A Memetic Approach for the Orienteering Problem

Valentino Santucci[1][0000−0003−1483−7998] and
Marco Baioletti[2][0000−0001−5630−7173]

[1] Department of Humanities and Social Sciences
University for Foreigners of Perugia, Italy
valentino.santucci@unistrapg.it

[2] Department of Mathematics and Computer Science
University of Perugia, Italy
marco.baioletti@unipg.it

Abstract. In this paper we present a new memetic approach to solve the orienteering problem. The key method of our proposal is the procedure `ReduceExtend` which, starting from a permutation of all the vertices in the orienteering problem, produces a feasible path with a locally optimal score. This procedure is coupled with an evolutionary algorithm which navigate the search space of permutations. In our experiments we have considered the following algorithms: the algebraic differential evolution algorithm, and the three continuous algorithms CMA-ES, DE and PSO equipped with the random key technique. The experimental results show that the proposed approach is competitive with the state of the art results of some selected benchmark instances.

1 Introduction

The Orienteering Problem is an important routing problem widely studied in literature [18]. The aim is to generate a path through a set of given nodes, which would maximize total score and would not exceed the given budget.

Formally, an instance of the Orienteering Problem (OP) is given by means of a complete graph $G = (V, E)$ where $V = \{v_0, \ldots, v_{n+1}\}$ is the vertex set and E is the edge set. Moreover, every vertex $v_i \in V$ has a score $s_i \geq 0$. The first and last vertex have null score, i.e., $s_0 = s_{n+1} = 0$. Every edge $e_{ij} \in E$ is marked with a travel time $t_{ij} \geq 0$. Finally, a time budget $T_{\texttt{max}} > 0$ is also given.

The aim of the OP is to determine a path through some vertices of G whose total travel time is limited by $T_{\texttt{max}}$ and its total score is maximized. Formally, given the sequence of vertices $p = \langle p_0, p_1, \ldots, p_l, p_{l+1} \rangle$, its total travel time $t(p)$ and total score $s(p)$ are defined as follows:

$$t(p) = \sum_{i=0}^{l} t_{p_i, p_{i+1}}; \tag{1}$$

$$s(p) = \sum_{i=0}^{l+1} s_{p_i}. \tag{2}$$

Hence, the OP aims at a sequence of vertices $p = \langle p_0, p_1, \ldots, p_l, p_{l+1} \rangle$ such that:

- the first and the last vertexes are fixed to the given start and end vertices, i.e., $p_0 = v_0$ and $p_{l+1} = v_{n+1}$;
- each vertex of V appears at most once in p, i.e., p has to be an Hamiltonian path over a subset of V;
- the total travel time $t(p)$ satisfies inequality (3):

$$t(p) \leq T_{\texttt{max}}; \tag{3}$$

- the total score $s(p)$ is maximized.

The OP is an NP-hard optimization problem [18, 41]. Moreover, the popular traveling salesman problem (TSP) is a special case of the OP. Indeed, a TSP instance is an OP instance with an infinite budget of time and constant scores on the vertices. For this reason, the search space of the OP is larger than that of the TSP thus, in this sense, solving the OP is computationally more difficult than solving the TSP.

In this paper we propose a novel evolutionary-memetic approach for solving the OP. Evolutionary algorithms are widely popular meta-heuristics which allow to handle computationally difficult optimization problems. They iteratively evolve a population of solutions by means of genetic or swarm-intelligence operators. Often, they are combined with local search methods in order to perform local refinements and to improve their effectiveness. The hybridizations of an evolutionary algorithm with a local search procedure are generally known in literature as memetics algorithms [31]. Here, we introduce a novel heuristic local search procedure for the OP and we combine it with evolutionary algorithms designed to navigate the search space of permutations.

The rest of the paper is organized as follows:

- Section 2 presents a short overview of the OP literature;
- Section 3 describes the main scheme of the approach proposed;
- Section 4 introduces the memetic procedure for the OP;
- Section 5 presents and discusses the experimental results;
- Section 6 concludes this work by summarising the key research outputs and drawing some considerations for possible future developments.

