European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Multi-objective evolutionary algorithms for a reliability location problem

Javier Alcaraz*, Mercedes Landete, Juan F. Monge, José L. Sainz-Pardo

Departament of Statistics, Mathematics and Computer Science, Center of Operations Research, Miguel Hernández University of Elche, Spain

ARTICLE INFO

Article history:
Received 11 December 2018
Accepted 28 October 2019
Available online xxx

Keywords:
Location
Multi-objective location problems
Reliability models
Pareto frontier
Multi-objective evolutionary algorithms
Metaheuristics

ABSTRACT

Some location problems with unreliable facilities present two different objectives, one consisting of minimizing the opening and transportation costs if none of the facilities fail and another consisting of minimizing the expected transportation costs. Usually, these different targets are combined in a single objective function and the decision maker can obtain some different solutions weighting both objectives. However, if the decision maker prefers to obtain a diverse set of non-dominated optimal solutions, then such procedure would not be effective. We have designed and implemented two multi-objective evolutionary algorithms for the realibility fixed-charge location problem by exploiting the peculiarities of this problem in order to obtain sets of solutions that are properly distributed along the Pareto-optimal frontier. The computational results demonstrate the outstanding efficiency of the proposed algorithms, although they present clear differences.

1. Introduction

Facility Location Problems are formulated in order to obtain the most economical location for a number of facilities, equipment or devices in order to serve the demand of a set of customers. There has been an increasing interest in location problems where some open facilities may become unavailable due to different factors, over the last few years. Current technologies such as sensor location (Li & Ouyang, 2011) and (Antunes & Dolores, 2016), or data placement in networks (Baev & Rajaraman, 2001), to name just a few, require efficient methods to find optimal solutions in a short computation time.

There are cases known as p-median Location Problems in which the number of open facilities is previously set (Alcaraz, Landete, & Monge, 2012), in other cases this number is returned by solving the problem (Alcaraz, Landete, Monge, & Sainz-Pardo, 2015). Problems with fixed installation and allocation costs are known as Fixed-Charge Location Problems (FLPs) (Fernandez & Landete, 2015). If the probability of failure of certain facilities or assignments is taken into account, we are facing Location Problems under uncertainty (Shen, 2011).

The Reliability Fixed-Charge Location Problem (RFLP) is one facility location problem under uncertainty with two different objectives. Given a set of locations where facilities can be installed and a set of customers, and assuming that each facility has a known probability of disruption, the problem consists of deciding where to install the facilities and how to serve the customers, whether failures occur or not, in such a way that the convex combination of two objectives is minimized. The first objective is the total cost if no failure occurs while the second cost is the expected cost if failures take place. Furthermore, the solution of the RFLP entails an open-facility priority list for each customer: for a given scenario, a customer is served from the first available facility on their list. If no facility fails, customers would be served by the first facilities on their list. To model the prioritized assignments, several assignment levels for each customer are defined: we consider that a customer is served from the facility at its rth level if the facilities at previous levels have failed.

The RFLP has been formulated by Snyder and Daskin (2005) where the authors propose an exact method based on Langrangian Relaxation. In Alcaraz et al. (2012) a hybrid methaheuristic is developed to solve this problem and is checked by means of an extensive computational study. Later, Alcaraz et al. (2015) reformulate the mathematical programming model presented above as a set packing problem, studying certain aspects of its polyhedral properties and identifying all the clique facets and propose an improved compact formulation for the problem, checking its performance by means of an extensive computational study.

For multi-objective problems, the Pareto set is composed of all best compromises between the different objectives. The Pareto set is achieved if there are no other dominant solutions in the search space. The Pareto frontier is defined as the image of the Pareto set in the objective space (Coello, Lamont, & Veldhuizen, 2007).

* Corresponding author.
E-mail address: jalcaraz@umh.es (J. Alcaraz).

https://doi.org/10.1016/j.ejor.2019.10.043
0377-2217/

So far, the two-objective RFLP has been solved by grouping together the two objectives in a convex combination. This approach entails that the decision maker has to decide in advance, and based on their subjective experience, the weight for each objective, which at times is not always easy. Moreover, weighting the objectives does not lead, in this type of problems, to get the Pareto front, which can be very useful for the decision maker to evaluate the different alternative solutions. Although this frontier can be obtained with exact approaches, the computation times required to do so are sometimes excessive, as we will show in our computational experiment. Therefore, we think it is necessary to have a multi-objective metaheuristic capable to find the Pareto-optimal front, or at least a good approximation of it in a reasonable time.

Since no heuristic or metaheuristic multi-objective approaches have been developed to obtain the Pareto frontier of the RFLP before, this paper fills the gap contributing to the management of the problem. We have designed two multi-objective evolutionary algorithms: one based on the general scheme proposed by NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002) and another based on the SPEA2 algorithm (Zitzler, Laumanns, & Thiele, 2001). Both are capable of approximating the Pareto optimal frontier in an efficient way by incorporating appropriate problem-specific knowledge. To demonstrate their efficiency, we have carried out a computational study and we have compared the frontier given by the metaheuristics with that obtained by an exact multi-objective procedure. The selection of the metaheuristic multi-objective optimization algorithms is based on the good results that both algorithms have obtained in other multi-objective problems in general (Coello et al., 2007; Pardalos, Žilinskas, & Žilinskas, 2017; Zhou et al., 2011), and in other location problems in particular (Antunes & Dolores, 2016; Bashiri & Rezanezhad, 2015; Eghbali, Abedzadeh, & Setak, 2014).

