Knowledge-Based Systems 154 (2018) 93–101

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Multi-Objective Artificial Bee Colony algorithm applied to the bi-objective orienteering problem

Rodrigo Martín-Moreno, Miguel A. Vega-Rodríguez*

Department of Computer and Communications Technologies, University of Extremadura, Escuela Politécnica, Campus Universitario s/n, Cáceres 10003, Spain

ARTICLE INFO

Keywords:
Artificial Bee Colony
Multi-objective optimization
Bi-objective orienteering problem
Swarm intelligence
Evolutionary computation

ABSTRACT

We propose a new evolutionary computation approach for solving the multi-objective orienteering problem. This problem has applications in different fields like routing problems or logistic problems. In our case, the final motivation is the design of individual tourist routes. The tourists have different priorities about points of interests grouped into categories (for example, cultural or leisure), so, a multi-objective solution system is needed. In order to obtain the best Pareto solutions, the Artificial Bee Colony algorithm (based on swarm intelligence) has been adapted to the multi-objective context. The performance of this approach has been compared with two previous algorithms from the literature for the bi-objective orienteering problem (P-ACO and P-VNS), in benchmark instances and real-world instances. The results indicate that this new approach is good for solving the multi-objective orienteering problem.

1. Introduction

There is a huge variety of decision support systems that are used for different purposes in the companies such as marketing, finances, manufacturing, logistics or human resources, for example. However, it is very infrequent that companies provide systems to people which help them designing a tourist route that fits in their preferences, rather they offer different preplanned routes with low possibility of customization. When visitors are planning a stay in a tourist destination, they commonly want to visit some locations or Points Of Interests (POIs). These locations are from cathedrals, historical places or museums to restaurants, pubs or theaters. In a destination, there are a lot of POIs, chiefly if it has an important cultural heritage or a large variety for leisure. As it is impossible to visit all the places that destinations offer, the visitor must prioritize which POIs are worth to be visited using budget, time and interest, and decide what is their order in the route. Hence a decision support system that helps visitors in that process might be very interesting. Nevertheless the final decision will be made by the visitor, because this system will offer only the best solutions found that satisfy the user requirements, but will never consider the emotional part that biases in the process of planning a visit.

The fact of creating routes that connect POIs can be defined as an Orienteering Problem (OP) [1]. In this case, our focus is on the Multi-Objective Orienteering Problem (MOOP), where there are several categories for each point of interest (for example, cultural or leisure) and each of them has distinct benefits for every category. We have developed a Multi-Objective Artificial Bee Colony (MOABC) algorithm, based on the single-objective ABC algorithm proposed by Karaboga and Basturk [2], in order to solve MOOP in a competitive way. This is the first time that MOABC algorithm is applied to solve multi-objective orienteering problems and the conclusions obtained are very interesting. As we will see, MOABC results are very competitive when they are compared with the results obtained by other multi-objective algorithms (P-ACO and P-VNS) from the state-of-the-art in this field. The experiments have been made in both benchmark instances and real-world instances (a total of 216 instances grouped in 10 sets), and we have used three state-of-art performance multi-objective indicators to report and compare the results. Moreover, the outperformance of MOABC has been confirmed after a statistical analysis.

This paper is structured as follows. Section 2 introduces the reader to the work made in this field. Section 3 presents the formal problem definition and its mathematical formulation. Section 4 describes our multi-objective approach to solve this problem. Section 5 discusses about the results obtained and compares the quality metrics of our algorithm with respect to other previously published algorithms, including a statistical analysis. Section 6 concludes this paper and summarizes possible future works.

2. Related work

Multi-objective optimization is an important field, with a lot of activity in the past two decades. Within multi-objective optimization,

* Corresponding author.
E-mail addresses: rmartinky@alumnos.unex.es (R. Martín-Moreno), mavega@unex.es (M.A. Vega-Rodríguez).

https://doi.org/10.1016/j.knosys.2018.05.005
Received 14 November 2017; Received in revised form 2 May 2018; Accepted 5 May 2018
Available online 07 May 2018
0950-7051/ © 2018 Elsevier B.V. All rights reserved.

the solving of MOOPs is an area of high applicability. Regrettably, there is not much research work in this area.

Mainly, there are three different approaches with multiple objective functions. The first of them implies building a function that combines the objectives using some kind of weights or thresholds, but the problem is that these have to be previously defined. The result obtained with this approach is a solution that fits well with the weighted relation among objectives given by the fitness function. A second approach is based on switching among the objectives (criteria) during the selection phase. Every time a selection is made, potentially a different objective function (criterion) will be used. The main problem of the objective (criterion) switching is that the solution population tends to converge to solutions which are superior in one objective, but poor at others. Finally, the third alternative is based on including the objectives isolated from each other and calculating the best (that is, Pareto-optimal or non-dominated) candidate solutions in order to provide the decision-maker the possibility of exploring all the best trade-off solutions. Here, we use this third alternative.

In the literature we can find extensions of well-known metaheuristics to the multi-objective context: genetic algorithms [3], evolutionary algorithms (see [4]), and some swarm algorithms [5], for example. A complete survey on multi-objective evolutionary algorithms can be found in [6]. In our case, the ABC algorithm has been adapted to the multi-objective context. We have based our work on the ABC algorithm because it has been widely and successfully studied and applied to solve real-world problems in multiple fields [7], including single-objective optimization (e.g. [8–10]) and multi-objective optimization (e.g. [11–13]).

The orienteering problem was defined by Tsiligirides [1], and some authors have considered it as a kind of TSP (Traveling Salesman Problem) with profits (see [14]) or selective TSP. In OP, every vertex has associated some benefit, and the goal is to visit a group of vertices that maximize the sum of benefits, while fulfilling the corresponding tour length/cost constraint. Other related problems are the vehicle routing problem with profits [15] and the team orienteering problem (see [16,17]), where the problem is extended to multiple tours. Two complete surveys on orienteering problem can be found in [18,19].

Regarding the multi-objective orienteering problem, the problem addressed in this paper, very few proposals can be found in the literature. Within bi-objective orienteering problem we can highlight [20] that used a P-ACO (Pareto Ant Colony Optimization) algorithm and a P-VNS (Pareto Variable Neighborhood Search) algorithm combined with path relinking and [21] that used an evolutionary algorithm also combined with path relinking.Martí et al. [22] used GRASP combined with path relinking.

