A PSO-Based Memetic Algorithm for the Team Orienteering Problem

Duc-Cuong Dang[1], Rym Nesrine Guibadj[1,2], and Aziz Moukrim[1]

[1] Université de Technologie de Compiègne
Heudiasyc, CNRS UMR 6599, BP 20529, 60205 Compiègne, France
[2] VEOLIA Transport, MERCUR subsidiary
15, rue du Louvre, 75001 Paris, France
{duc-cuong.dang,rym-nesrine.guibadj,aziz.moukrim}@hds.utc.fr

Abstract. This paper proposes an effective Particle Swarm Optimization (PSO)-based Memetic Algorithm (PSOMA) for the Team Orienteering Problem (TOP). TOP is a particular vehicle routing problem whose aim is to maximize the profit gained from visiting clients while not exceeding a travel cost/time limit. Our PSOMA features optimal splitting techniques and genetic crossover operators. Furthermore, the memetic characteristic of our PSOMA is strengthened by an intensive use of local search techniques and also by a low value of 0.07 for inertia. In our experiments with the standard benchmark for TOP, PSOMA attained a gap of only 0.016%, as compared to 0.041%, the best known gap in the literature.

Keywords: Swarm Intelligence, Metaheuristics, Team Orienteering Problem, Optimal Split.

Introduction

The term Team Orienteering Problem (TOP), first introduced in [9], comes from an outdoor game played in mountainous or forested areas. In this game a team of several players try to collect as many reward points as possible within a given time limit. The Vehicle Routing Problem (VRP), analogous to the game that we denote simply by TOP, is the problem where a limited number of vehicles are available to visit customers from a potential set, the travel time of each vehicle being limited by a time quota, customers having different corresponding profits, and each customer being visited once at most once. The aim of TOP is to organize an itinerary of visits so as to maximize the total profit.

TOP is a variant of the Orienteering Problem (OP, also known as the Selective Traveling Salesman Problem) for multiple vehicles. As an extension of OP [10], TOP is clearly NP-Hard. OP and its variants have attracted a good deal of attention in recent years as a result of their practical applications and their hardness [5, 7, 9, 10, 14, 16, 20, 21, 22]. Readers are referred to [23] for a recent survey of these problems.

In this paper we are interested in TOP as the core variant of OP for multiple vehicles. This work was motivated by several lines of research first put forward

C. Di Chio et al. (Eds.): EvoApplications 2011, Part II, LNCS 6625, pp. 471–480, 2011.

by Veolia Environnement [4, 5]. Solving TOP to optimality has not received much attention. As far as we know, there are only two exact algorithms for TOP [6, 8]. Both are branch-and-price algorithms, but the second has the advantage of being easily adaptable to different variants of TOP. In contrast to exact solving approaches, a number of heuristics and metaheuristics have been developed for TOP [1, 4, 9, 12, 19, 20]. Three of these methods are considered to be state-of-the-art algorithms for TOP. The first is the slow version of Variable Neighborhood Search (SVNS) in [1]. The second is the Memetic Algorithm (MA) in [4], and the third is the slow version of Path Relinking approach (SPR) in [19].

The main contribution of this paper is a new memetic algorithm, called PSOMA, that can provide high quality solutions for TOP. The algorithm is relatively close to MA proposed in [4] and features the same basic components such as tour-splitting technique, population initializer and local search neighborhoods. However the global scheme has been changed to Particle Swarm Optimization (PSO): the recombination operator taking account of three sequences instead of two in MA and especially configurable with PSO parameters; a *swarm* of *particles*, i.e. couples of sequences, instead of a population of sequences in MA; a systematical application of recombination operator to every particle of the swarm in comparison to the stochastic selection of sequences for crossover operator in MA. With this memetic variant of PSO we were able to determine that good quality solutions require a very low value for *inertia*, and this can be attributed to the memetic characteristic of the algorithm. Experiments conducted on standard benchmark instances have shown clearly that with such an inertia, PSOMA was able to obtain better results than the state-of-the-art algorithms, including MA, with less computational effort.

The remainder of this paper is organized as follows. Section 1 provides a formal formulation of TOP. Our PSO-based Memetic Algorithm (PSOMA) is described in Section 2. Section 3 describes our empirical method for tuning PSOMA parameters and discusses computational results on benchmark instances. Finally, some conclusions and further developments are discussed in Section 4.

