	**792.8**	**792.8**	**792.8**	**792.8**	790.5	790.5	791.3	783.0	**792.8**	**792.8**	**792.8**
p.6.4	712.8	**714.0**	**714.0**	**714.0**	**714.0**	**714.0**	**714.0**	**714.0**	**714.0**	710.4	**714.0**	**714.0**	**714.0**
p.7.2	633.5	639.6	641.4	640.6	642.6	642.7	641.5	641.0	641.2	631.2	**642.8**	**642.8**	**642.8**
p.7.3	592.5	596.7	597.7	597.1	599.2	599.9	599.4	598.6	599.2	585.6	599.8	**600.0**	**600.0**
p.7.4	514.6	517.2	516.9	516.9	518.9	**519.1**	518.2	518.4	518.4	503.0	518.9	**519.1**	**519.1**

Best results are in boldface.

Table 3
Average computation time (s).

Set	TMH	GTP	GTF	FVF	SVF	SEQ	DET	RAN	SIM	GLS	PSOMA[a]	PSOiA	ALNS[b]	ALNS[c]
p1	–	4.7	1.6	0.1	7.8	5.8	5.2	4.9	5.2	2.2	0.2	2.2	0.8	1.4
p2	–	0.0	0.0	0.0	0.0	3.2	3.0	2.8	3.0	2.3	0.0	0.4	0.0	0.0
p3	–	6.0	1.6	0.2	10.2	6.5	6.0	5.7	6.0	1.0	0.5	3.2	1.4	2.6
p4	184.5	105.3	282.9	22.5	457.9	36.8	31.8	30.7	32.0	12.7	78.5	218.6	34.7	77.3
p5	16.8	69.5	26.6	34.2	158.9	17.4	15.1	14.3	15.1	5.0	12.9	49.5	10.4	22.1
p6	14.1	66.3	20.2	8.7	147.9	16.1	14.1	13.5	14.2	5.8	4.0	47.1	6.1	12.3
p7	84.3	159.0	256.8	10.3	309.9	30.4	24.6	23.3	24.7	16.7	54.5	97.5	31.7	66.8

TMH: DEC Alpha XP1000 Computer.
GTP, GTF, FVF, SVF: Intel Pentium 4 with 2.80 GHz and 1 GB RAM.
SEQ, DET, RAN, SIM: Intel PC with 3.0 GHz.
GLS: Intel Pentium 4 with 2.80 GHz and 1 GB RAM.
PSOMA, PSOiA: AMD Opteron 2.60 GHz, Linux.
ALNS: Intel Core i7-2600 CPU with 3.40 GHz, 16 GB RAM, Windows 7.
[a] PSOMA with parameter ω = 0.07.
[b] ALNS with parameter I_{max} = 5000.
[c] ALNS with parameter I_{max} = 10,000.

The first number in the problem set name is the set number, e.g., p1.2, p1.3, and p1.4 belong to problem set 1. The coordinates and rewards of each stop are identical in all instances of the same problem set. Each problem set has three sub problem sets with differing numbers of available teams. The second number in the problem set name indicates the number of teams, e.g., p1.3 implies there are three teams available. Column 2 in the table shows the number of stops in the particular instance. Each sub problem set has a certain number of problem instances, each with a different value of T_{max}. Column 3 shows the number of problem instances and column 4 shows the overall range of T_{max} over all instances. Data sets 1 and 3 originate from Tsiligirides (1984).

We compare our solution approach, denoted in the following tables as ALNS, with the following 12 algorithms, available in the literature, on the set of 387 benchmark instances.

TMH: the tabu search algorithm of Tang and Miller-Hooks (2005)
GTH: the tabu search with penalty strategy of Archetti et al. (2007)
GTF: the tabu search with feasible strategy of Archetti et al. (2007)
FVF: the fast variable neighborhood search of Archetti et al. (2007)

Table 4
Current best solutions.