The remainder of this paper is organized as follows. In Section 2 the problem is described and the notation is introduced. In Section 3, an overview of multi-objective optimization methods is presented. In Section 4, the new multi-objective metaheuristics are described in detail and the specific procedures designed for them are also explained. Section 5 shows the computational study performed to analyze the efficiency of the new approaches, by comparing them with an exact approach. Our conclusions are presented in the final section.

2. Problem description and formulation

Let I be the set of customers and J the set of potential facility locations. Let NF be the subset of facilities in J which may not fail and F the subset of facilities in J which may fail: $J = F \cup NF$, $F \cap NF = \emptyset$. For each customer $i \in I$ a demand h_i is required and the cost of sending one unit of product from facility $j \in J$ to this customer is d_{ij}, however the penalty cost θ_i represents the cost of not serving a customer and the sum of all these costs represents the transportation cost. The non-service of a customer is represented as the service from a dummy facility $u \in NF$ which is at a distance $d_{iu} = \theta_i$. Finally, opening a plant at location $j \in J$ has a cost of f_j ($f_u = 0$) and the sum of all these costs is the set-up cost. If the probability that each facility in F has of failing is q then we can also calculate the expected failure cost as the expected transportation cost taking into account random facility failures. For all $j \in J$, let X_j be a binary variable which takes the value one if a facility is open at location j. For all $i \in I, j \in J, r \in R = \{0, \ldots, |F|\}$ let Y_{ijr} be a binary variable which takes the value one if customer i is allocated to facility j at level r, if j is the r-th backup facility of i. In this problem, two different objectives are formulated: (1) minimizing the sum of the transportation cost and the set-up cost,

$$w_1 = \sum_{j \in J} f_j X_j + \sum_{i \in I} \sum_{j \in J} h_i d_{ij} Y_{ij0};$$

(2) minimizing the expected failure cost,

$$w_2 = \sum_{i \in I} h_i \Big[\sum_{j \in NF} \sum_{r=0}^{|F|-1} d_{ij} q^r Y_{ijr} + \sum_{j \in F} \sum_{r=0}^{|F|-1} d_{ij} q^r (1-q) Y_{ijr} \Big].$$

Therefore, the RFLP is presented as a multi-objective optimization problem with two different objectives while to solve the RFLP, the authors propose a mathematical program with only one objective (Snyder & Daskin, 2005), formulated as the weighted sum of the two objectives of the problem:

$$\text{(RFLP) minimize} \alpha w_1 + (1-\alpha) w_2$$

$$s.t. \sum_{j \in F} Y_{ijr} + \sum_{j \in NF} \sum_{s=0}^{r} Y_{ijs} = 1 \qquad \forall i \in I, r \in R \tag{1}$$

$$Y_{ijr} \le X_j \qquad \forall i \in I, j \in J, r \in R \tag{2}$$

$$\sum_{r \in R} Y_{ijr} \le 1 \qquad \forall i \in I, j \in J \tag{3}$$

$$\begin{aligned} &X_u = 1 \\ &X_j \in \{0,1\} \quad \forall j \in J \\ &Y_{ijr} \in \{0,1\} \quad \forall i \in I, j \in J, r \in R \end{aligned} \tag{4}$$

where α is a value in [0,1]. Constraints (1) state that customer i either has a r-th backup facility or has a s-th backup facility ($s < r$) in NF. Constraints (2) guarantee that customers are assigned to open facilities. Constraints (3) ensure that there is no more than one backup facility at each level and Constraint (4) establishes that the dummy facility is open.

Given that processing this model with the original set R of indexes requires an excessive computational load, the set R is commonly replaced in the literature by $R = \{0, \ldots, 4\}$. In the paper (Snyder & Daskin, 2005), it is assumed that relaxing the integrality of the allocation variables is possible in order to reduce the computational effort, and this assumption has been broadly discussed by Sainz-Pardo et al. Sainz-Pardo, Alcaraz, Landete, and Monge (2017) demonstrating that it can produce errors in some cases.

3. An overview of multi-objective optimization methods

In problems with multiple objectives, as in this case, the multiple objectives add to the difficulty of combinatorial optimization problems, making these types of problems very difficult to solve in an exact way (Ehrgott & Ruzika, 2008) even if they are derived from easy single objective optimization problems.