1 Formulation of the Problem

TOP is modeled with a graph $G = (V \cup \{d\} \cup \{a\}, E)$, where $V = \{1, 2, ..., n\}$ is the set of vertices representing customers, $E = \{(i, j) \mid i, j \in V\}$ is the edge set, and d and a are respectively departure and arrival vertices for vehicles. Each vertex i is associated with a profit P_i, and each edge $(i, j) \in E$ is associated with a travel cost $C_{i,j}$ which is assumed to be symmetric and satisfying the triangle inequality. A tour r is represented as an ordered list of $|r|$ customers from V, so $r = (i_1, \ldots, i_{|r|})$. Each *tour* begins at the departure vertex and ends at the arrival vertex. We denote the total profit collected from a tour r as $P(r) = \sum_{i \in r} P_i$, and the total travel cost/time as $C(r) = C_{d,i_1} + \sum_{x=1}^{x=|r|-1} C_{i_x,i_{x+1}} + C_{i_{|r|},a}$. A tour r is feasible if $C(r) \leq L$ with L being a predefined travel cost/time limit. The fleet is composed of m identical vehicles. A *solution* S is consequently a set of m (or fewer) feasible tours in which each customer is visited only once.

The goal is to find a solution S such that $\sum_{r \in S} P(r)$ is maximized. One simple way of reducing the size of the problem is to consider only *accessible* clients. A client is said to be accessible if a tour containing only this client has a travel cost/time less than or equal to L. For mixed integer linear programming formulations of TOP see [6, 8, 12, 23].

2 PSO-Based Memetic Algorithm

Particle Swarm Optimization (PSO) [13, 18] is one of swarm intelligence techniques with the basic idea of simulating the collective intelligence and social behavior of wild animals that can be observed, for example, in fish schooling and bird flocking. PSO was first used for optimization problem in continuous space as follows. A set known as a *swarm* of candidate solutions, referred to as *particles*, is composed of positions in the search space. The swarm explores the search space according to Equations 1 and 2. In these equations, x_i^t and v_i^t are respectively the position and the velocity of particle i at instant t. Three values w, c_1 and c_2, called respectively *inertia*, *cognitive* factor and *social* factor, are parameters of the algorithm. Two values r_1 and r_2 are random numbers generated in the interval $[0, 1]$. Each particle i memorizes its best known position up to instant t as x_i^{lbest}, and the best known position up to instant t for the swarm is denoted as x^{gbest}.

$$v_i^{t+1}[j] = w.v_i^t[j] + c_1.r_1.(x_i^{lbest}[j] - x_i^t[j]) + c_2.r_2.(x^{gbest}[j] - x_i^t[j]) \quad (1)$$

$$x_i^{t+1}[j] = x_i^t[j] + v_i^{t+1}[j] \quad (2)$$

With this design, PSO is highly successful at performing optimizations in continuous space [2, 11]. In contrast, when applied to problems of combinatorial optimization, PSO encounters difficulties in interpreting positions and velocities, as well in defining position update operators. As result, there are a variety of discrete PSO variants (DPSO) [3], and it is difficult to choose an appropriate variant for any given combinatorial optimization such as TOP.

Memetic algorithms (MA) [15] represent an optimization technique that attempts to simulate social evolution rather than genetic or biological evolution. Most MA designs incorporate various local search techniques into a global search scheme, e.g. a genetic algorithm. MA and PSO are both based on social evolution or behavior rather than biological ones, and there are benefits to be gained from combining techniques into a single form, that we call a *PSO-based MA*, or *PSOMA*, for solving combinatorial optimization problems. This section examines PSOMA in detail as a technique for solving TOP.

2.1 Position Representation and Improvement of Positions

A position in PSOMA is a permutation π of all accessible clients in a particular problem scenario. [4] gives a *splitting* algorithm that optimally transforms the permutation into a solution of TOP in $O(m.n^2)$. This algorithm guarantees that if the tours forming one of the optimal solutions of TOP are subsequences in

a permutation π^*, the optimal solution will be returned in the output of the algorithm. This is made possible by considering only *saturated* tours respecting the permutation order, i.e subsequences of the permutation respecting the travel cost/time limit and containing the highest number of clients, then formulating the selection of the most profitable m saturated tours as a longest path problem under a k-cardinality constraint (kCC-LPP) on an auxiliary acylic graph. Here the value of k is $2.m+1$ and kCC-LPP can be solved efficiently through dynamic programming. In our approach, we use the PSO principle to find such a π^* permutation.