2 Related Work

In this paper we are interested to the classical OP as defined in Section 1. Nevertheless, note that other OP variants have been introduced in the literature such as, for example, the team orienteering problem [12] where a fleet of traveling agents is considered, the OP with time windows [24], the time dependent OP [42] where the traveling times are functions of some network's properties, the

stochastic OP [43], or the generalized OP [15] where the objective function is a non-linear function.

The classical OP arises in several applications: a traveling salesperson with not enough time to visit all possible cities [40], trucks delivering goods to customers where each customer has a priority and the truck has fuel constraint [17], the single-ring design problem when building telecommunication networks [39], building a mobile tourist guide [37], etc.

In the beginning, several exact algorithms have been proposed to solve the OP. For instance, [25] and [32] use a branch-and-bound algorithm, [26] introduces a cutting plane method for the OP, and [14, 16] further extended the branch-and-cut approach. A classification of the exact algorithms for the OP is presented in [13].

Unfortunately, as for other combinatorial problems, exact approaches allow to handle only relatively small instances. Therefore, meta-heuristics are nowadays the most effective approaches for the OP. Among these: variants of the particle swarm optimization algorithm have been employed in [36] and [44], a variable neighborhood search approach is adopted in [27], greedy randomized search procedures for the OP are introduced in [11] and [28], while an evolutionary algorithm for the OP is depicted in [23].

Finally, two survey articles about the OP are [18] and [41].

3 The proposed memetic approach

We start by noting that, since the first and last vertices of the path to find are fixed to, respectively, v_0 and v_{n+1}, an OP solution can be represented as a permutation of some vertices of $\tilde{V} = V \setminus \{v_0, v_{n+1}\}$.

Let P be the search space of all possible paths from v_0 to v_{n+1}, then P contains all the possible permutations for every subset $U \subseteq \tilde{V}$. Hence,

$$|P| = \sum_{k=0}^{n} \binom{n}{k} k! = (n+1)!$$

Therefore the search space of the OP is larger than that of the more classical permutation-based problems like, for instance, PFSP (Permutation Flowshop Scheduling Problem), QAP (Quadratic Assignment Problem), TSP (Traveling Salesman Problem), or LOP (Linear Ordering Problem).

In the following: we will use the term *full permutation* to refer to a permutation of the whole set $\tilde{V}$, and we denote by S the set of all the full permutations. Note however that S is only a subset of all the possible OP solutions in P, i.e., $S \subset P$.

Our proposal is to tackle the OP by means of two algorithmic components:

1. an evolutionary algorithm $\mathcal{A}$ that navigates the space of full permutations S, and

2. the procedure `ReduceExtend` which takes in input a full permutation $\pi \in S$ and produces a path $p \in P$ that is feasible under the constraint (3) and whose score $s(p)$ is locally optimal (under some locality definitions as explained in Section 4).

$\mathcal{A}$ and `ReduceExtend` interact with each other by following the memetic principles [31]. $\mathcal{A}$ is the high level scheme which moves in the (smaller) space of full permutations and whose search is guided by a fitness function which embed the `ReduceExtend` procedure. Indeed, given an individual $\pi \in S$ – in the $\mathcal{A}$'s population – its fitness is computed as $s(\texttt{ReduceExtend}(\pi))$, i.e., the OP score of the feasible and locally optimal path returned by the `ReduceExtend` procedure.

This scheme allows to exploit the available evolutionary algorithms (EAs) devised for the permutations search space [33, 1, 10]. Two main classes of permutation-based EAs are considered:

1. the EAs whose individuals are explicitly represented as full permutations in S, and
2. the EAs which evolve populations formed by real vector individuals and that rely on the random-key procedure in order to decode a real vector into a full permutation [10].

For the first class we have considered the recently proposed Algebraic Differential Evolution for Permutations (ADEP) [33, 6], while, for the second class we have chosen the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [19], the Differential Evolution (DE) [38, 29], and the Particle Swarm Optimization (PSO) [22].