A good range of methods have been proposed to obtain, or at least approximate, this frontier, including exact, heuristic and metaheuristic strategies. As regards exact methods, one of the most simple approaches is the Weighted Sum Method (WSM) (Zadech, 1963), where the different objectives of the problem are weighted and combined in a single objective function. By varying the different weights appropriately and solving the objective for each combination of weights, the corresponding mathematical program may give different non-dominated solutions. However, this method presents several deficiencies with respect to depicting the Pareto optimal set (Marler & Arora, 2010). The ε-Constraint Method (Levialdi, 1971) is based on a scalarization where one of the objective functions is optimized while all the other objective functions are bounded by means of additional constraints. However, the main drawback of these methods is their inability to capture solutions on non-convex regions of the Pareto

Please cite this article as: J. Alcaraz, M. Landete and J.F. Monge et al., Multi-objective evolutionary algorithms for a reliability location problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2019.10.043

frontier. These non-convex fronts are characteristic of multi-objective combinatorial problems. In addition, evenly varying the weights does not typically result in an even distribution of Pareto solutions (Messac & Mattson, 2004). More recent methods have been developed to overcome these disadvantages. Several variants of the ε-Constraint Method have have been proposed, outperforming the original method considerably, improving its speed and making it capable of managing non-convex regions when integer variables are considered in the models, guaranteeing the complete Pareto set (Ehrgott & Ruzika, 2008; Mavrotas, 2009) Later, these techniques were outperformed again by Mavrotas and FLorios (2013) for problems with more than two objectives. The Comprise Programming Method is able to approximate the Pareto set in both convex and non-convex frontiers (Chen, Wiecek, & Zhang, 1999) but fails to generate well-distributed solutions along the frontier. This drawback is overcome by several methods, the Physical Programming approach (Messac, 1996), the Normal Boundary Intersection Method (Das & Dennis, 1998) or the Normal Constraint Method (Hancock & Mattson, 2013; Messac, Ismail-Yahaya, & Mattson, 2003; Messac & Mattson, 2004), which are capable of generating a set of well-distributed Pareto solutions on convex and non-convex frontiers. Recent results on non-convex multi-objective optimization problems and methods are presented in Pardalos et al. (2017).

Since exact methods might require an excessive computational time to solve large instances, heuristic and metaheuristic algorithms have been developed to solve multi-objective problems and provide the Pareto optimal frontier or an approximation of it.

In the literature, we can find several and very diverse heuristic approaches proposed to solve a wide variety of multi-objective optimization problems. Generally, these heuristics are designed to solve a specific problem and make use of the problem structure to work efficiently. However, we can cite two general heuristics to solve problems with several objectives. Both are the multi-objective version of the single-objective approach. First, the Local Search Multi-objective Heuristic (Ehrgott & Gandibleux, 2004), in which the generation and consideration of neighbors are often based on Pareto optimality (Moalic, Caminada, & Lamrous, 2013). Secondly, the Multi-objective Greedy Heuristic (Ehrgott & Gandibleux, 2004), which performs well in problems where the greedy algorithm can yield an optimal solution of a single objective version of the problem.

These approaches are used as a general framework to develop specific heuristics for the different problems that can use the problem knowledge to approximate the Pareto frontier efficiently or they can work as part of a multi-objective evolutionary algorithm (MOEA). These are, nowadays the most promising ways to solve hard multi-objective optimization problems. Among the general MOEA, we can cite two approaches which are the most used to solve multi-objective problems or as the basis of other hybrid algorithms: the Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb et al., 2002) and the Strength Pareto Evolutionary Algorithm (SPEA2) (Zitzler et al., 2001). NSGA-II handles a population during the evolution process and solutions are sorted into non-domination levels. The algorithm not only offers a set of non-dominated solutions but this set is a good spread of solutions in order to cover a good part of the Pareto frontier. SPEA2 is also based on the concept of non-dominance, and the strength of an individual in the population is calculated depending on how many individuals each individual dominates and is dominated by. It also incorporates mechanisms to guarantee the preservation of boundary solutions. These algorithms and some other similar MOEAs have been hybridized and adapted to solve a wide variety of hard multi-objective combinatorial optimization problems (Coello et al., 2007; Zhou et al., 2011).

3.1. Multi-objective optimization in location problems

In the past two decades, presenting and solving multi-criteria location problems have witnessed substantial growth, and have opened new windows to location science in different businesses. An overview on Multi-objective Location Problems can be found in Nickel, Puerto, and Rodríguez-Chía (2015). Several exact, heuristic and metaheuristic multi-objective approaches have been developed to solve this type of hard location problem (Farahani, Seifi, & Asgari, 2010) and most of them only consider two objectives, since any more would considerably increase the difficulty of the problem.

Considering the reliable location problems, the number of multi-objective approaches is considerably reduced. We can find some recent developments. Konak and Smith (2011) develop a bi-objective genetic algorithm to optimize a Reliable Two-Node Connected Networks where one of the objectives directly considers the reliability of the network. In Alcaraz et al. (2015), the authors present two hybrid metaheuristics, a genetic algorithm and a scatter search approach, to solve the Reliability p-Median Problem formulated in Snyder and Daskin (2005), where two objectives are combined in a single weighted objective function and the results are compared with those given by an exact procedure. Eghbali et al. (2014) study the Multi-objective reliable hub covering location, considering customer convenience and also develop a metaheuristic approach based on the NSGA-II algorithm, proposing a tuning of the algorithm parameters and solving a real case. Bashiri and Rezanezhad (2015) propose a multi-objective genetic algorithm to solve a p-hub location problem and compare the solutions given by the metaheuristic with those obtained with the ε-Constraint Method. Jalalia, Seifbarghyb, Sadeghia, and Ahmadic (2016) study a reliable location problem and propose three different multi-objective metaheuristics, a genetic algorithm, a simulated annealing approach and multi-objective biogeography-based optimization. The proposal of Antunes and Dolores (2016) to solve a real problem of sensor location in water distribution networks to detect contamination events also consists of a genetic algorithm based on NSGA-II.