3. Problem definition

The multi-objective orienteering problem can be specified based on a directed graph $G = (V, A)$. This directed graph has a set of vertices, $V = \{v_0, v_1, v_2, \ldots, v_{n+1}\}$, and a set of arcs, $A = \{(v_i, v_j): v_i, v_j \in V \wedge v_i \neq v_j \wedge v_i \neq v_{n+1} \wedge v_j \neq v_0\}$. Every vertex $v_i \in V \setminus \{v_0, v_{n+1}\}$ has associated K benefits b_{ik} $(k = 1, \ldots, K)$. The starting and ending vertices, v_0 and v_{n+1}, do not have benefits associated. Furthermore, each arc $(v_i, v_j) \in A$ has a cost c_{ij} that can be interpreted as distance, money or time spent for going from v_i to v_j.

In all the instances used in this work, v_0 and v_{n+1} represent the same point. For this reason, we call a solution as "tour" instead of "path". The goal of the multi-objective orienteering problem is to find the best tours (which maximize the benefits in all the objectives) from v_0 to v_{n+1}, while fulfilling the tour length/cost constraint $C_{\max}$.

Therefore, we can mathematically define the problem as:

$$\textit{maximize} \quad F(x) = (f_1(x), \ldots, f_K(x)), \tag{1}$$

$$f_k = \sum_{v_i \in V \setminus \{v_0, v_{n+1}\}} (b_{ik} \cdot y_i) \quad (k = 1, \ldots, K), \tag{2}$$

where y_i, a binary variable, has a value equal to 1 when v_i is visited, and else a value of 0. Furthermore, we have to take into account that:

$$\sum_{v_i \in V \setminus \{v_j\}} a_{ij} = y_j \quad (v_j \in V \setminus \{v_0\}), \tag{3}$$

$$\sum_{v_j \in V \setminus \{v_i\}} a_{ij} = y_i \quad (v_i \in V \setminus \{v_n + 1\}), \tag{4}$$

$$\sum_{\{v_i, v_j\} \in S} a_{ij} \leq |S| - 1 \quad (S \subseteq V \wedge S \neq \emptyset), \tag{5}$$

$$y_0 = y_{n+1} = 1, \tag{6}$$

$$\sum_{(v_i, v_j) \in A} c_{ij} \cdot a_{ij} \leq C_{max}, \tag{7}$$

$$a_{ij} \in \{0, 1\} \quad \left((v_i, v_j) \in A\right), \tag{8}$$

$$y_i \in \{0, 1\} \quad (v_i \in V). \tag{9}$$

The binary variable a_{ij} has a value equal to 1 when $(v_i, v_j) \in A$ is used, and else a value of 0. Eq. (1) indicates that for solving MOOP we have to maximize the different objective functions. Eq. (2) defines each objective function as the addition of the corresponding benefits. Eq. (3),4 imply that for each vertex visited only one arc is ingoing and only one arc is outgoing. Eq. (5) avoids subtours. Eq. (6) implies that the starting and ending points are used in all the tours. Eq. (7) guarantees that the tour cost is not greater than the established limit $C_{\max}$. In our case, we will solve the bi-objective orienteering problem, that is, $K = 2$.

4. Solution procedure

Golden et al. [23] demonstrated that OP is NP-hard; no polynomial time algorithm has been designed, or is expected to be designed, to solve this problem to optimality, and especially when multiple objectives exist.

Therefore, we need to apply a metaheuristic solution technique to the multi-objective orienteering problem. In this section, we describe the MOABC algorithm that we have designed and developed. The single-objective Artificial Bee Colony (ABC) algorithm was originally proposed by Karaboga and Basturk [2], and we adapt it to the multi-objective context and the particular solving of MOOP.

We represent every solution to MOOP as a list of points (those included in the corresponding tour), which is the most natural way for representing a solution for this problem.

4.1. Multi-objective optimization

Due to the multi-objective nature of the problem to solve, it is very difficult to choose exactly an optimal solution where all the objectives are maximized. Nevertheless, if we restrict to non-dominated solutions the choice will be bounded to a reasonable amount of candidate solutions. The next definitions help to clear this aspect.

A solution x dominates a solution x' if x is not worse than x' in any of the objective functions, and is better at least in one of the objective functions. Formally: for $F(x) = (f_1(x), \ldots, f_K(x))$ to be maximized, x dominates x' if $f_k(x) \geq f_k(x')$ for all $k = 1, \ldots, K$, and $f_k(x) > f_k(x')$ for at least one k. If this happens, we write $x \succ x'$.

If no solution dominates the solution x^*, we say that x^* is non-dominated or Pareto-efficient. In this case, we say that $z^* = F(x^*) = (f_1(x^*), \ldots, f_K(x^*))$ is a non-dominated vector. The set of all non-dominated vectors is called non-dominated frontier or Pareto front. The relation $\succ$ can be extended from the solution space to the objective space. In that case, given two vectors $z = (z_1, \ldots, z_K)$ and $z' = (z'_1, \ldots, z'_K)$,

```
input  : $T_{\max}$ (maximum execution time), $PS$ (population size), and $limit$ (abandonment criterion)
output: $NDS\_File$ (file with the non-dominated solutions)
$t \leftarrow 0$; $NDS\_File \leftarrow \emptyset$;
// InitializeFoodSources($PS$), see subsection 4.2.1
for $i \leftarrow 1$ to $PS$ do
    $EBee[i] \leftarrow$ generateRandomSolution();
end
while $t < T_{\max}$ do
    // EmployedBeesPhase($PS$), see subsection 4.2.2
    for $i \leftarrow 1$ to $PS$ do
        $neighborEBee \leftarrow$ generateNeighborEBee($EBee[i]$);
        if $neighborEBee \succ EBee[i]$ then
            $EBee[i] \leftarrow neighborEBee$;
        end
    end
    // ComputeNectarAmounts($PS$), see subsection 4.2.3
    for $i \leftarrow 1$ to $PS$ do
        MOfitness($EBee[i]$);
    end
    // OnlookerBeesPhase($PS$), see subsection 4.2.4
    for $i \leftarrow 1$ to $PS$ do
        $OBee[i] \leftarrow$ ProbabilisticSelectionEBee();
        $neighborOBee \leftarrow$ generateNeighborOBee($OBee[i]$);
        if $neighborOBee \succeq OBee[i]$ then
            $OBee[i] \leftarrow neighborOBee$;
        end
    end
    // ScoutBeesPhase($limit, PS$), see subsection 4.2.5
    for $i \leftarrow 1$ to $PS$ do
        checkAbandon($EBee[i], limit$); checkAbandon($OBee[i], limit$);
    end
    // See subsection 4.2.6
    SaveNon-dominatedSolutions($NDS\_File$);
end
```

Algorithm 1. Multi-Objective Artificial Bee Colony algorithm.

we write $z \succ z'$ if $z_k \geq z'_k$ for all $k = 1, \ldots, K$ and $z_k > z'_k$ for at least one k.