The second class algorithms adopts the random key procedure [10] to convert a real vector $x \in \mathbb{R}^n$ to an n-length permutation $\pi \in S$ by replacing each entry x_i of x with the vertex $v_r \in \tilde{V}$ such that r is the ranking of x_i among the other entries of x.

Finally note that, embedding the memetic procedure inside the fitness function of the evolutionary algorithm at hand is referred in the literature as a Baldwinian approach [20], i.e., the individual of the EA is refined through local search but it is kept unmodified in the population of the EA. However, note that, since ADEP directly navigates the space of permutations, it has been possible to also devise a simple procedure which converts a generic OP solution in P to a full permutation in S. Therefore, differently from the second-class approaches, ADEP has also been implemented following the Lamarckian approach [20], i.e., the refined individual is modified in the population of the EA.

4 The `ReduceExtend` procedure

The procedure `ReduceExtend` takes in input a full permutation $\pi \in S$ and produces a path $p \in P$ which is feasible and locally optimal.

Three main subprocedure have been devised: `Reduce`, `Extend`, `ScrambleExtend`. They are connected together in the definition of `ReduceExtend` as follows:

$$\texttt{ReduceExtend}(\pi) = \texttt{ScrambleExtend}(\texttt{Extend}(\texttt{Reduce}(\pi))), \tag{4}$$

where π is a given full permutation, i.e., $\pi \in S$ and `ReduceExtend`(π) is a feasible and locally optimal path in P.

Therefore, `ReduceExtend` acts as a projection from the smaller space S to the larger space P. In the following subsections we describe the three subprocedures `Reduce`, `Extend`, `ScrambleExtend`.

Moreover, subsection 4.4 introduces the `PathToFullPermutation` operator that, given a path $p \in P$, stochastically generates a full permutation $\pi =$ `PathToFullPermutation`(p) which is consistent with p. As explained in Section 3, this operator allows to devise a Lamarckian variant of the ADEP algorithm.

4.1 `Reduce`

In this phase, the full permutation $\pi \in S$ is projected into the larger space P by producing a feasible path $p \in P$.

`Reduce` starts with $p = \pi$ and iteratively removes vertices from p till the constraint $t(p) \leq T_{\texttt{max}}$ is satisfied.

At each iteration, the vertex to be removed is selected to be the one which minimizes the normalized score lost $nsl(p_i)$ defined as the ratio between the score lost and the gain of travel time if the vertex p_i is removed from the path p. Formally,

$$nsl(p_i) = \frac{s(p_i)}{t_{p_{i-1}p_i} + t_{p_ip_{i+1}} - t_{p_{i-1}p_{i+1}}}. \tag{5}$$

Therefore, the `Reduce` procedure modifies the path in input by improving its total travel time $t(p)$ but penalizing its total score $s(p)$. The greedy choice based on the normalized score lost represents a tradeoff between the score lost and the time gain achieved during the iterations of `Reduce`.

4.2 `Extend`

The greedy behaviour of the `Reduce` procedure may produce a path p which can be extended-back without violating the $T_{\texttt{max}}$ constraint. Therefore, the procedure `Extend` iteratively inserts a vertex into p until the constraint is violated.

At each iteration, the vertex v – to insert in p – is selected to be the one with the largest score among the vertices not already belonging to p and such that, when inserted into p, does not violate the constraint. The selected vertex is inserted at the position of p which minimizes the travel time increase.

Therefore, oppositely with respect to `Reduce`, the `Extend` procedure modifies the path in input by improving its total score $s(p)$ but penalizing its total travel time $t(p)$. Moreover, the vertex and position selection criteria guarantee a good tradeoff between the score gain and the time lost achieved during the iterations of `Extend`.

4.3 `ScrambleExtend`

`ScrambleExtend` aims to improve the score of the path p by iteratively alternating two local search steps as done in variable neighborhood search procedures

[30]. The two refinement steps consist of a `2OPT` search [21], and one step of the `Extend` procedure previously described.