The authors of [4] also provided an efficient local search technique (LS) to improve a given permutation. Their intensive use of LS made the method a memetic method. Consequently, in our PSOMA, whenever a new position is found it has a *pm* probability of being improved using the same LS. It contains 3 neighborhoods:

- *shift operator*: evaluate all possibilities of moving a customer i from its original position to any other position in the permutation.
- *swap operator*: evaluate all possibilities of exchanging two customers i and j in the permutation.
- *destruction/repair operator*: evaluate the possibility of removing a random number (between 1 and $\frac{n}{m}$) of clients from an identified solution and then rebuilding the solution with a best insertion algorithm. Best insertion uses the well-known intuitive criterion for TOP that maximizes $\frac{\Delta P_i}{\Delta C_i}$ [4, 19].

The procedure is as follows. One neighborhood is randomly chosen to be applied to the particle position. As soon as an improvement is found, it is applied and the LS procedure is restarted from the new improved position. The LS is stopped when all neighborhoods are fully applied without there being any improvement.

2.2 Genetic Crossover Operator to Update Position

In combinatorial optimization, the particle position update of PSO can be interpreted as a recombination of three positions/solutions according to inertia, cognitive and social parameters. There are various ways of defining this kind of recombination operator [3]. In our approach, the recombination operator is similar to a genetic crossover whose core component is an extraction of l clients from a permutation π while avoiding clients from M set. This avoiding set allows extracted subsequences from successive calls of the core component to be non collapsed, so then they can be reassembled into a new valid sequence. The extracted subsequence is denoted π_M^l and the procedure is described as follows:

- Step 1: generate a random location r in π and initialize π_M^l to empty.
- Step 2: browse clients from $\pi[r]$ to $\pi[n]$ and add them to the end of π_M^l if they are not in M. The set of clients to be avoided M is updated during the process. If $|\pi_M^l|$ reaches l then terminate, otherwise go to Step 3.
- Step 3: browse clients from $\pi[r]$ down to $\pi[1]$ and add them to the beginning of π_M^l if they are not in M. The set of clients to be avoided M is updated during the process. If $|\pi_M^l|$ reaches l then terminate.

With the core component, the position update procedure of particle x from the swarm S with respect to the three PSO parameters w, c_1 and c_2 is described as follows. For convenience, the current, local best and global best positions of the particle are denoted respectively $S[x].pos$, $S[x].lbest$ and $S[best].lbest$:

- Phase 1: apply sequentially but in a random order the core component to extract subsequences from $S[x].pos$, $S[x].lbest$ and $S[best].lbest$ with a common set of clients to be avoided M, initialized to empty. The desired numbers of clients to be extracted for $S[x].pos$, $S[x].lbest$ and $S[best].lbest$ are respectively $w.n$, $(1-w).n.\frac{c_1.r_1}{(c_1.r_1+c_2.r_2)}$ and $(1-w).n.\frac{c_2.r_2}{(c_1.r_1+c_2.r_2)}$. Here r_1 and r_2 are real numbers whose values are randomly generated in the interval $[0, 1]$ with a uniform distribution.
- Phase 2: link three extracted subsequences in a random order to update $S[x].pos$.

Our particle position update procedure therefore works with the standard PSO parameters w, c_1 and c_2, the only restriction being that w has to be in the interval $[0, 1]$. Our PSOMA can be classified as PSO with position only, because no velocity vector is employed. It might be remarked that the core component was created to adapt to a linear permutation order, but it can easily be adapted to a circular order by changing Step 3.

2.3 Swarm Local Best Update

In some situations, PSO can be trapped in a local optimum, especially when all the local best positions of particles in the swarm are identical. To avoid this premature convergence, whenever a new position is found by a particle x in the swarm S, instead of updating $S[x].lbest$, the algorithm will search for an appropriate particle y in the swarm and update $S[y].lbest$. The update rule is similar to [17] but simplified:

1. The update procedure is applied if and only if the performance of new position $S[x].pos$ is better than the worst local best $S[worst].lbest$.
2. If there exists a particle y in the S such that $S[y].lbest$ is similar $S[x].pos$, then replace $S[y].lbest$ with $S[x].pos$.
3. If no such particle y according to Rule 2 exists, replace $S[worst].lbest$ with $S[x].pos$. Each successful application of this rule indicates that a new local best has been *discovered* by the swarm.