4.2. MOABC

The ABC algorithm is a metaheuristic algorithm, based on swarm intelligence, that was proposed by Karaboga and Basturk [2]. The algorithm is based on the foraging behavior of honey bee colonies and consists of three main components: employed bees, onlooker bees, and scout bees.

The first component, employed bees, search for food sources (solutions). There are also two other interesting behaviors that are required for self-organizing and swarm intelligence: recruitment of new foraging bees (onlooker bees) to the good food sources (positive feedback) and abandonment of exhausted food sources (negative feedback), generating scout bees.

We have adapted the original ABC algorithm to the multi-objective context and the particular solving of MOOP. A pseudocode of our MOABC algorithm is shown in Algorithm 1. The following subsections explain all the parts of this algorithm. In our case, fitness evaluations are performed by using the multi-objective operators rank and crowding [3]. As we will see, these multi-objective operators are used in "ComputeNectarAmounts(*PS*)" (line 13) for assigning higher selection probabilities to solutions with better multi-objective fitness, and also in "SaveNon-dominatedSolutions(*NDSFile*)" (line 25) for keeping the solutions with better fitness as the employed bees for the next iteration of the algorithm. A more detailed explanation is given in the corresponding Sections 4.2.3 and 4.2.6, respectively.

4.2.1. Initialize food sources

All the solutions of the population of food sources (*PS*: population size) are randomly initialized by employed bees. The random initialization of every solution follows these steps. First, we compute the 20 best movements from every point (vertex) to any other point. This calculation is only computed one time, and then used when is necessary. The best movements are calculated by using Eq. (10). If we are at a point i, for going to point j the $ratio_{ij}$ is calculated, where b_{j1} and b_{j2} are the benefits of point j and c_{ij} is the cost for going from point i to point j. Therefore, Eq. (10) relates benefits with cost. Note that the values b_{j1}, b_{j2}, and c_{ij} are normalized in order to avoid any bias. The movements with higher ratios are considered better movements.

$$ratio_{ij} = \frac{(b_{j1} + b_{j2})}{c_{ij}}. \tag{10}$$

After that, beginning at the starting point (vertex), we randomly select one movement among the 20 best movements. This procedure is repeated until more points cannot be added due to the cost limit $C_{\max}$. Observe that all the tours finish at the ending point and the rest of constraints are also fulfilled during the random generation of a solution.

4.2.2. Employed bees phase

Employed bees search for new solutions, within their neighborhoods, that are better than their current solutions. That is, they find a neighbor solution and then evaluate its quality.

In our case, in order to obtain a neighbor solution for an employed bee we use two operators: shortening and vertex insertion. Shortening tries to rearrange the vertices, within a tour, for minimizing the tour cost. We use the shortening operator 2-opt proposed by Croes [24]. For runtime reasons, the sequence length of a tour segment to analyze by the operator 2-opt is limited to 30. This same value is used in [20], whose algorithms we will use in our comparisons. The main idea behind the operator 2-opt is to take a tour that crosses over itself and reorder it (that is, visit its points in a different order) so that it does not cross over itself. Vertex insertion checks the feasibility to insert additional points after a shortening. For every possible insertion position, the best ten points (according to Eq. (10)) from non-visited points are used as set of candidate points to insert, and one of them is randomly selected. Every possible insertion position that lies within the cost limit $C_{\max}$ is checked.

After producing the neighbor solution, it is compared with the current solution. If the neighbor solution dominates the current solution, then it replaces the current solution for this employed bee.

4.2.3. Compute Nectar Amounts

Two multi-objective optimization operators are used to determine which are the best solutions associated to employed bees: rank and crowding. The first operator ranks the solutions in different Pareto fronts according to their dominance relations, whereas the second one takes into account the solutions' crowding distance, which prefers differentiating solutions, i.e., more diverse solutions. A more detailed explanation of these two multi-objective optimization operators can be found in [3]. Once these two operators are applied, for every solution x (employed bee) we combine the values from these two operators in one single value (called multi-objective fitness, MOfitness) thanks to Eq. (11). MOfitness is an important value because higher multi-objective fitness value will imply higher selection probability, and the subsequent selection (see next subsection) is based on those probabilities.

$$MOfitness(x) = \frac{1}{2^{rank(x)} + \frac{1}{1 + crowding(x)}}. \tag{11}$$

4.2.4. Onlooker bees phase

Employed bees share their solution fitness information with onlooker bees and onlooker bees select their solutions, probabilistically, based on the probability values computed by using the MOfitness values (see Eq. (11)). A fitness-based selection method is applied, more specifically the roulette wheel selection proposed by Goldberg [25]. Therefore, better solutions will recruit more onlooker bees (positive feedback).

The probability $p(x^i)$ with which a solution x^i is selected by an onlooker bee is computed by applying Eq. (12).

$$p(x^i) = \frac{MOfitness(x^i)}{\sum_{m=1}^{PS} MOfitness(x^m)}. \tag{12}$$

After a solution x^i is probabilistically selected by an onlooker bee, a neighborhood solution is generated by using two operators: vertex exchange and shortening. Vertex exchange tries to eliminate every existing point in the tour and to exchange it with a non-visited point. In every position, the non-visited point is selected among the best ten non-visited points (according to Eq. (10)) for this position. After the vertex exchange operator has been applied, the shortening operator is used in order to improve the solution. This shortening operator is the same as the one explained for the employed bees.

After producing the neighbor solution, it is compared with the originally selected solution. If the neighbor solution is not dominated by the original one, then it replaces the original solution for this onlooker bee.

4.2.5. Scout bees phase

Employed and onlooker bees whose solutions cannot be enhanced after an established number of attempts, indicated by a configuration parameter of the MOABC algorithm and denominated *limit* (or abandonment criterion), become scout bees and their solutions are abandoned. Note that MOABC is using more scout bees than in the original ABC, where at most one scout bee is used per iteration [26]. In MOABC all employed and onlooker bees with attempts > = *limit* become scout bees and as such MOABC consumes more fitness evaluations than the original ABC. Anyway, the subsequent comparisons with P-ACO and P-VNS (Section 5.3) are fair due to the termination condition is based on equal execution time. The scout bees randomly search for new solutions (in this way, improving the exploration of the algorithm). Therefore, those solutions that cannot be improved by exploitation are abandoned (negative feedback). The new solution associated with a scout bee is randomly generated in the same way as in Section 4.2.1 (Initialize Food Sources). After that, the scout bee is improved by using the three previously explained operators: vertex insertion, exchange, and shortening.