4. Design of two multi-objective evolutionary algorithms for solving the RFLP

When we deal with a multi-objective NP-hard problem, multi-objective evolutionary algorithms (MOEAs) can be used to obtain a set of non-dominated solutions. In this paper, we propose two different MOEAs to solve the Reliability Fixed-charge Location Problem: a hybrid metaheuristic based on the Non-dominated Sorting Genetic Algorithm NSGA-II (Deb et al., 2002) and another based on the Strength Pareto Evolutionary Algorithm SPEA2 (Zitzler et al., 2001). The purpose of these algorithms is to obtain sets of solutions adequately distributed near the Pareto-optimal frontier in reasonable computation times when compared with the exact algorithms. We would like to emphasize that, both NSGA-II and SPEA2, represent general schemes and adapting them to solve a particular problem involves a considerable design effort, both in terms of an appropriate encoding for the solutions and appropriate procedures that are able to adequately combine the solutions. The elements designed must be capable of incorporating the singularities of the problem to explore the solution space in an efficient way. In the next section, we describe the main features and characteristics that we have designed and are shared by both metaheuristics: the way to encode the solutions to the problem, the mechanism to generate the initial population of solutions and the crossover procedure, which incorporates two additional mechanisms, *adding* and *deleting* in order to generate quality solutions in an efficient way. Later we present the main scheme of

Please cite this article as: J. Alcaraz, M. Landete and J.F. Monge et al., Multi-objective evolutionary algorithms for a reliability location problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2019.10.043

both metaheuristics and the specific procedures and operators they employ.

4.1. Design of common features of the metaheuristics

4.1.1. Encoding

In the design of genetic algorithms, the first and one of the most important decisions to be taken consists of designing the way to encode the chromosomes, also called individuals or solutions. In the RFLP, a feasible solution is given by determining the facilities to be open. Once the set of open facilities is known, we can obtain the optimal value of the allocation variables Y_{ijr} of the mathematical model presented before, by assigning to customer i the nearest open facility at $r=0$, the second nearest open facility at $r=1$ and so on.

In order to represent the open facilities that offer a solution, one possibility is to to use a chromosome string with a fixed length $|J|$ (the number of facilities in the problem) where each one of the genes is assigned to a different facility in J. Then, the binary alphabet is used, in order to determine which of the facilities are open. If a given gene contains the number 1, it means that the corresponding facility is open and and 0 means the opposite. However, we have designed a chromosome string with a variable length. The length of a given chromosome will be equal to the number of open facilities in the solution that it represents. The alphabet employed in this encoding is given by the natural numbers between 1 and the number of facilities in the problem. Each gene will contain a different number of the alphabet, representing a different facility to be open. For example (1, 5, 7, 9) represents a solution where facilities 1, 5, 7 and 9 are open and the rest are closed. The chromosome string is not an ordered list, that is, for example, (1, 5, 9, 7) and (7, 1, 5, 9) are valid chromosomes and represent the same solution. Obviously, the number of one facility cannot be represented more than once in a given chromosome.

4.1.2. Initial population

The first step consists of creating the initial population of size N. Two ways are usually used to create solutions for the first population: the solutions are the result of applying a heuristic method which obtains feasible solutions or they are built with a random generator. The first method has the advantage of providing good quality solutions, but requires more computation time than the second. Generating solutions randomly is faster but the quality of the solutions will probably be worse than with the heuristic approach. Some preliminary results seem to indicate that the second mechanism performs better and is the one we have implemented in our algorithm.

To randomly generate each one of the N solutions of the initial population we first decide the length of the chromosome, that is, the number of open facilities that the solution represents. To do so, a random number p, being $1 \le p \le |J|$ is generated. Then we randomly choose p plants, with homogeneous probability, among the set of facilities in order to build the chromosome. As we mentioned before, the information contained in the chromosome, i.e., the plants selected to be open, can be directly transformed into the solution to the mathematical model (RFLP).

4.1.3. Crossover

We have designed a competitive crossover mechanism that incorporates problem-specific knowledge in order to generate quality and diverse solutions. Given two parents selected to undergo the crossover operation through the selection mechanism, three offspring are created and these descendants are included in the population that is being created. The three offspring have different lengths, that is, the number of open plants in each of the solutions generated is different. It must be kept in mind that, with the encoding we propose, the length of the chromosomes is variable between 1 and $|J|$, the number of plants that can be open in the problem. This mechanism ensures diversity with respect to the length of the solutions.