The `2OPT` search scrambles the vertices in p aiming to reduce the travel time $t(p)$ without decreasing the score $s(p)$. Indeed, no vertex is introduced or removed from p, they are only swapped.

The `2OPT` local search is widely adopted in the TSP [21]: given a path p it iteratively selects the best neighboring path till a local optimum is reached. Neighbors are defined by means of `2OPT` moves, i.e., the path p' is a neighbor of the path p if it is obtained by replacing two arcs in p with two arcs not in p (but using the same vertices of p). As described in [8], `2OPT` moves are equivalent to reversing chunks of a given path. Therefore, the `2OPT` neighborhood of p has size $\binom{|p|}{2}$.

Note that the `2OPT` local search has the effect of decreasing the travel time $t(p)$, thus creating room for the insertion of a new vertex. For this reason, every `2OPT` local search is followed by a single step of the `Extend` procedure (described in Section 4.2) which can possibly increase the score.

The iterations of `ScrambleExtend` terminate when it is no more possible to improve the score of the incumbent path. Note that, `ScrambleExtend` does not remove any vertex from the input path p, thus it can only increase its score $s(p)$. Anyway, the original ordering of vertices may be modified by the `2OPT` steps.

4.4 `PathToFullPermutation`

Given a feasible path $p \in P$, $PathToFullPermuation(p)$ generates a full permutation $\pi \in S$ which can be used to modify the individual of the evolutionary algorithm at hand following a Lamarckian style of evolution [20].

In order to promote the population diversity in the Lamarckian algorithm, `PathToFullPermutation` has been designed by following a simple stochastic mechanism: it iteratively insert a randomly selected vertex not yet in p in a random position of p till p is a full permutation.

Despite the stochastic approach, the items in the original p maintain the same relative order among them in the final full permutation.

5 Experiments

Experiments have been held using the 89 instances from the five benchmark suites listed in Table 1 and obtained from the website
`https://www.mech.kuleuven.be/en/cib/op`.

Five EAs have been considered: ADEP-B (Baldwinian), ADEP-L (Lamarckian), CMA-ES, DE and PSO. Note that, as described in Section 3 the latter three algorithms are equipped with the random-key decoding scheme and follows the Baldwinian approach.

All the five algorithms have been run 10 times per instance and every execution terminates as soon as the (known) optimal score has been reached or

Table 1. Benchmarks

Benchmark	#Instances	Size
Tsiligirides_set1	18	31
Tsiligirides_set2	11	21
Tsiligirides_set3	20	32
Chao_set64	14	64
Chao_set66	26	66

when 10 000 evaluations have been performed. Therefore, the goal of this experimentation is to investigate how many times and how quickly the optimum is reached.

Table 2 shows two data for each algorithm and every benchmark suites: the number of instances that have been solved in all the 10 executions, and the average number of fitness evaluations performed by the algorithm. Obviously, the first measure has to be maximized while the second one has to minimized. The last row of the table averages the results across the different benchmarks.

Table 2. Experimental results

Benchmark	ADEP-B	ADEP-L	CMA-ES	DE	PSO
Tsiligirides_set1	18/18	18/18	18/18	18/18	18/18
	87	**34**	95	79	114
Tsiligirides_set2	11/11	11/11	11/11	11/11	11/11
	35	**24**	42	28	32
Tsiligirides_set3	20/20	20/20	20/20	20/20	20/20
	30	**19**	28	29	26
Chao_set64	14/14	14/14	14/14	14/14	13/14
	466	**149**	728	618	685
Chao_set66	24/26	**26/26**	18/26	17/26	18/26
	1439	**260**	2220	2605	2297
Average	87/89	**89/89**	81/89	80/89	80/89
	411	**97**	623	672	631

Interestingly, all the algorithms have been able to reach the optimum in every instance at least once. Moreover, most of the times the optimum has been obtained in every execution and by employing a small number of fitness evaluations. These results clearly validates our memetic approach.