4.2.6. *Save non-dominated solutions*

On each iteration, the non-dominated solutions are computed and saved into a file (*NDSFile*) in order to preserve them while new solutions are generated. The operator used to compute the non-dominated solutions is the rank (non-dominated sorting) operator, which classify the solutions in different Pareto fronts according to their dominance relations. Furthermore, the crowding operator is also applied. In this way, both operators are used in order to sort all the current solutions (employed, onlooker, and scout bees). A more detailed explanation of these two operators (rank and crowding) can be found in [3]. After that, the best *PS* (population size) solutions are held as the employed bees for the next iteration, and as said, the non-dominated solutions (the solutions with $rank = 0$) are saved into a file (*NDSFile*).

5. Results

Our MOABC algorithm is implemented in C++, with the GNU compiler g++ 4.8.4 on a 64-bit Linux operating system (Ubuntu server 14.04 LTS). All the experiment runs are executed on an Intel Pentium 4 D CPU with 3.2 GHz and 1 GB of RAM in order to get a very similar computational environment to the one that was used for testing P-ACO and P-VNS algorithms.

5.1. *Benchmark and real-world instances*

We have used the same instances that were used by Schilde et al. [20] and that are available at http://prolog.univie.ac.at/research/OP. The instance sets could be categorized in 10 groups, according to the number of vertices. In each group there are several instances with a different maximum tour length constraint ($C_{\max}$). The first 3 groups (2_p21, 2_p32, and 2_p33) with 21, 32, and 33 vertices and 11, 18, and 20 instances respectively were created by Tsiligirides [1]. The 2 next groups (2_p64 and 2_p66) with 64 and 66 vertices and 14 and 26 instances were proposed by Chao et al. [27]. These 5 groups were originally created by the authors with a single objective, so Schilde et al. [20] added a second objective. The distance between vertices is computed by using the Euclidean distance. These 5 groups can be considered as benchmark instances.

The next 5 groups were created totally by Schilde et al. [20] and are real-world instances using some European cities like Padua or Vienna, and Austria regions. Within these 5 groups we have 97 and 273 vertices (2_p97 and 2_p273) with 20 instances each one and with a tour length constraint ($C_{\max}$) between 1 and 20 km; and 559, 562 and 2143 vertices (2_p559, 2_p562, and 2_p2143) with 29 instances and a $C_{\max}$ from 10 to 150 km. All of them are provided with a distance matrix between vertices. In addition, for these 5 instance groups it is used a service distance of 0.5 km that can be taken as the time needed to visit a POI.

5.2. *Evaluation metrics*

In order to compute the metrics, we have used the software available at http://www.tik.ee.ethz.ch/sop/pisa/?page=assessment.php, according to the instructions provided by Knowles et al. [28].

All the indicators (metrics) used in this work have been computed using normalized values of the objective functions. As we want to compare with the results from the algorithms P-ACO and P-VNS [20], we have used the same number of independent runs as these algorithms used, i.e. 10 per instance. For some unary indicators a reference set is necessary, so we have generated a reference set made with the non-dominated solutions of the 10 executions of the algorithms to compare (MOABC, P-ACO, and P-VNS). We have used the solutions provided by Schilde et al. [20] for each instance computed by P-ACO and P-VNS, and we have established a comparison among the three algorithms according to several indicators (metrics).

More specifically, the following three indicators have been used: hypervolume, epsilon, and R3. First of all, these three indicators are Pareto-compliant, which is a desirable feature. Hypervolume indicator has been used due to its prevalence in the literature. Probably, it is the most well-known and used indicator in multi-objective optimization. It is strictly Pareto-compliant, being the only known indicator that satisfies the strict Pareto compliance property. Furthermore, it is very complete because it assesses convergence, uniformity, and spread. It does not require a reference set, it only requires a reference point. Epsilon indicator is also very well-known and used in the literature. It has different properties than the hypervolume because it requires a reference set and is weakly Pareto-compliant. Epsilon indicator assesses the convergence. Finally, R3 indicator is well-known and used in the literature. It is weakly Pareto-compliant, but in contrast with the two previous indicators, the R3 indicator requires both a reference set and a reference point. R3 indicator assesses uniformity and spread. As we can see, each indicator is based on different properties; therefore, using them all provides a range of comparisons rather than just one, which gives a more complete and balanced view of the assessment, increasing its reliability.

5.2.1. *Hypervolume indicator*

Hypervolume measures the area (volume or hypervolume) of the objective space that is weakly dominated by a solutions set A (the non-dominated frontier of an approximation set). A more detailed description can be found in [29]. In the case of two objectives, this indicator is calculated by using Eq. (13).

$$I_H(A) = f_1(x^1)\cdot f_2(x^1) + \sum_{x^i,x^j\in A:i+1=j,i\geq 1} \{[f_2(x^j) - f_2(x^i)]\cdot f_1(x^j)\}, \tag{13}$$

where $f_k(x^i)$ is the objective function value for objective k returned by solution x^i. We have normalized all the objective values to the range [1.0,2.0], with 1.0 being the worst result and 2.0 the best one. We have used the point (0,0) as nadir. Hence, the hypervolume values are in the interval [1,4], where values closer to 4 imply better solutions.

5.2.2. *Unary epsilon indicator*

We use the multiplicative version of this indicator. The unary epsilon indicator computes the minimal (*inf*) factor ϵ in such a way that when every point in the approximation set A is multiplied by ϵ, then the resulting set weakly dominates the reference set R. A more detailed explanation can be found in [30]. The ϵ indicator is computed by using Eq. (14).

$$I_\epsilon(A, R) = \inf_{\epsilon\in\Re}\{\ \forall\, x^2 \in R \quad \exists\, x^1 \in A: x^1 \succeq_\epsilon x^2\}. \tag{14}$$

For a maximization problem, a solution x^1 ϵ-dominates a solution x^2, written as $x^1 \succeq_\epsilon x^2$, if $f_k(x^1)\cdot\epsilon \geq f_k(x^2)$ is satisfied for all the objective values k. We have normalized all the objective values to the range [1.0,2.0], with 1.0 being the worst result and 2.0 the best one. When this indicator is equal to 1.0, all the solutions of the reference set R were found. In conclusion, epsilon indicator values closer to 1.0 imply better solutions.