Algorithm 1 shows, in pseudocode, the procedure carried out to generate the three offspring, Off_1, Off_2 and Off_3 from the two parents, M and F. The first step consists of fixing the desired lengths of the offspring. The first offspring, Off_1, will have a random length, l_1, between 1 and the minimum length of the parents. The length of the second offspring, l_2, is randomly chosen between the lengths of both parents and, finally, the length of Off_3 will be between the longest parent and the number of plants that can be potentially open, $|J|$. Generating three offspring with the lengths chosen in this way, we ensure that all the solutions, regardless of their length, can be generated with this procedure. This introduces the desired variability in the population with respect to the number of plants open.

Algorithm 1: procedure_crossover(M, F).

```
1  l1 = random_int(1, min(length(M), length(F)));
2  l2 = random_int(length(M), length(F));
3  l3 = random_int(max(length(M), length(F)), |J|);
4  P = gene_union(M, F);
5  for i=1 to 3 do
6      if length(P)=li then
7          Offi = P;
8      else
9          j=random(1,2);
10         if li < length(P) then
11             Offi = deleting(P, li, Wj);
12         else
13             Offi = adding(P, li, Wj);
14 return (Off1, Off2, Off3)
```

Once the lengths of the offspring have been chosen, these offspring need to be generated. In order to do so, we first take the union of the parents' genes, obtaining a feasible solution, P, with a given length. If the length of P is equal to any of the three offspring lengths, the corresponding offspring is directly P. If not, the solution P will be transformed into the corresponding offspring by the procedure of *adding* or *deleting* genes until reaching the desired length. We have carefully designed these procedures in order to increase or decrease the length of the solution in an efficient way instead of in a random way. To invoke any of these two procedures, it is necessary first to randomly choose which of the two objectives w_1 or w_2 will receive priority treatment, and it is known as *priority objective*.

The procedures of *adding* and *deleting*, which are presented in pseudocode in Algorithms 2 and 3 receive the solution to be lengthened or shortened P, the desired length l and the priority objective W as arguments. Both iterate, increasing or decreasing the length of the solution by one unit until the given length is reached. These procedures start working with a copy of P, and a gene is added/deleted to/from the candidate in each iteration until the length of the candidate is equal to the desired length. The *adding* procedure, selects, in each iteration, the facility, not present in the candidate, that leads to the best value of the priority objective when it is incorporated to the candidate, and this facility is included in the candidate solution, increasing its length by one. On the other hand, in each round, the *deleting* mechanism selects which of the facilities present in the candidate solution obtains the best value in the priority objective when it is eliminated, and is

Please cite this article as: J. Alcaraz, M. Landete and J.F. Monge et al., Multi-objective evolutionary algorithms for a reliability location problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2019.10.043

Algorithm 2: procedure_adding(P, l, W).

```
while length(P) < l do
    best_obj_value = ∞;
    for j = 1 to |J| do
        Q = P;
        if belongs(j, P)= false then
            Q =add_facility(j, P) ;
            obj_value =evaluate_solution(Q, W);
            if obj_value < best_obj_value then
                best_obj_value = obj_value;
                k = j;
    P =add_facility(k, P) ;
return (P)
```

Algorithm 3: procedure_deleting(P, l, W).

```
while length(P) > l do
    best_obj_value = ∞;
    for i=1 to length(P) do
        Q =delete_facility(P(i), P) ;
        obj_value =evaluate_solution(Q, W);
        if obj_value < best_obj_value then
            best_obj_value = obj_value;
            k = i;
    P =delete_facility(P(k), P) ;
return (P)
```

then deleted from the candidate solution, reducing the length of the candidate by one unit.

The crossover procedure we have designed incorporates, through the *adding* and *deleting* procedures, an important amount of problem knowledge. This fact allows creating quality offspring but requires an important computational effort. Moreover, these procedures generate very diverse solutions that, inheriting the good characteristics of their parents, allow new genes not present in any of the parents to be introduced in the offspring or even not inherit genes present in both parents. This mechanism could allow genetic material that may be lost during the evolution process to be reintroduced or even new characteristics which are not present in any of the population solutions. Therefore, any solution of the problem space could be created through this sophisticated mechanism. For this reason, the two metaheuristics we propose in this work do not require a mutation mechanism, given that the variability provided by this type of procedure is already achieved by the crossover mechanism designed.

4.2. Multi-objective metaheuristic based on NSGA-II

Algorithm 4 shows the main loop of the multi-objective genetic algorithm (MOGA) that we propose to solve the RFLP, which is based on the general scheme of NSGA-II. An initial population of size N is generated, and then a new population is obtained from it and changed iteratively step by step. In every iteration, an offspring population Q_t is created from the current population P_t. After combining P_t and Q_t the population R_t of size $2N$ is built. In order to reduce the size of the combined population to N, R_t is sorted according to non-dominance of the solutions and then the N members of R_t with best non-domination rank are selected to belong to the new current population P_{t+1}. Any ties among members of the last non-domination rank selected are broken calculating the

Algorithm 4: NSGA-II_RFLP.

```
P_0 = create_initial_population(N);
t = 0;
while not stopping_criterion do
    Q_t = make_new_population(P_t);
    R_t = P_t ∪ Q_t;
    F = fast_non_dominated_sort(R_t);
    P_{t+1} = ∅;
    i = 1;
    while |P_{t+1}| + |F_i| ≤ N do
        P_{t+1} = P_{t+1} ∪ F_i;
        i = i + 1;
    if |P_{t+1}| < N then
        i_distance_assignement(F_i) ;
        sort(F_i, i_distance);
        P_{t+1} = P_{t+1} ∪ F_i[1 : (N − |P_{t+1}|)];
    t = t + 1;
F = fast_non_dominated_sort(P_{t−1});
return F_1
```

distances between the solutions in that set. This main loop will be repeated until the stopping criterion is satisfied. Then, the non-dominated solutions will be extracted from the final population to build the definitive set of solutions as a result of the algorithm being executed. The stopping criterion could consist of defining a maximum processing time, limiting the number of iterations without changes in the population or similar conditions.