The similarity measure in Rule 2 is based on two criteria: the total collected profit and the travel cost/time of the identified solution. Two positions are said to be similar or identical if the evaluation procedure on these positions returns the same profit and a difference in travel cost/time that is less than δ. The implementation of the update rules was made efficient through the use of a binary search tree to sort particles by the performance of their local best positions using the two criteria.

2.4 Global PSO Algorithm

Particle positions in the swarm, including local best positions, are initialized to a random sequence. In order to accelerate the algorithm, a small portion of the swarm containing K particles will have local best positions generated with Iterative Destruction/Construction Heuristics (IDCH) described in [4]. PSOMA is stopped when the upper bound [6] is reached or after $itermax$ consecutive iterations have failed to give rise to new local best. The global scheme is summarized in Algorithm 1.

```
Data: S a swarm of N particles;
Result: S[best].lbest best position found;
begin
    initialize and evaluate each particle in S;
    iter ← 1;
    while iter ≤ itermax do
        foreach x in [1..N] do
            update S[x].pos (see Section 2.2);
            if rand(0,1) < pm then
                apply local search on S[x].pos (see Section 2.1);
            end
            evaluate S[x].pos (see Section 2.1);
            update lbest of S (see Section 2.3);
            if (a new local best is discovered) then
                iter ← 1;
            else
                iter ← iter + 1;
            end
        end
    end
end
```

Algorithm 1. Basic PSOMA scheme

3 Parameter Configuration and Numerical Results

Our algorithm is coded in C++ using the Standard Template Library (STL) for data structures. The program is compiled with GNU GCC in a Linux environment, and all experiments were conducted on an AMD Opteron 2.60 GHz. This section describes in detail our method for tuning PSOMA parameters, in particular the PSO inertia parameter. We also compare the performances of our approach with those of state-of-the-art algorithms in the literature, using 387 instances from the standard benchmark for TOP [9]. These instances comprise 7 sets. Inside each set the original number of customers and customer positions are constant, however the maximum tour duration L varies, then the number of accessible clients are different for each instance. The number of vehicles m also varies between 2 and 4.

3.1 Parameter Tuning

We decided to reuse most of the population management parameter values listed in [4]. To simplify the PSO parameter tuning, we decided to set the ratio between c_1 and c_2 to equality, meaning that a particle has the same probability of moving either to local or to global best. These parameters are summarized as follows.

- N, population size, is set to 40,
- K, the number of local best positions initialized with IDCH, is set to 5,
- $itermax$, the stopping condition, is set to $10 \cdot \frac{n}{m}$,
- pm, the local search rate, is set to $1 - \frac{iter}{itermax}$,
- δ, the similarity measurement of particles, is set to 0.01,
- c_1, the PSO cognitive factor, is set to 1.41,
- c_2, the PSO social factor, is set to 1.41.

The only remaining parameter is inertia w, with values from 0 to 1.0. According to [18] the choice of the inertia value is crucial to the performance of PSO. So two experiments were designed to determine the best value for w and also the best variant of the algorithm to use.

Our first experiment was designed to identify the prominent region for the value of w. The algorithms were tested with large steps between values of w, these values being 0, 0.1, 0.3, 0.5, 0.7 and 0.9. For each of these values the algorithms were executed 10 times for each instance of the benchmark. The sum of the highest profits for each instance could then be compared with the corresponding sum from the best known results in the literature. These best