5.2.3. *R3 indicator*

This indicator is applied to compare the approximation sets found by the three algorithms (MOABC, P-ACO, and P-VNS), taking into account a set of utility functions. Hence, a set Λ of 500 uniformly distributed weight vectors $\vec{\lambda} = \{\lambda_1, ..., \lambda_K\} \in \Lambda$ is generated and applied for computing the utilities for the corresponding approximation sets using an augmented Tchebycheff function (Eq. (15)). A detailed description of the R3 indicator can be found in [31]. We have normalized again all the objective values to the range [1.0,2.0], with 1.0 being the worst result and 2.0 the best one. Remember that $z = F(x) = (f_1(x),...,f_K(x)) = (z_1,...,z_K)$ (see Section 4.1). Knowing that

$$u\left(\vec{\lambda}, x\right) = -\left(\max_{j\in 1...K} \lambda_j \cdot |z_j^* - z_j| + \rho\cdot \sum_{j\in 1...K} |z_j^* - z_j|\right) \tag{15}$$

is the utility reached by a solution x with the weight vector $\vec{\lambda}$ and

$$u^*\left(\vec{\lambda}, A\right) = \max_{x\in A} u\left(\vec{\lambda}, x\right) \tag{16}$$

is the maximal utility reached with the weight vector $\vec{\lambda}$ on an approximation set A, the R3 indicator can be computed as is indicated in Eq. (17).

$$I_{R3}(A, R) = \frac{\sum_{\vec{\lambda}\in\Lambda} \frac{u^*\left(\vec{\lambda}, A\right) - u^*\left(\vec{\lambda}, R\right)}{u^*\left(\vec{\lambda}, R\right)}}{|\Lambda|}. \tag{17}$$

We have set ρ to a value equal to 0.01 and z^* to a value equal to 2.0 in every objective (a point that weakly dominates all the points in the approximation set). When the R3 indicator has a value close to 0, the approximation set A generates utility values that are close to the utility values generated by the reference set R. In conclusion, R3 indicator values closer to 0 imply better solutions.

5.3. Comparing MOABC vs. P-ACO and P-VNS

In this subsection, we compare the results obtained by MOABC versus the results from other two multi-objective optimization algorithms. As we want to compare with the results from the algorithms P-ACO and P-VNS [20], we have used the same number of independent runs as these algorithms used, i.e. 10 per instance. We have used the solutions provided by Schilde et al. [20] for each instance computed by P-ACO and P-VNS, which used their best parameter settings. In that reference the configuration parameters for P-ACO and P-VNS are also detailed. In the case of our algorithm (MOABC), following the recommendation of Veček et al. [32] we have studied not only the *PS* (population size) parameter, but also the *limit* parameter. After performing preliminary tests with a range of different values for each parameter (*PS* and *limit*), we have set these two MOABC configuration parameters to a population size (*PS*) of 60 and an abandonment criterion (*limit*) of 10. All the algorithms were stopped after using the same runtime, in particular, the same maximum execution time as was used by the P-ACO algorithm. These execution times are available at http://prolog.univie.ac.at/research/OP along with the corresponding instance sets, and the results of P-ACO and P-VNS for all these instance sets. In the case of MOABC, in order to achieve the same termination condition (that is, the same maximum execution time) as P-ACO and P-VNS we have performed the following procedure: in the C++ implementation of our algorithm (MOABC) we have used the function *alarm*() (*ualarm*()) for causing the system to generate a *SIGALRM* signal for the process after the number of seconds (microseconds) specified. Previously to this, we have used the functions *sigemptyset*() (to initialize the signal set to empty), *sigaddset*() (to add the *SIGALRM* signal), and *signal*() to correctly define a handler for this signal and precisely stop our algorithm.

In first place, we present the comparison based on the hypervolume indicator (I_H). Table 1 shows the results for the three algorithms and for every instance set. Remember that every instance set includes a determined number of instances (see Section 5.1) and we have repeated every experiment 10 times.

Table 1
Hypervolume indicator results. The numbers in bold mean the superiority of that algorithm. Larger values are better.

Sets	$I_H(MOABC)$	$I_H(P-ACO)$	$I_H(P-VNS)$
2_p21	**3.525**	3.390	3.471
2_p32	3.366	**3.375**	3.250
2_p33	3.575	**3.638**	3.615
Average (Small)	**3.489**	3.468	3.445
2_p64	3.363	**3.586**	3.569
2_p66	3.346	**3.609**	3.554
2_p97	**3.501**	3.315	3.335
Average (Medium)	3.403	**3.503**	3.486
2_p273	**3.472**	3.329	3.320
2_p559	**3.648**	3.415	3.441
2_p562	**3.634**	3.451	3.370
2_p2143	**3.640**	3.144	2.839
Average (Large)	**3.598**	3.335	3.242
Average (All)	**3.497**	3.435	3.391

Table 1 presents the mean results per instance set. For example, instance set 2_p21 has 11 instances, therefore, for this instance set, this table shows the mean results of 110 executions (remember that most of the instance sets include at least 20 instances, implying mean results of at least 200 executions). Furthermore, we have grouped the instance sets in three groups taking into account their sizes (number of vertices): small, medium, and large. Therefore, we also present the average results for each of these three groups. Finally, we also show the average taking into account all the instance sets.

In small instances, MOABC gets the best average hypervolume, and also provides better results on instance set 2_p21 than P-ACO and P-VNS. In medium instances, MOABC algorithm obtains the best hypervolume for the instance set 2_p97. In large instances, MOABC algorithm is definitely better than P-ACO and P-VNS. It obtains the best hypervolumes for all the large instances (that is, the most difficult ones). In fact, it is here where the differences among the three algorithms are bigger, especially in the last instance set with 2143 vertices (2_p2143). As conclusion, in Table 1 we can see that MOABC has the best overall average hypervolume.

In second place, and using the same methodology as for the hypervolume indicator, Table 2 shows the results for the three algorithms and their comparison based on the unary epsilon indicator (I_ϵ).

In small instances, MOABC provides the best average epsilon indicator, and also obtains better results on instance sets 2_p21 and 2_p32 than P-ACO and P-VNS. In medium instances, MOABC algorithm again provides the best average epsilon indicator, and also gets the best results for the instance set 2_p97. In large instances, MOABC algorithm is clearly better than P-ACO and P-VNS. It obtains the best results for all

Table 2
Unary epsilon indicator results. The numbers in bold mean the superiority of that algorithm. Lower values are better.

Sets	$I_\epsilon(MOABC)$	$I_\epsilon(P-ACO)$	$I_\epsilon(P-VNS)$
2_p21	**1.000**	1.108	1.034
2_p32	**1.024**	1.028	1.124
2_p33	1.074	**1.032**	1.050
Average (Small)	**1.032**	1.056	1.069
2_p64	1.085	**1.036**	1.038
2_p66	1.104	**1.033**	1.049
2_p97	**1.000**	1.158	1.137
Average (Medium)	**1.063**	1.076	1.074
2_p273	**1.072**	1.196	1.161
2_p559	**1.037**	1.118	1.113
2_p562	**1.039**	1.112	1.128
2_p2143	**1.076**	1.188	1.265
Average (Large)	**1.056**	1.153	1.167
Average (All)	**1.050**	1.095	1.103

Table 3
R3 indicator results. The numbers in bold mean the superiority of that algorithm. Lower values are better.