In multi-objective problems, the usual way to decide on the best solutions from a set, is to select the non-dominated solutions. In order to classify the solutions, different ranks of domination are used. For this reason, NSGA-II algorithms are based on non-dominance. Given a set of solutions, all the non-dominated solutions are assigned to a non-domination rank equal to 0 ($i_{rank} = 0$) and form the first non-dominated frontier F_1, while the rest of the solutions in the set that are only dominated by solutions on frontier F_1 will score $i_{rank} = 1$ and form the second frontier F_2, the solutions which are only dominated by solutions from fronts F_1 or F_2 will score $i_{rank} = 2$ and form F_3, and so on.

Thus, if one solution is only dominated by solutions with a higher i_{rank} and the solutions with a lower i_{rank} do not dominate it, the non-domination rank classifies the solutions in different fronts with the same level of dominance (i_{rank}) and it is a way to establish different non-dominance levels.

As we are also interested in obtaining sets of well-distributed solutions, that is, solutions which cover most of the Pareto-optimal frontier, the distance of one solution in a set provides the degree of proximity to other solutions of the set around it. It is calculated by the average side length of the cuboid enclosing the solution from the normalized objective function. Thus, in the problems with two objective functions w_1 and w_2, the *i_distance* of a solution i in a set S is calculated by:

$$d_i(S) = \frac{w_1[i+1] - w_1[i-1]}{max_1(S) - min_1(S)} + \frac{w_2[i+1] - w_2[i-1]}{max_2(S) - min_2(S)}$$

where max_1 and max_2 are the maximum values of both objectives given by the solutions in S and min_1 and min_2 are the minimum values.

As we mentioned before, when we want to reduce the size of the population from $2N$ to N we choose, in order, the solutions with the best domination rank. If all the members with identical domination rank cannot form part of the new population, we choose those with a larger crowding distance in order to select diverse solutions. For a more detailed description of the general

Please cite this article as: J. Alcaraz, M. Landete and J.F. Monge et al., Multi-objective evolutionary algorithms for a reliability location problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2019.10.043

scheme of NSGA-II and how the fitness of individuals is calculated, readers are referred to Deb et al. (2002). In the following paragraphs, the details and procedures carried out in the metaheuristic approach are presented.

The technique to generate a new population is one of the most important procedures to be designed in the metaheuristic based on NSGA-II and the one which distinguishes some procedures from others. It consists, basically, of two steps, which are repeated until the new population is fulfilled: first, the selection procedure selects two chromosomes in the current population to undergo the crossover operation. Then, the selected parents undergo the crossover operation, described above, which has been carefully designed, incorporating problem-specific knowledge in order to generate quality offspring. The result of the crossover is a set of three offspring with different lengths that are included in the new population.

NSGA-II employs a binary tournament selection mechanism and the crowded-comparison operator is used to decide the winner of the tournament. This operator guides the selection process at the various stages of the algorithm toward a uniformly distributed Pareto-optimal frontier. In this case, the tournament is won by the candidate with a better non-domination rank, and ties are broken choosing the chromosome with a higher crowding-distance. Therefore, the crowding-distance needs to be calculated for each solution in every iteration when a new population is formed. In order to avoid it, he have designed a new selection mechanism developed in two steps, which reduces the number of calculations needed by NSGA-II. In the first step, the non domination rank of the parent to be selected is chosen. In the second step, one solution of that rank is randomly selected as a parent. Therefore, the crowding-distance of every individual in the population is not needed. These two steps are repeated for each parent to form a couple. The way in which the i_{rank} of the parent is selected is guided by the roulette wheel selection mechanism. To prioritize the selection of solutions with small i_{rank} values, we assign a weight of $(l - i_{rank})/\sum_{j=1}^{l} j$ to each rank, where l is the number of non-domination ranks. Then, we use the roulette wheel selection to select a rank. Once the rank has been selected, in the second step, one solution of that rank is randomly selected as one of the parents. This procedure is performed twice, one for each parent.