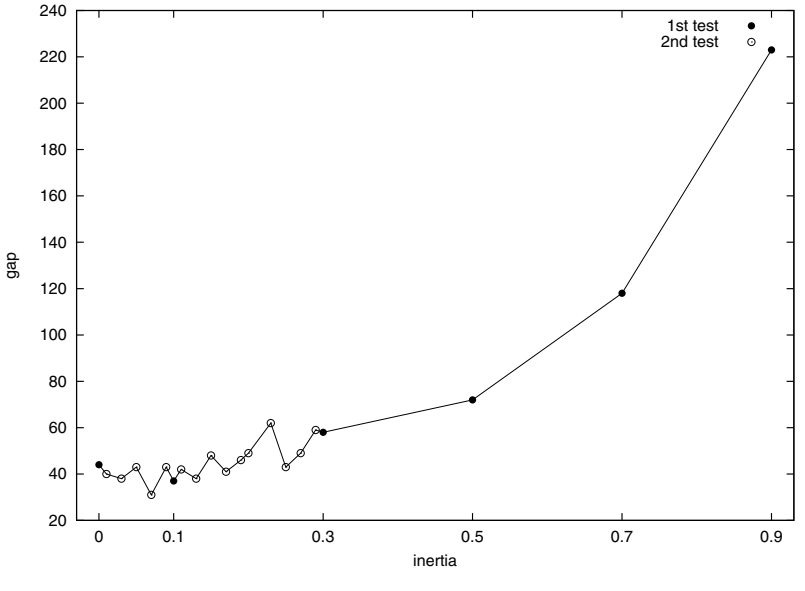

Fig. 1. Gap to the best known results for PSOMA with different values of w

Table 1. Comparison with state-of-the-art algorithms

Method	Parameter	Average CPU time on data sets							Gap	
		1	2	3	4	5	6	7	Profit	Percent
PSOMA	$w = 0.10$	0.14	0.01	0.51	82.94	12.81	3.87	55.98	37	0.019
	$w = 0.07$	0.15	0.01	0.51	78.45	12.87	3.99	54.54	31	0.016
SPR[a]	*slow*	–	–	–	367.40	119.90	89.60	272.80	117	0.060
MA[b]	$k = 5$	1.31	0.13	1.56	125.26	23.96	15.53	90.30	81	0.041
	$k = 10$	1.74	0.24	2.06	170.21	33.52	22.33	109.00	54	0.028
SVNS[c]	*slow*	7.78	0.03	10.19	457.89	158.93	147.88	309.87	85	0.043

[a] Computations were carried out on an Intel Xeon 2.50 GHz
[b] Computations were carried out on an Intel Core 2 Duo 2.67 GHz
[c] Computations were carried out on an Intel Pentium 4 2.80 GHz

results are collected from [1, 4, 12, 19, 20] but not from [9] because the authors used a different rounding precision and some of their results exceeded the upper bounds given in [6]. Here we are interested in the difference (or *gap*) between those two sums. On the basis of the results shown in Figure 1 for this experiment, we chose to perform the next test using different values of w in the prominent region between 0 and 0.3.

In the second test, PSOMA was executed, as a complement to the first test, with the following values for w: 0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15, 0.17, 0.19, 0.20, 0.21, 0.23, 0.25, 0.27 and 0.29. Figure 1 gives a full graphical representation of the differences with the best known profits from the literature. The figure shows that the best value that we found for w is 0.07.

3.2 Numerical Comparisons with Existing Methods

Numerical results with comparison to the state-of-the-art algorithms, MA in [4], SVNS in [1] and SPR in [19], are given in Table 1. In this table, the relative gap in percent is the gap in profit over the sum of profits from the best known solutions in the literature. Results of SPR were not reported for all instances in [19], so CPU times of SPR for sets 1, 2 and 3 are not reported in the table. But according to the authors, the best known results in the literature can be used as score of SPR for unreported instances, hence its gap can be deduced.

After the first test we noticed that PSOMA with $w = 0.1$ already outperformed all the three state-of-the-art algorithms. MA, SVNS and SPR gave gaps of 81, 85 and 117 respectively in relation to the standard benchmark, while the gap given by PSOMA is 37. Even better results were obtained with $w = 0.07$, where the gap was reduced to 31. However, in order for the comparison with [4] to be fair, we obtained the source code from the authors and applied the same test protocol to MA with the stopping condition set to $10 \cdot \frac{n}{m}$, and with

10 executions per instance. These results are also shown in Table 1, the gap for MA obtained from this test was 54, rather than the original 81. Regarding computational time, PSOMA required less CPU time than MA.