Instance	Best	Instance	Best	Instance	Best	Instance	Best	Instance	Best	Instance	Best	Instance	Best
p1.2.a	**0**	p2.2.c	**140**	p3.3.f	**230**	p4.3.b	**38**	p5.2.r	1260	p5.4.v	**1320**	p7.2.j	646
p1.2.b	**15**	p2.2.d	**160**	p3.3.g	270	p4.3.c	**193**	p5.2.s	1340	p5.4.w	1390	p7.2.k	705
p1.2.c	**20**	p2.2.e	**190**	p3.3.h	**300**	p4.3.d	335	p5.2.t	1400	p5.4.x	1450	p7.2.l	767
p1.2.d	**30**	p2.2.f	**200**	p3.3.i	330	p4.3.e	**468**	p5.2.u	1460	p5.4.y	1520	p7.2.m	827
p1.2.e	**45**	p2.2.g	**200**	p3.3.j	380	p4.3.f	579	p5.2.v	1505	p5.4.z	1620	p7.2.n	888
p1.2.f	**80**	p2.2.h	**230**	p3.3.k	440	p4.3.g	653	p5.2.w	1565	p6.2.a	**0**	p7.2.o	945
p1.2.g	**90**	p2.2.i	**230**	p3.3.l	480	p4.3.h	729	p5.2.x	1610	p6.2.b	**0**	p7.2.p	1002
p1.2.h	110	p2.2.j	**260**	p3.3.m	520	p4.3.i	809	p5.2.y	1645	p6.2.c	**0**	p7.2.q	1044
p1.2.i	**135**	p2.2.k	275	p3.3.n	570	p4.3.j	861	p5.2.z	1680	p6.2.d	**192**	p7.2.r	1094
p1.2.j	155	p2.3.a	**70**	p3.3.o	590	p4.3.k	919	p5.3.a	**0**	p6.2.e	**360**	p7.2.s	1136
p1.2.k	175	p2.3.b	**70**	p3.3.p	640	p4.3.l	979	p5.3.b	**15**	p6.2.f	**588**	p7.2.t	1179
p1.2.l	**195**	p2.3.c	**105**	p3.3.q	680	p4.3.m	1063	p5.3.c	**20**	p6.2.g	**660**	p7.3.a	**0**
p1.2.m	**215**	p2.3.d	**105**	p3.3.r	**710**	p4.3.n	1121	p5.3.d	**60**	p6.2.h	**780**	p7.3.b	**46**
p1.2.n	**235**	p2.3.e	**120**	p3.3.s	720	p4.3.o	1172	p5.3.e	**95**[a]	p6.2.i	**888**	p7.3.c	**79**
p1.2.o	**240**	p2.3.f	**120**	p3.3.t	760	p4.3.p	1222	p5.3.f	**110**	p6.2.j	948	p7.3.d	**117**
p1.2.p	250	p2.3.g	**145**	p3.4.a	**20**	p4.3.q	1253	p5.3.g	**185**	p6.2.k	1032	p7.3.e	**175**
p1.2.q	265	p2.3.h	**165**	p3.4.b	**30**	p4.3.r	1273	p5.3.h	**260**	p6.2.l	1116	p7.3.f	**247**
p1.2.r	280	p2.3.i	**200**	p3.4.c	**90**	p4.3.s	1295	p5.3.i	**335**	p6.2.m	1188	p7.3.g	**344**
p1.3.a	**0**	p2.3.j	**200**	p3.4.d	**100**	p4.3.t	1305	p5.3.j	**470**	p6.2.n	1260	p7.3.h	425
p1.3.b	**0**	p2.3.k	**200**	p3.4.e	**140**	p4.4.a	**0**	p5.3.k	**495**	p6.3.a	**0**	p7.3.i	**487**
p1.3.c	**15**	p2.4.a	**10**	p3.4.f	**190**	p4.4.b	**0**	p5.3.l	595	p6.3.b	**0**	p7.3.j	564
p1.3.d	**15**	p2.4.b	**70**	p3.4.g	**220**	p4.4.c	**0**	p5.3.m	**650**	p6.3.c	**0**	p7.3.k	633
p1.3.e	**30**	p2.4.c	**70**	p3.4.h	**240**	p4.4.d	**38**	p5.3.n	**755**	p6.3.d	**0**	p7.3.l	684
p1.3.f	**40**	p2.4.d	**70**	p3.4.i	**270**	p4.4.e	**183**	p5.3.o	**870**	p6.3.e	**0**	p7.3.m	762
p1.3.g	**50**	p2.4.e	**70**	p3.4.j	310	p4.4.f	**324**	p5.3.p	**990**	p6.3.f	**0**	p7.3.n	820
p1.3.h	70[a]	p2.4.f	**105**	p3.4.k	**350**	p4.4.g	461	p5.3.q	1070	p6.3.g	**282**	p7.3.o	874
p1.3.i	**105**	p2.4.g	**105**	p3.4.l	**380**	p4.4.h	**571**	p5.3.r	1125	p6.3.h	**444**	p7.3.p	929