Sets	$I_{R3}(MOABC)$	$I_{R3}(P-ACO)$	$I_{R3}(P-VNS)$
2_p21	**0.000**	0.031	0.010
2_p32	**0.003**	**0.003**	0.030
2_p33	0.018	**0.005**	0.010
Average (Small)	**0.007**	0.013	0.017
2_p64	0.032	0.011	**0.007**
2_p66	0.039	**0.008**	0.011
2_p97	**0.000**	0.046	0.041
Average (Medium)	0.024	0.022	**0.020**
2_p273	**0.030**	0.067	0.063
2_p559	**0.011**	0.054	0.048
2_p562	**0.010**	0.041	0.059
2_p2143	**0.038**	0.135	0.204
Average (Large)	**0.022**	0.074	0.093
Average (All)	**0.018**	0.036	0.043

the large instances (that is, the most complex ones). Analyzing Table 2 we can observe that MOABC has the best overall average epsilon indicator, providing the best results for 7 out of 10 instance sets.

In third place, and using the same methodology as in the two previous tables, Table 3 presents the results for the three algorithms and their comparison based on the R3 indicator (I_{R3}).

In small instances, MOABC gets the best average R3 indicator, and also provides the best results on instance sets 2_p21 and 2_p32. In medium instances, MOABC algorithm gets the best results for the instance set 2_p97. In large instances, MOABC algorithm is definitely better than P-ACO and P-VNS. As in the two previous indicators, MOABC obtains the best results for all the large instances. Taking into account Table 3, we can conclude that MOABC has the best overall average R3 indicator, providing the best results for 7 out of 10 instance sets.

Summarizing the results from the three previous tables we conclude that:

- MOABC has the best overall average for the three quality indicators (I_H, I_ϵ, and I_{R3}).
- In small instances, MOABC gets the best average for the three indicators, and generally, provides better results than P-ACO and P-VNS on 2 out of 3 instance sets (2_p21 and 2_32).
- In medium instances, according to all the indicators, MOABC clearly outperforms the other two algorithms (P-ACO and P-VNS) on the instance set 2_p97 (one of the three instance sets within this group).
- In large instances, MOABC is the best algorithm, taking into account the three quality indicators, their corresponding averages, and also the individual results for each of the 4 instance sets of this group. The differences among the three algorithms are specially bigger in the last instance set with 2143 vertices (the most difficult one). Precisely for this instance set, in Fig. 1 we can observe a plot corresponding with the Pareto front approximation, with 50% attainment and a constraint ($C_{\max}$) of 150 km, for every compared algorithm (MOABC, P-ACO, and P-VNS). We can observe how solutions provided by MOABC clearly dominate the solutions of the other algorithms. Furthermore, the Pareto front approximation of MOABC has a higher number of points (non-dominated solutions found).

5.4. Statistical analysis of the results

In order to know if the differences (the improvements of MOABC with respect to the other algorithms) shown in the results of the previous subsection are statistically significant, we have performed the following statistical analysis for the results of the three indicators (hypervolume, epsilon, and R3). Firstly, a test for calculating residual normality is applied, in our case, the Kolmogorov–Smirnov and

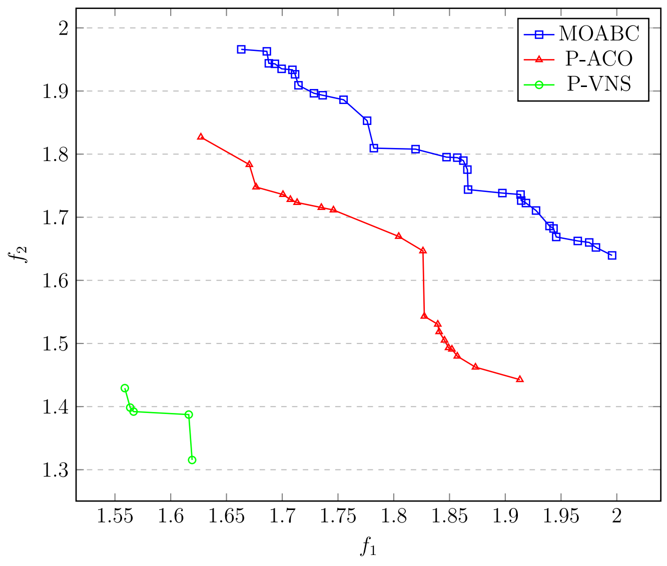

Fig. 1. Pareto front approximations (50% attainment) of each algorithm for the instance set 2_p2143 with $C_{\max} = 150$.

Shapiro–Wilk tests are applied. The main objective of these tests is to check whether the results follow a Gaussian distribution or not. If the values follow a Gaussian distribution, then the Levene test to check the homogeneity of the variances (or homoscedasticity) is also carried out. After that, if all these tests are positive (Kolmogorov–Smirnov, Shapiro–Wilk, and Levene) we can apply an ANOVA parametric analysis to verify if the differences in the results are statistically significant. If some of the tests (Kolmogorov–Smirnov, Shapiro–Wilk, or Levene) is negative, we have to apply a non-parametric analysis. In particular, we apply a Wilcoxon–Mann–Whitney test to verify if the differences in the results are statistically significant. The confidence level considered in this work is always 95% in all the statistical tests (a significance level of 5% or p-value under 0.05). This means that the differences are unlikely to have occurred by chance with a probability of 95%. A more detailed explanation about all these statistical procedures can be found in [33].

In first place, we perform the statistical analysis for the results of the hypervolume indicator. After applying the Kolmogorov–Smirnov and Shapiro–Wilk tests we detect that the results do not follow a Gaussian distribution; therefore, we have to perform a non-parametric analysis. We apply the Wilcoxon–Mann–Whitney test by pairs of algorithms. Table 4 shows the p-values obtained from this statistical test, highlighting the cases in which the differences are statistically significant. As we can see, the differences between MOABC and P-ACO are significant in most of the cases, except for 2_p32, 2_p33, and 2_p273. These results confirm our conclusions of the previous subsection. MOABC obtains better results than P-ACO, even in two of the cases where

Table 4
Statistical analysis for the hypervolume indicator results. Statistically significant differences are highlighted in bold.

Sets	Statistically significant differences (Yes/No) and exact p-value		
	MOABC vs. P-ACO	MOABC vs. P-VNS	P-ACO vs. P-VNS
2_p21	**Yes (0.041)**	No (0.280)	No (0.332)
2_p32	No (0.898)	**Yes (0.032)**	**Yes (0.025)**
2_p33	No (0.088)	No (0.160)	No (0.557)
2_p64	**Yes (0.000)**	**Yes (0.000)**	No (0.061)
2_p66	**Yes (0.000)**	**Yes (0.000)**	**Yes (0.000)**
2_p97	**Yes (0.002)**	**Yes (0.006)**	No (0.780)
2_p273	No (0.259)	**Yes (0.000)**	**Yes (0.002)**
2_p559	**Yes (0.000)**	**Yes (0.000)**	No (0.463)
2_p562	**Yes (0.000)**	**Yes (0.000)**	No (0.053)
2_p2143	**Yes (0.000)**	**Yes (0.000)**	**Yes (0.000)**

Table 5
Statistical analysis for the unary epsilon indicator results. Statistically significant differences are highlighted in bold.