Regarding the differences among the different schemes, the Lamarckian variant of ADEP clearly outperforms all the other schemes. Moreover, also the Bald-

winian ADEP has been able to outperform the random-key-based CMA-ES, DE and PSO. This clearly suggests that, though the literature presents a large variety of numerical schemes equipped with the random key decoding scheme, the algorithms purposely defined to navigate a combinatorial search space, as ADEP, have to be preferred (at least in the OP case).

6 Conclusion and Future Work

In this paper we have introduced a novel memetic approach for the Orienteering Problem.

In particular, the main contribution is the `ReduceExtend` procedure which transforms any full permutation into a locally optimal feasible solution for the OP. `ReduceExtend` is composed by three subprocedures purposely designed to maximize the objective function by also satisfying the time constraint.

This memetic procedure allows to handle the OP by using any evolutionary algorithm which works on the permutation search space. In this work, we have adopted and experimentally compared five algorithms. In one case it has been possible to use the Lamarckian evolution by introducing a randomized procedure for converting back a feasible OP solution into a full permutation of items. This algorithm, namely ADEP-L, resulted to be the most effective in the experiments we carried out.

Moreover, all the five approaches were able to reach the known optimal values, thus validating our proposal.

As future work, we expect to provide an algebraic formalization of our procedure using the algebraic framework described in [4, 2, 35, 9, 34, 7]. This framework can also be used to handle the OP by using the concept of product group as done in [5]. Finally, the memetic approach can be extended to other algorithms like, for example, the ant colony optmization for the permutation space [3].

Acknowledgement

The research described in this work has been partially supported by: the research grant "Fondi per i progetti di ricerca scientifica di Ateneo 2019" of the University for Foreigners of Perugia under the project "Algoritmi evolutivi per problemi di ottimizzazione e modelli di apprendimento automatico con applicazioni al Natural Language Processing"; and by RCB-2015 Project "Algoritmi Randomizzati per l'Ottimizzazione e la Navigazione di Reti Semantiche" and RCB-2015 Project "Algoritmi evolutivi per problemi di ottimizzazione combinatorica" of Department of Mathematics and Computer Science of University of Perugia.

References

1. Baioletti, M., Milani, A., Santucci, V.: Algebraic particle swarm optimization for the permutations search space. In: Proc. of 2017 IEEE Congress on Evolutionary Computation (CEC 2017). pp. 1587–1594 (2017)

2. Baioletti, M., Milani, A., Santucci, V.: Algebraic crossover operators for permutations. In: 2018 IEEE Congress on Evolutionary Computation (CEC 2018). pp. 1–8 (2018). https://doi.org/10.1109/CEC.2018.8477867, doi:10.1109/CEC.2018.8477867
3. Baioletti, M., Milani, A., Santucci, V.: A new precedence-based ant colony optimization for permutation problems. In: Simulated Evolution and Learning. pp. 960–971. Springer International Publishing, Cham (2017). https://doi.org/https://doi.org/10.1007/978-3-319-68759-9_79
4. Baioletti, M., Milani, A., Santucci, V.: Automatic algebraic evolutionary algorithms. In: Proc. of Int. Workshop on Artificial Life and Evolutionary Computation (WIVACE 2017). pp. 271–283. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-78658-2_20
5. Baioletti, M., Milani, A., Santucci, V.: Learning bayesian networks with algebraic differential evolution. In: Proc. of 15th Int. Conf. on Parallel Problem Solving from Nature – PPSN XV. pp. 436–448. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-99259-4_35
6. Baioletti, M., Milani, A., Santucci, V.: MOEA/DEP: An algebraic decomposition-based evolutionary algorithm for the multiobjective permutation flowshop scheduling problem. In: Proc. of European Conference on Evolutionary Computation in Combinatorial Optimization - EvoCOP 2018. pp. 132–145. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-77449-7_9
7. Baioletti, M., Milani, A., Santucci, V.: Variable neighborhood algebraic differential evolution: An application to the linear ordering problem with cumulative costs. Information Sciences **507**, 37 – 52 (2020). https://doi.org/https://doi.org/10.1016/j.ins.2019.08.016, http://www.sciencedirect.com/science/article/pii/S0020025519307509
8. Baioletti, M., Milani, A., Santucci, V., Bartoccini, U.: An experimental comparison of algebraic differential evolution using different generating sets. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1527–1534. GECCO '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3319619.3326854, http://doi.acm.org/10.1145/3319619.3326854
9. Baioletti, M., Milani, A., Santucci, V., Tomassini, M.: Search moves in the local optima networks of permutation spaces: The qap case. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1535–1542. GECCO '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3319619.3326849, http://doi.acm.org/10.1145/3319619.3326849
10. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing **6**(2), 154–160 (1994)
11. Campos, V., Martí, R., Sánchez-Oro, J., Duarte, A.: Grasp with path relinking for the orienteering problem. Journal of the Operational Research Society **65**(12), 1800–1813 (2014). https://doi.org/10.1057/jors.2013.156
12. Chao, I.M., Golden, B.L., Wasil, E.A.: The team orienteering problem. European Journal of Operational Research **88**(3), 464 – 474 (1996). https://doi.org/https://doi.org/10.1016/0377-2217(94)00289-4, http://www.sciencedirect.com/science/article/pii/0377221794002894
13. Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Transportation Science **39**(2), 188–205 (2005)
14. Fischetti, M., González, J.J.S., Toth, P.: Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing **10**(2), 133–148 (1998). https://doi.org/10.1287/ijoc.10.2.133