p1.3.j	**115**	p2.4.h	**120**	p3.4.m	**390**	p4.4.i	657	p5.3.s	1190	p6.3.i	**642**	p7.3.q	987
p1.3.k	**135**	p2.4.i	**120**	p3.4.n	440	p4.4.j	732	p5.3.t	1260	p6.3.j	**828**	p7.3.r	1026
p1.3.l	**155**	p2.4.j	**120**	p3.4.o	**500**	p4.4.k	821	p5.3.u	1345	p6.3.k	894	p7.3.s	1081
p1.3.m	175	p2.4.k	**180**	p3.4.p	**560**	p4.4.l	880	p5.3.v	1425	p6.3.l	1002	p7.3.t	1120
p1.3.n	190	p3.2.a	**90**	p3.4.q	560	p4.4.m	919	p5.3.w	1485	p6.3.m	1080	p7.4.a	**0**
p1.3.o	205[a]	p3.2.b	**150**	p3.4.r	600	p4.4.n	977	p5.3.x	1555	p6.3.n	**1170**	p7.4.b	**30**
p1.3.p	**220**	p3.2.c	**180**	p3.4.s	**670**	p4.4.o	1061	p5.3.y	1595	p6.4.a	**0**	p7.4.c	**46**
p1.3.q	230	p3.2.d	220	p3.4.t	**670**	p4.4.p	1124	p5.3.z	1635	p6.4.b	**0**	p7.4.d	**79**
p1.3.r	250	p3.2.e	260	p4.2.a	**206**	p4.4.q	1161	p5.4.a	**0**	p6.4.c	**0**	p7.4.e	**123**
p1.4.a	**0**	p3.2.f	300	p4.2.b	**341**	p4.4.r	1216	p5.4.b	**0**	p6.4.d	**0**	p7.4.f	**164**
p1.4.b	**0**	p3.2.g	360	p4.2.c	452	p4.4.s	1260	p5.4.c	**20**	p6.4.e	**0**	p7.4.g	**217**
p1.4.c	**0**	p3.2.h	410	p4.2.d	531	p4.4.t	1285	p5.4.d	**20**	p6.4.f	**0**	p7.4.h	**285**
p1.4.d	**15**	p3.2.i	460	p4.2.e	618	p5.2.a	**0**	p5.4.e	**20**	p6.4.g	**0**	p7.4.i	**366**
p1.4.e	**15**	p3.2.j	510	p4.2.f	687	p5.2.b	**20**	p5.4.f	**80**	p6.4.h	**0**	p7.4.j	**462**
p1.4.f	**25**	p3.2.k	550	p4.2.g	757	p5.2.c	**50**	p5.4.g	**140**	p6.4.i	**0**	p7.4.k	520
p1.4.g	**35**	p3.2.l	590	p4.2.h	835	p5.2.d	**80**	p5.4.h	**140**	p6.4.j	**366**	p7.4.l	590
p1.4.h	**45**	p3.2.m	620	p4.2.i	918	p5.2.e	**180**	p5.4.i	**240**	p6.4.k	**528**[a]	p7.4.m	646
p1.4.i	**60**	p3.2.n	660	p4.2.j	965	p5.2.f	**240**	p5.4.j	**340**	p6.4.l	696	p7.4.n	730
p1.4.j	**75**	p3.2.o	690	p4.2.k	1022	p5.2.g	**320**	p5.4.k	**340**	p6.4.m	912	p7.4.o	781
p1.4.k	**100**	p3.2.p	720	p4.2.l	1074	p5.2.h	**410**	p5.4.l	**430**	p6.4.n	**1068**	p7.4.p	846
p1.4.l	**120**	p3.2.q	760	p4.2.m	1132	p5.2.i	**480**	p5.4.m	**555**	p7.2.a	**30**	p7.4.q	909
p1.4.m	130	p3.2.r	790	p4.2.n	1174	p5.2.j	**580**	p5.4.n	**620**	p7.2.b	**64**	p7.4.r	970
p1.4.n	**155**	p3.2.s	800	p4.2.o	1218	p5.2.k	**670**	p5.4.o	**690**	p7.2.c	**101**	p7.4.s	1022
p1.4.o	**165**	p3.2.t	**800**	p4.2.p	1242	p5.2.l	800	p5.4.p	765	p7.2.d	**190**	p7.4.t	1077
p1.4.p	**175**	p3.3.a	**30**	p4.2.q	1268	p5.2.m	860	p5.4.q	**860**	p7.2.e	**290**		
p1.4.q	**190**	p3.3.b	**90**	p4.2.r	1292	p5.2.n	925	p5.4.r	**960**	p7.2.f	**387**		
p1.4.r	**210**	p3.3.c	**120**	p4.2.s	1304	p5.2.o	1020	p5.4.s	1030	p7.2.g	459		
p2.2.a	**90**	p3.3.d	**170**	p4.2.t	1306	p5.2.p	1150	p5.4.t	**1160**	p7.2.h	521		
p2.2.b	**120**	p3.3.e	**200**	p4.3.a	**0**	p5.2.q	1195	p5.4.u	**1300**	p7.2.i	580		