4 Conclusion

This paper presented a novel memetic algorithm for the Team Orienteering Problem which incorporates the PSO principle into the main scheme. Numerical results on the standard benchmark for TOP demonstrate the competitiveness of this approach. The new PSOMA outperformed the prior GA/MA design in terms both of computation time and solution quality. Because the old GA/MA is one of the state-of-the-art algorithms, the new PSOMA has considerably improved the solving method for TOP, the newly attained gap being 0.016%. This success is due to the good design of the recombination operator for updating particle positions, as well as to an appropriate choice of parameters. Furthermore, a small inertia value, corresponding to the best configuration of the algorithm, strengthens the memetic characteristic of the approach. In summary, the results presented in this paper are encouraging for the application of Particle Swarm Optimization for solving combinatorial problems, as has already been indicated in [3]. As part of our future development of this method we intend to investigate the performance of the approach for certain variants of TOP, such as variants with time windows or with capacity constraints.

References

[1] Archetti, C., Hertz, A., Speranza, M.: Metaheuristics for the team orienteering problem. Journal of Heuristics 13(1) (February 2006)
[2] Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization. part I: background and development. Natural Computing 6(4), 467–484 (2007)
[3] Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization. part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Natural Computing 7(1), 109–124 (2008)
[4] Bouly, H., Dang, D.C., Moukrim, A.: A memetic algorithm for the team orienteering problem. 4OR 8(1), 49–70 (2010)
[5] Bouly, H., Moukrim, A., Chanteur, D., Simon, L.: Un algorithme de destruction/construction itératif pour la résolution d'un problème de tournées de véhicules spécifique. In: MOSIM 2008 (2008)
[6] Boussier, S., Feillet, D., Gendreau, M.: An exact algorithm for team orienteering problems. 4OR 5(3), 211–230 (2007)
[7] Butt, S., Cavalier, T.: A heuristic for the multiple tour maximum collection problem. Computers and Operations Research 21, 101–111 (1994)
[8] Butt, S.E., Ryan, D.M.: An optimal solution procedure for the multiple tour maximum collection problem using column generation. Computers and Operations Research 26, 427–441 (1999)
[9] Chao, I.M., Golden, B., Wasil, E.: The team orienteering problem. European Journal of Operational Research 88 (1996)

[10] Golden, B., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics 34, 307–318 (1987)
[11] Kameyama, K.: Particle swarm optimization - a survey. IEICE Transactions 92-D(7), 1354–1361 (2009)
[12] Ke, L., Archetti, C., Feng, Z.: Ants can solve the team orienteering problem. Computers and Industrial Engineering 54(3), 648–665 (2008)
[13] Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
[14] Montemanni, R., Gambardella, L.: Ant colony system for team orienteering problems with time windows. Foundations of Computing and Decision Sciences 34, 287–306 (2009)
[15] Moscato, P.: Memetic Algorithms: a short introduction. In: New Ideas in Optimization, pp. 219–234. McGraw-Hill, New York (1999)
[16] Schilde, M., Doerner, K.F., Hartl, R.F., Kiechle, G.: Metaheuristics for the bi-objective orienteering problem. Swarm Intelligence 3(3), 179–201 (2009)
[17] Sha, D.Y., Hsu, C.Y.: A hybird particle swarm optimization for job shop scheduling problem. Computers and Industrial Engineering 51(4), 791–808 (2006)
[18] Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Evolutionary Programming, pp. 591–600 (1998)
[19] Souffriau, W., Vansteenwegen, P., Van den Berghe, G., Van Oudheusden, D.: A path relinking approach for the team orienteering problem. Computers & Operations Research 37(11), 1853–1859 (2010)
[20] Tang, H., Miller-Hooks, E.: A tabu search heuristic for the team orienteering problem. Computer & Operations Research 32, 1379–1407 (2005)
[21] Tricoire, F., Romauch, M., Doerner, K.F., Hartl, R.F.: Heuristics for the multi-period orienteering problem with multiple time windows. Computers & Operations Research 37, 351–367 (2010)
[22] Vansteenwegen, P., Souffriau, W., Van den Berghe, G., Van Oudheusden, D.: Metaheuristics for tourist trip planning. In: Metaheuristics in the Service Industry. LNEMS, vol. 624, pp. 15–31. Springer, Heidelberg (2009)
[23] Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: A survey. European Journal of Operational Research 209(1), 1–10 (2011)