15. Geem, Z.W., Tseng, C.L., Park, Y.: Harmony search for generalized orienteering problem: Best touring in china. In: Wang, L., Chen, K., Ong, Y.S. (eds.) Advances in Natural Computation. pp. 741–750. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
16. Gendreau, M., Laporte, G., Semet, F.: A branch-and-cut algorithm for the undirected selective traveling salesman problem. Networks **32**(4), 263–273
17. Golden, B.L., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics (NRL) **34**(3), 307–318 (1987)
18. Gunawan, A., Lau, H.C., Vansteenwegen, P.: Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research **255**(2), 315 – 332 (2016)
19. Hansen, N., Muller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation **11**(1), 1–18 (2003)
20. Hart, W.E., Krasnogor, N., Smith, J.E.: Memetic evolutionary algorithms. In: Recent advances in memetic algorithms, pp. 3–27. Springer (2005)
21. Helsgaun, K.: General k-opt submoves for the lin–kernighan tsp heuristic. Mathematical Programming Computation **1**(2), 119–163 (2009)
22. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of IEEE Intern. Conf. on Neural Networks. vol. 4, pp. 1942–1948 (1995)
23. Kobeaga, G., Merino, M., Lozano, J.A.: An efficient evolutionary algorithm for the orienteering problem. Computers Operations Research **90**, 42 – 59 (2018). https://doi.org/https://doi.org/10.1016/j.cor.2017.09.003, http://www.sciencedirect.com/science/article/pii/S0305054817302241
24. Labadie, N., Mansini, R., Melechovský, J., Calvo, R.W.: The team orienteering problem with time windows: An lp-based granular variable neighborhood search. European Journal of Operational Research **220**(1), 15 – 27 (2012). https://doi.org/https://doi.org/10.1016/j.ejor.2012.01.030, http://www.sciencedirect.com/science/article/pii/S0377221712000653
25. Laporte, G., Martello, S.: The selective travelling salesman problem. Discrete Applied Mathematics **26**(2), 193 – 207 (1990). https://doi.org/https://doi.org/10.1016/0166-218X(90)90100-Q, http://www.sciencedirect.com/science/article/pii/0166218X9090100Q
26. Leifer, A.C., Rosenwein, M.B.: Strong linear programming relaxations for the orienteering problem. European Journal of Operational Research **73**(3), 517 – 523 (1994). https://doi.org/https://doi.org/10.1016/0377-2217(94)90247-X, http://www.sciencedirect.com/science/article/pii/037722179490247X
27. Liang, Y.C., Kulturel-Konak, S., Lo, M.H.: A multiple-level variable neighborhood search approach to the orienteering problem. Journal of Industrial and Production Engineering **30**(4), 238–247 (2013). https://doi.org/10.1080/21681015.2013.818069
28. Marinakis, Y., Politis, M., Marinaki, M., Matsatsinis, N.: A memetic-grasp algorithm for the solution of the orienteering problem. In: Le Thi, H.A., Pham Dinh, T., Nguyen, N.T. (eds.) Modelling, Computation and Optimization in Information Systems and Management Sciences. pp. 105–116. Springer International Publishing, Cham (2015)
29. Milani, A., Santucci, V.: Asynchronous differential evolution. In: IEEE Congress on Evolutionary Computation. pp. 1–7 (July 2010). https://doi.org/10.1109/CEC.2010.5586107
30. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & operations research **24**(11), 1097–1100 (1997)