*Known exact solutions are in boldface.
[a] Although Chao et al. (1996a) give better solutions, they round the final length of a path to one decimal place. Thus we do not count their solution for the current best solutions.

Table 5
Effects of the three improvement algorithms.

Set	With Algorithms 3,4,5		With Algorithms 4, 5		With Algorithms 3, 5		With Algorithm 3,4	
	Reward	Time (s)	Reward	Time (s)	Reward	Time (s)	Reward	Time (s)
p.1.2	149.1	1.0	149.1	1.2	149.1	0.8	149.1	0.3
p.1.3	125.0	1.3	125.0	2.2	125.0	1.1	125.0	0.3
p.1.4	101.0	0.2	101.0	0.7	101.0	0.2	101.0	0.0
p.2.2	190.5	0.1	190.5	0.1	190.5	0.1	190.5	0.0
p.2.3	136.4	0.0	135.0	0.0	136.4	0.0	136.4	0.0
p.2.4	94.5	0.0	92.7	0.0	94.5	0.0	94.5	0.0
p.3.2	496.0	2.0	496.0	3.0	496.0	1.9	496.0	0.8
p.3.3	411.5	1.7	411.5	4.0	411.5	1.7	411.5	0.6
p.3.4	336.5	0.5	336.5	1.6	336.5	0.6	336.5	0.2
p.4.2	917.1	32.2	915.5	32.1	916.3	27.0	917.1	16.7
p.4.3	856.2	38.9	855.7	38.8	856.2	31.6	856.2	14.7
p.4.4	804.1	41.2	803.5	46.1	804.1	34.5	804.1	12.5
p.5.2	897.8	11.4	897.6	9.3	897.8	9.8	897.8	4.2
p.5.3	783.6	12.2	783.6	10.6	783.6	10.3	783.6	3.5
p.5.4	708.8	9.2	708.8	8.4	708.8	8.1	708.8	1.9
p.6.2	819.3	11.3	819.3	8.0	819.3	9.5	819.3	2.8
p.6.3	792.8	10.7	792.8	9.5	792.8	8.8	792.8	1.8
p.6.4	714.0	9.7	714.0	9.0	714.0	8.5	714.0	1.4
p.7.2	642.8	31.6	642.8	30.2	642.8	27.8	642.8	17.3
p.7.3	600.0	32.2	599.6	33.5	599.7	27.6	599.9	11.1
p.7.4	519.1	34.5	519.1	34.2	519.1	29.9	519.1	7.2
Total	195,929	5176.7	195,833	5204.0	195,909	4406.5	195,927	1854.3