4.3. Multi-objective metaheuristic based on SPEA2

Algorithm 5 shows the general scheme of SPEA2 and represents the main loop of the multi-objective metaheuristic that we propose to solve the RFLP. The first step consists of generating the initial population, P_0, sized N and creating an empty set $\overline{P}_0$, the archive. Then, the following steps are performed by iteration. First, the fitness assignment of the solutions in P_t and $\overline{P}_t$ are calculated. Then, P_t and $\overline{P}_t$ are combined to create the population R_t and the non-dominated solutions of that population are extracted to build the updated archive $\overline{P}_{t+1}$. If the size of the updated archive exceeds a predefined size $\overline{N}$, reduce $\overline{P}_{t+1}$ by means of the truncation operator. Otherwise, if the size is less than $\overline{N}$ then fill $\overline{P}_{t+1}$ with dominated solutions in $\overline{P}_t$ and P_t. Then, a procedure to generate a new population is applied over the archive. This main loop will be repeated until the stopping criterion is satisfied. The result of the algorithm is the set P_t, in which all the solutions are non-dominated.

Algorithm 5: SPEA2_RFLP.

```
1  P_0 = create_initial_population(N);
2  t = 0;
3  P̄_0 = ∅;
4  while not stopping_criterion do
5      fitness_assignment(P_t, P̄_t);
6      R_t = P_t ∪ P̄_t;
7      P̄_{t+1} = extract_non_dominated(R_t);
8      if size(P̄_{t+1}) > N̄ then
9          P̄_{t+1}=reduce(P̄_{t+1});
10     else
11         P̄_{t+1}=fill(P̄_{t+1});
12     P_{t+1} = generate_new_population(P̄_{t+1});
13     t = t + 1;
14 return P̄_t
```

To calculate the fitness of a solution, SPEA2 considers both the solutions it dominates and the solutions that dominate it. Each solution in the population P_t and the archive $\overline{P}_t$ is assigned a strength value $S(i)$ representing the total number of solutions in P_t and $\overline{P}_t$ that it dominates. The raw fitness $R(i)$ of an individual i is calculated as the sum of the strengths of the individuals in P_t and $\overline{P}_t$ that dominate i. Additional density information is incorporated to the fitness of an individual in order to discriminate between solutions with the same raw fitness. For each individual i, we calculate the distances, in the objective space, to all individuals in archive and population and these distances are stored in a list $L(i)$, which is sorted in increasing order. The density $D(i)$ of an individual i is given by the inverse of the distance to the kth element of $L(i)$ plus 2, for $k = \sqrt{N + \overline{N}}$. Finally, the fitness of an individual i, $F(i)$ is given by $F(i) = R(i) + D(i)$. More details about the general scheme of SPEA2 and how the fitness of individuals is calculated can be consulted in Zitzler et al. (2001)

In SPEA2, the mechanism to build the new population is, as it occurs in NSGA-II, the one which needs to be carefully designed based on the problem to solve and is one of the most important procedures. It should include a mechanism to select the parents to undergo the crossover operation, the crossover itself and, if designers considers it necessary, a mutation operator. In our case, we have implemented the standard binary tournament proposed in SPEA2. As regards the crossover operation, we have implemented the crossover technique we have designed and which has been described in section 4.1.3, which makes use of the *deleting* and *adding* procedures to combine the information contained in the parents producing quality offspring, in an efficient way. As we mentioned before, this crossover procedure has been designed to introduce a large amount of variability in the population and that is the reason why our metaheuristics do not need a mutation mechanism.

5. Computational experience

We have carried out a computational experience in order to analyze the performance of the metaheuristics we have designed to solve the RFLP. We have tested our techniques over a set of well-known instances from the page http://www.math.nsc.ru/AP/benchmarks/UFLP/Engl/uflp_dg_eng.html.

The transportation matrix has 10 non-forbidden (non-infinite) elements from the set {0, 1, 2, 3, 4} in each column. In other words, each customer has 10 potential facilities chosen at random with uniform distribution from all facilities. The distance from a given customer to one of its ten potential facilities is in {0, 1, 2, 3, 4}. The rest of the distances are infinite. We have solved a set of instances with 50 nodes obtained by reading distances like $d_{\lfloor i/2 \rfloor, \lfloor j/2 \rfloor}$ $\forall i, j \in \{1, \ldots, 100\}$. In all these data sets, the set of customers and the set of facilities are equal i.e. all the cities act as customers and all of them may become a facility. The instances are classified in three groups depending on their difficulty for exact methods based on the linear relaxation: Gap-A, Gap-B, Gap-C. Every group consists of 30 instances. Fixed costs are $f_j = 500$ $\forall j \in F$ and emergency costs $\theta_i = 100$ $\forall i \in V$. The parameter q is set to 0.05.

Please cite this article as: J. Alcaraz, M. Landete and J.F. Monge et al., Multi-objective evolutionary algorithms for a reliability location problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2019.10.043

Table 1
AUGMECON vs. NSGA-II_RFLP. Comparison of frontiers for GapA instances.