Sets	Statistically significant differences (Yes/No) and exact p-value		
	MOABC vs. P-ACO	MOABC vs. P-VNS	P-ACO vs. P-VNS
2_p21	**Yes (0.000)**	**Yes (0.000)**	**Yes (0.010)**
2_p32	No (0.549)	**Yes (0.000)**	**Yes (0.000)**
2_p33	**Yes (0.000)**	**Yes (0.040)**	**Yes (0.023)**
2_p64	**Yes (0.000)**	**Yes (0.000)**	No (0.565)
2_p66	**Yes (0.000)**	**Yes (0.000)**	**Yes (0.000)**
2_p97	**Yes (0.000)**	**Yes (0.000)**	No (0.261)
2_p273	**Yes (0.000)**	**Yes (0.000)**	No (0.296)
2_p559	**Yes (0.000)**	**Yes (0.000)**	No (0.330)
2_p562	**Yes (0.000)**	**Yes (0.000)**	No (0.992)
2_p2143	**Yes (0.000)**	**Yes (0.000)**	**Yes (0.000)**

Table 6
Statistical analysis for the R3 indicator results. Statistically significant differences are highlighted in bold.

Sets	Statistically significant differences (Yes/No) and exact p-value		
	MOABC vs. P-ACO	MOABC vs. P-VNS	P-ACO vs. P-VNS
2_p21	**Yes (0.000)**	**Yes (0.000)**	**Yes (0.014)**
2_p32	No (0.792)	**Yes (0.000)**	**Yes (0.000)**
2_p33	**Yes (0.000)**	**Yes (0.004)**	No (0.142)
2_p64	**Yes (0.000)**	**Yes (0.000)**	**Yes (0.000)**
2_p66	**Yes (0.000)**	**Yes (0.000)**	No (0.976)
2_p97	**Yes (0.000)**	**Yes (0.000)**	No (0.338)
2_p273	**Yes (0.006)**	**Yes (0.000)**	**Yes (0.007)**
2_p559	**Yes (0.000)**	**Yes (0.000)**	No (0.593)
2_p562	**Yes (0.000)**	**Yes (0.000)**	No (0.812)
2_p2143	**Yes (0.000)**	**Yes (0.000)**	**Yes (0.000)**

MOABC does not obtain better results (2_p32 and 2_p33), the differences with P-ACO are not significant. If we focus on the differences between MOABC and P-VNS, again the differences are statistically significant in almost all the cases, except for 2_p21 and 2_p33. Therefore, these results also corroborate our previous conclusions (Section 5.3): MOABC obtains better results than P-VNS.

In second place, we apply the statistical analysis to the results from the unary epsilon indicator. Again, after performing the Kolmogorov–Smirnov and Shapiro–Wilk tests we conclude that the results do not follow a Gaussian distribution; therefore, we have to do a non-parametric analysis. We apply the Wilcoxon–Mann–Whitney test by pairs of algorithms. Table 5 shows the p-values obtained from this statistical test, highlighting the cases in which the differences are statistically significant. As we can observe, the differences between MOABC and P-ACO are significant in almost all the cases, except for 2_p32. The same can be said for the differences between MOABC and P-VNS, they are statistically significant in all the cases. Therefore, these statistical results confirm our conclusions of the previous subsection: MOABC obtains better results than P-ACO and P-VNS.

Finally, in third place, we perform the statistical analysis of the results from the R3 indicator. As in the previous indicators, after applying the Kolmogorov–Smirnov and Shapiro–Wilk tests we detect that the results do not follow a Gaussian distribution; therefore, we have to perform a non-parametric analysis. We apply the Wilcoxon–Mann–Whitney test by pairs of algorithms. Table 6 shows the p-values obtained from this statistical test, highlighting the cases in which the differences are statistically significant. As we can see, the differences between MOABC and P-ACO are significant in almost all the cases, except for 2_p32. Moreover, the same can be said for the differences between MOABC and P-VNS, they are statistically significant in all the cases. Therefore, these statistical results corroborate our conclusions of the Section 5.3: MOABC obtains better results than P-ACO and P-VNS.

6. Conclusions and future work

We have developed a metaheuristic solution procedure for the bi-objective orienteering problem. Our proposal, MOABC, is a multi-objective optimization algorithm based on honey bees behavior. Our final motivation is to apply this solution approach to a tourist route planner in order to provide optimized touristic tours according to user preferences (benefit values of each POI) and a tour length constraint. MOABC has been compared with two previously published solution approaches, P-ACO and P-VNS (both are multi-objective approaches). The comparisons were made on benchmark and real-world instances of the problem (a total of 216 instances grouped in 10 sets) and we have used three state-of-art performance multi-objective indicators to report and compare the results of the three algorithms. MOABC approach is better, in general, than P-ACO and P-VNS, according to the average performance indicators, and much better as larger is the problem instance. In fact, the outperformance of MOABC has been confirmed after a statistical analysis. Therefore, we can conclude that MOABC is a very interesting approach to solve the bi-objective orienteering problem.

As said, in the future, we intend to apply this solution approach to a tourist route planner. In addition, we will try to improve these results by using other different solution procedures (for example, other variants of ABC or other possible algorithms). Their design for this specific problem, their implementation and execution, and the comparison with these multi-objective algorithms is of interest for future research. Finally, the application of MOABC to other kinds of orienteering problems is also an interesting line of future work.

Acknowledgments

This work was partially funded by the AEI (State Research Agency, Spain) and the ERDF (European Regional Development Fund, EU), under the contract TIN2016-76259-P (PROTEIN project). This work also was partially supported by the Junta de Extremadura (Spain) under the project IB16002.

References

[1] T. Tsiligirides, Heuristic methods applied to orienteering, J. Oper. Res. Soc. 35 (9) (1984) 797–809, http://dx.doi.org/10.2307/2582629.

[2] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471, http://dx.doi.org/10.1007/s10898-007-9149-x.

[3] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197, http://dx.doi.org/10.1109/4235.996017.

[4] K. Deb, Multi-objective evolutionary algorithms, Springer Handbook of Computational Intelligence, Springer Berlin Heidelberg, 2015, pp. 995–1015, http://dx.doi.org/10.1007/978-3-662-43505-2_49.