SVF: the slow variable neighborhood search of Archetti et al. (2007)
SEQ: the sequential algorithm of Ke et al. (2008)
DET: the deterministic-concurrent algorithm of Ke et al. (2008)
RAN: the random-concurrent algorithm of Ke et al. (2008)
SIM: the simultaneous algorithm of Ke et al. (2008)
GLS: the guided local search algorithm of Vansteenwegen et al. (2009)
PSOMA: the PSO-based memetic algorithm of Dang et al. (2011)
PSOiA: the effective PSO-inspired algorithm of Dang et al. (2012)

Since there are random choices in our proposed approach, we execute 10 runs on each instance in order to select the best value. The computation time is measured by the average of 10 computation runs. Note that Archetti et al. (2007), Dang et al. (2011, 2012) measure the computation time similarly. We set the following parameters as: I_{max} (number of iterations) = 5000, N (number of maximum solutions in the solution pool) = 20 in Algorithm 1, and D_{max} (maximum stop numbers that can be deleted from a route) = 3, R_{max} (number of max iterations) = 100 in Algorithm 5. These parameters were identified through preliminary experiments considering both solution quality and computation time. Note that our iteration numbers, 5000 and 10 replications, are comparable to Ropke and Pisinger (2006) and Pisinger and Ropke (2007), who use 25,000 LNS iterations and 10 replications for large size problems and 5 replications for small size problems. Note also that Ke et al. (2008) uses 2000 for the maximal number of cycles and 20 for the number of ants.

Table 2 reports the best reward average for each algorithm for each sub problem set. With the exception of our ALNS, PSOMA and PSOiA, all values are taken from the original literature. The results of PSOMA and PSOiA are obtained through email communication with their developers. Some values are missing in column GLS because Vansteenwegen et al. (2009) do not report them. Table 2 reveals that problem sets 1–3 are relatively easy since most of the algorithms find the same best solutions. PSOiA and ALNS are the best performers, obtaining the best solutions with the largest reward average for all of the problem sets.

In order to determine if the number of iterations was a factor, we tested the algorithm with I_{max} (number of iterations) = 10,000. Our experimental results show that ALNS with an increased number of iterations can improve the best solution for problem instance p4.2.q from 1267 to 1268, which is the best known solution value, recently updated by Dang et al. (2012). The solution is depicted in Fig. 1. It implies that our ALNS with 10,000 iterations can identify the best known solutions for 386 problems out of 387 benchmark problems. For the only one remaining problem (p4.4.m), we attained 976 compared to 977 which is the known best solution. In short, our ALNS with 10,000 iterations can solve all the benchmark problems with a total gap of 1.

Table 3 reports the average computation time for the various algorithms. There are some missing values in the column TMH because Tang and Miller-Hooks (2005) do not report the values for all problem instances. Although it is hard to compare the computation times directly since the speed of computers are different, through the table we can see the computation times of our ALNS are comparable to the other approaches.

Table 4 summarizes the current best solution for each problem instance. Most of the values are derived from Archetti et al. (2007); many were found earlier by Chao et al. (1996a) or Tang and Miller-Hooks (2005). Several best solutions were obtained by Bouly et al. (2010) and Dang et al. (2012). The solution values in boldface are exact optimal solutions reported by Boussier et al. (2007) and they are used as upper bounds terminate condition in Dang et al. (2011) and our method.

To evaluate the effects of the three improvement algorithms, Algorithms 3, 4, and 5 (including Algorithm 5-1 and Algorithm 5-2), we also experiment with different combinations of the algorithm using only 5000 iterations. Table 5 shows the experimental results. Columns 2 and 3 reiterate the results of the ALNS method shown in Tables 2 and 3. Columns 4 and 5 show the results of the ALNS method without Algorithm 3 (the local search improvement algorithm). Columns 6 and 7 show the results for the ALNS method without Algorithm 4 (the shifting and insertion algorithm). Finally, columns 8 and 9 show the results for the ALNS method without Algorithms 5 (the replacement Algorithm of 5-1 and 5-2). The Reward and Time columns show the average values over the ten trials for each problem set. The last row gives the summary values, total reward and total time for all of the problem sets.