31. Moscato, P., Cotta, C.: A Gentle Introduction to Memetic Algorithms, pp. 105–144. Springer US, Boston, MA (2003)
32. Ramesh, R., Yoon, Y.S., Karwan, M.H.: An optimal algorithm for the orienteering tour problem. ORSA Journal on Computing **4**(2), 155–165 (1992). https://doi.org/10.1287/ijoc.4.2.155
33. Santucci, V., Baioletti, M., Milani, A.: Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. IEEE Transactions on Evolutionary Computation **20**(5), 682–694 (2016)
34. Santucci, V., Baioletti, M., Milani, A.: Tackling permutation-based optimization problems with an algebraic particle swarm optimization algorithm. Fundamenta Informaticae **167**(1-2), 133–158 (2019). https://doi.org/10.3233/FI-2019-1812, cited By 0
35. Santucci, V., Baioletti, M., Di Bari, G., Milani, A.: A binary algebraic differential evolution for the multidimensional two-way number partitioning problem. In: Evolutionary Computation in Combinatorial Optimization. pp. 17–32. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-16711-0_2
36. Sevkli, Z., Sevilgen, F.E.: Discrete particle swarm optimization for the orienteering problem. In: IEEE Congress on Evolutionary Computation. pp. 1–8 (July 2010). https://doi.org/10.1109/CEC.2010.5586532
37. Souffriau, W., Vansteenwegen, P., Berghe, G.V., Oudheusden, D.V.: A path relinking approach for the team orienteering problem. Computers Operations Research **37**(11), 1853 – 1859 (2010). https://doi.org/https://doi.org/10.1016/j.cor.2009.05.002, http://www.sciencedirect.com/science/article/pii/S0305054809001464, metaheuristics for Logistics and Vehicle Routing
38. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4), 341–359 (1997). https://doi.org/10.1023/A:1008202821328
39. Thomadsen, T., Stidsen, T.: The quadratic selective travelling salesman problem. informatics and mathematical modelling technical report 2003-17. Technical University of Denmark (2003)
40. Tsiligirides, T.: Heuristic methods applied to orienteering. Journal of the Operational Research Society **35**(9), 797–809 (Sep 1984). https://doi.org/10.1057/jors.1984.162, https://doi.org/10.1057/jors.1984.162
41. Vansteenwegen, P., Souffriau, W., Oudheusden, D.V.: The orienteering problem: A survey. European Journal of Operational Research **209**(1), 1 – 10 (2011)
42. Verbeeck, C., Sörensen, K., Aghezzaf, E.H., Vansteenwegen, P.: A fast solution method for the time-dependent orienteering problem. European Journal of Operational Research **236**(2), 419 – 432 (2014). https://doi.org/https://doi.org/10.1016/j.ejor.2013.11.038, http://www.sciencedirect.com/science/article/pii/S0377221713009557
43. İlhan, T., Iravani, S.M.R., Daskin, M.S.: The orienteering problem with stochastic profits. IIE Transactions **40**(4), 406–421 (2008). https://doi.org/10.1080/07408170701592481
44. Şevkli, A., Sevilgen, F.: Stpso: Strengthened particle swarm optimization. Turkish Journal of Electrical Engineering and Computer Sciences **18**(6), 1095–1114 (2010). https://doi.org/10.3906/elk-0909-18,