[5] C.A.C. Coello, C. Dhaenens, L. Jourdan, Advances in Multi-Objective Nature Inspired Computing, Studies in Computational Intelligence, Vol. 272, Springer, 2010, http://dx.doi.org/10.1007/978-3-642-11218-8.

[6] A. Zhou, B.Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art, Swarm Evol. Comput. 1 (1) (2011) 32–49, http://dx.doi.org/10.1016/j.swevo.2011.03.001.

[7] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: Artificial Bee Colony (ABC) algorithm and applications, Artif. Intell. Rev. 42 (1) (2014) 21–57, http://dx.doi.org/10.1007/s10462-012-9328-0.

[8] Y. Zhang, L. Wu, Optimal multi-level thresholding based on maximum Tsallis entropy via an Artificial Bee Colony approach, Entropy 13 (4) (2011) 841–859, http://dx.doi.org/10.3390/e13040841.

[9] Y. Zhang, L. Wu, S. Wang, Magnetic resonance brain image classification by an improved Artificial Bee Colony algorithm, Prog. Electromagnet. Res. 116 (2011) 65–79, http://dx.doi.org/10.2528/PIER11031709.

[10] S. Wang, Y. Zhang, Z. Dong, S. Du, G. Ji, J. Yan, J. Yang, Q. Wang, C. Feng, P. Phillips, Feed-forward neural network optimized by hybridization of PSO and ABC for abnormal brain detection, Int. J. Imaging Syst. Technol. 25 (2) (2015) 153–164, http://dx.doi.org/10.1002/ima.22132.

[11] W. Zou, Y. Zhu, H. Chen, B. Zhang, Solving multiobjective optimization problems using Artificial Bee Colony algorithm, Discrete Dyn. Nat. Soc. 2011 (2011) 569784, http://dx.doi.org/10.1155/2011/569784.

[12] H. Zhang, Y. Zhu, W. Zou, X. Yan, A hybrid multi-objective Artificial Bee Colony algorithm for burdening optimization of copper strip production, Appl. Math. Model. 36 (6) (2012) 2578–2591, http://dx.doi.org/10.1016/j.apm.2011.09.041.

[13] J. Huo, L. Liu, An improved multi-objective Artificial Bee Colony optimization algorithm with regulation operators, Information 8 (1) (2017) 18, http://dx.doi.org/10.3390/info8010018.

[14] D. Feillet, P. Dejax, M. Gendreau, Traveling salesman problems with profits, Transp. Sci. 39 (2) (2005) 188–205, http://dx.doi.org/10.1287/trsc.1030.0079.

[15] C. Archetti, A. Hertz, M.G. Speranza, Metaheuristics for the team orienteering problem, J. Heuristics 13 (1) (2007) 49–76, http://dx.doi.org/10.1007/s10732-006-9004-0.

[16] P. Vansteenwegen, W. Souffriau, G.V. Berghe, D.V. Oudheusden, A guided local search metaheuristic for the team orienteering problem, Eur. J. Oper. Res. 196 (1) (2009) 118–127, http://dx.doi.org/10.1016/j.ejor.2008.02.037.

[17] I.-M. Chao, B.L. Golden, E.A. Wasil, The team orienteering problem, Eur. J. Oper. Res. 88 (3) (1996) 464–474, http://dx.doi.org/10.1016/0377-2217(94)00289-4.

[18] A. Gunawan, H.C. Lau, P. Vansteenwegen, Orienteering problem: a survey of recent variants, solution approaches and applications, Eur. J. Oper. Res. 255 (2) (2016) 315–332, http://dx.doi.org/10.1016/j.ejor.2016.04.059.

[19] P. Vansteenwegen, W. Souffriau, D.V. Oudheusden, The orienteering problem: a survey, Eur. J. Oper. Res. 209 (1) (2011) 1–10, http://dx.doi.org/10.1016/j.ejor.2010.03.045.

[20] M. Schilde, K.F. Doerner, R.F. Hartl, G. Kiechle, Metaheuristics for the bi-objective orienteering problem, Swarm Intell. 3 (3) (2009) 179–201, http://dx.doi.org/10.1007/s11721-009-0029-5.

[21] D. Purevsuren, G. Cui, S.u. Rehman, Evolutionary multiobjective optimization algorithms with path relinking for bi-orienteering problem, Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on, (2015), pp. 132–135, http://dx.doi.org/10.1109/ICSESS.2015.7339021.

[22] R. Martí, V. Campos, M.G. Resende, A. Duarte, Multiobjective GRASP with path relinking, Eur. J. Oper. Res. 240 (1) (2015) 54–71, http://dx.doi.org/10.1016/j.ejor.2014.06.042.

[23] B.L. Golden, L. Levy, R. Vohra, The orienteering problem, Naval Res. Logist. (NRL) 34 (3) (1987) 307–318.

[24] G.A. Croes, A method for solving traveling-salesman problems, Oper. Res. 6 (6) (1958) 791–812, http://dx.doi.org/10.1287/opre.6.6.791.

[25] D.E. Goldberg, Genetic algorithms in search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[26] M. Mernik, S.-H. Liu, D. Karaboga, M. Črepinšek, On clarifying misconceptions when comparing variants of the Artificial Bee Colony algorithm by offering a new implementation, Inf. Sci. 291 (2015) 115–127, http://dx.doi.org/10.1016/j.ins.2014.08.040.

[27] I.-M. Chao, B.L. Golden, E.A. Wasil, A fast and effective heuristic for the orienteering problem, Eur. J. Oper. Res. 88 (3) (1996) 475–489, http://dx.doi.org/10.1016/0377-2217(95)00035-6.

[28] J. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers, Technical report no. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006. Revised version, Feb.

[29] N. Beume, C.M. Fonseca, M. López-Ibáñez, L. Paquete, J. Vahrenhold, On the complexity of computing the hypervolume indicator, IEEE Trans. Evol. Comput. 13 (5) (2009) 1075–1082, http://dx.doi.org/10.1109/TEVC.2009.2015575.

[30] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G.d. Fonseca, T. 2003.810758, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132, http://dx.doi.org/10.1109/TEVC.2003.810758.

[31] M.P. Hansen, A. Jaszkiewicz, Evaluating the quality of approximations to the non-dominated set, Tech. rep. imm-rep-1998-7, (1998).

[32] N. Veček, S.H. Liu, M. Črepinšek, M. Mernik, On the importance of the Artificial Bee Colony control parameter 'limit', Inf. Technol. Control 46 (4) (2017) 566–604, http://dx.doi.org/10.5755/j01.itc.46.4.18215.

[33] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, 5th edition, Chapman & Hall/CRC, NY, USA, 2011.