Without Algorithm 3, the computation time is not significantly affected but the total solution gap becomes large (98). Similarly,

without Algorithm 4, this gap is 22, larger than the case with Algorithm 4. Nevertheless, it still outperforms PSOMA. Without Algorithm 5, the ALNS runs quite rapidly but cannot identify best known solutions for 3 benchmark problems. In summary, the local search improvement, the shifting and insertion algorithm and the replacement algorithm all contribute the solution quality.

5. Conclusions

This paper has proposed an augmented large neighborhood search (ALNS) method with three improvement algorithms, local search improvement algorithm, shift and insertion algorithm, and replacement algorithm, for the team orienteering problem. This particular combination of algorithms within the LNS framework can generate the best known solutions for 386 of 387 benchmark problems and solve the remaining one instance with a gap of 1, while the computation time of the proposed approach is comparable to the other approaches. From these results, we conclude that the proposed approach outperforms the existing approaches.

Acknowledgements

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (No. 2012R1A1A2005243).

References

Ahuja, R. K., Ergun, O., Orlin, J. B., & Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics, 123*, 72–102.

Archetti, C., Hertz, A., & Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. *Journal of Heuristics, 13*, 49–76.

Bouly, H., Dang, D.-C., & Moukrim, A. (2010). A memetic algorithm for the team orienteering problem. *4OR, 8*, 49–70.

Boussier, S., Feillet, D., & Gendreau, M. (2007). An exact algorithm for team orienteering problems. *4OR, 5*, 211–230.

Butt, S., & Cavalier, T. (1994). A heuristic for the multiple tour maximum collection problem. *Computers and Operations Research, 21*, 101–111.

Butt, S., & Ryan, D. M. (1999). An optimal procedure for the multiple tour maximum collection problem using column generation. *Computers and Operations Research, 26*, 427–441.

Chao, I.-M., Golden, B. L., & Wasil, E. A. (1996a). The team orienteering problem. *European Journal of Operational Research, 88*, 464–474.

Chao, I.-M., Golden, B. L., & Wasil, E. A. (1996b). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research, 88*, 475–489.

Dang, D., Guibadj, R. N., & Moukrim, A. (2012). An effective PSO-inspired algorithm for the team orienteering problem. (Working paper).

Dang, D.-C., Guibadj, R. N., & Moukrim, A. (2011). A PSO-based memetic Algorithm for the team orienteering problem. Applications of evolutionary computation. *Lecture Notes in Computer Science, 6625*, 471–480.

Feillet, D., Dejax, P., & Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science, 39*, 188–205.

Golden, B., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics, 34*, 307–318.

Hansen, P., & Mladenovic, N. (1997). Variable neighborhood search. *Computers and Operations Research, 24*(11), 1097–1100.

Jun, Y., & Kim, B.-I. (2012). New best solutions to VRPSPD benchmark problems by a perturbation based algorithm. *Expert Systems with Applications, 39*(5), 5641–5648.

Ke, L., Archetti, C., & Feng, Z. (2008). Ants can solve the team orienteering problem. *Computers and Industrial Engineering, 54*, 648–665.

Laporte, G., & Martello, S. (1990). The selective travelling salesman problem. *Discrete Applied Mathematics, 26*, 193–207.

Or, I. (1976). *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. Ph.D. Thesis. Evanston, IL: Department of Industrial Engineering and Management Sciences, Northwestern University.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research, 34*, 2403–2435.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*(4), 455–472.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., & Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics, 159*, 139–171.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems, Technical report, Scotland: Department of Computer Science, University of Strathclyde.

Tang, H., & Miller-Hooks, E. (2005). A tabu search heuristic for the team orienteering problem. *Computers and Operations Research, 32*, 1379–1407.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem*. Philadelphia, PA: SIAM.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *Journal of the Operational Research Society, 35*, 797–809.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Van Oudheusdem, D. (2009). A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research, 196*, 118–127.