	AUGMECON		NSGA-II_RFLP						
Instance	#FP	T.	#FP	#Non-dom.	#Dom.	% Success	%Dom.	HVR	T.
gapA332	20	9521	19	15	4	75.00	21.05	0.9998	101
gapA432	18	5455	16	13	3	72.22	18.75	0.9990	57
gapA532	23	17,145	22	17	5	73.91	22.73	0.9976	112
gapA632	18	11,713	13	10	3	55.56	23.08	0.9970	56
gapA732	26	31,265	22	16	6	61.54	27.27	0.9925	81
gapA832	26	17,657	23	21	2	80.77	8.70	0.9984	74
gapA932	17	6207	17	13	4	76.47	23.53	0.9986	97
gapA1032	22	22,412	18	15	3	68.18	16.67	0.9971	53
gapA1132	26	5784	24	20	4	76.92	16.67	0.9992	116
gapA1232	17	10,554	17	17	0	100.00	0.00	1.0000	88
gapA1332	26	29,582	23	20	3	76.92	13.04	0.9995	83
gapA1432	26	33,035	26	18	8	69.23	30.77	0.9984	106
gapA1532	23	20,850	23	17	6	73.91	26.09	0.9990	71
gapA1632	24	10,835	20	19	1	79.17	5.00	0.9968	45
gapA1732	26	12,780	23	19	4	73.08	17.39	0.9995	97
gapA1832	24	16,659	23	19	4	79.17	17.39	0.9990	115
gapA1932	14	8492	14	11	3	78.57	21.43	0.9997	43
gapA2032	19	7434	17	15	2	78.95	11.76	0.9985	85
gapA2132	26	39,674	24	8	16	30.77	66.67	0.9961	86
gapA2232	20	21,582	20	12	8	60.00	40.00	0.9994	156
gapA2332	19	12,578	17	14	3	73.68	17.65	0.9987	71
gapA2432	16	3655	16	16	0	100.00	0.00	1.0000	57
gapA2532	21	12,359	18	8	10	38.10	55.56	0.9931	87
gapA2632	22	27,943	20	13	7	59.09	35.00	0.9984	120
gapA2732	21	19,263	21	9	12	42.86	57.14	0.9984	131
gapA2832	23	13,501	20	17	3	73.91	15.00	0.9991	109
gapA2932	19	27535	17	12	5	63.16	29.41	0.9992	71
gapA3032	17	6211	17	17	0	100.00	0.00	1.0000	66
gapA3132	24	14,420	23	18	5	75.00	21.74	0.9980	100
gapA3232	18	7947	18	18	0	100.00	0.00	1.0000	149

Our experiments were conducted on a PC with a 2.60 GHz Intel Celeron, 3.5 GB of RAM, and operating system Microsoft Windows 7 Professional. To solve the linear programming models, we have used the optimization engine IBM ILOG CPLEX v12.6. The algorithm has been coded in C++ and tested on the same PC previously described in order to maintain the same experimental environment.

The population size of both metaheuristics has been fixed to 100 individuals and in the case of SPEA2_RFLP the archive size is also 100. These parameters have been chosen after some preliminary experiments varying these values. In order to obtain an approximation of the Pareto-optimal frontier we have run both new metaheuristics, stopping them after one hundred iterations without changing the new population. To generate all the points on the Pareto front for each instance, we have implemented and run the effective implementation of the ε-Constraint Method (Mavrotas, 2009) (AUGMECON), which is suitable for generating the exact Pareto set for binary linear problems in an efficient way. To prevent a large number of solutions which only differ by a negligible amount, we define parameter δ:

- a solution with objective values w_1^1 and w_2^1 is considered coincident with another solution with objective values w_1^2 and w_2^2 if $|w_1^1 - w_1^2| \leq \delta$ and $|w_2^1 - w_2^2| \leq \delta$.
- a solution with objective values w_1^1 and w_2^1 is non-dominated by another solution with objective values w_1^2 and w_2^2 if $w_1^2 - w_1^1 > \delta$ or $w_2^2 - w_2^1 > \delta$.

We must consider that, if we add one additional open facility to a given solution, it will always produce a new non-dominated solution with a decrease of the value in w_2. However, the second statement given above prevents a large number of solutions which only differ by a negligible amount (lower than δ) in the value of w_2, to be generated.

In AUGMECON, it is necessary to fix the parameter to move the right hand side in the corresponding iterations. So as not to lose solutions of the Pareto frontier, the value of the this parameter must be lower or equal to δ. In our experiments, we have set this parameter equal to $\delta = 0.1$.

Tables 1 and 2 compare the results obtained when solving GapA instances with the exact method, AUGMENCON, and the corresponding metaheuristic, NSGA-II_RFLP and SPEA2_RFLP respectively. The three columns with the label *#FP* indicate the number of frontier points and is equal to the cardinal of the set of non-dominated points that each one of the methods gives as result, that is, the number of points belonging to the frontier found by the corresponding method. Columns *#Non-dom.* show the number of points on the frontier given by the corresponding metaheuristic which are non dominated by any point of the exact frontier, i.e., the number of points that belong to the optimal Pareto-frontier. Likewise, columns *#Dom.* represent the number of solutions found by the corresponding metaheuristic that are dominated by any of the points given by the exact approach. Columns *% Success* show the percentage of heuristic solutions that belong to the Pareto frontier. Given that these previous percentages do not indicate the proximity from the dominated solutions obtained by the metaheuristics to the Pareto frontier, we have also computed the *hypervolume ratio* metric proposed by Zitzler and Thiele (Zitzler, Deb, & Thiele, 2000), that provides a better indicator about the proximity and the spread between the heuristic frontier and the optimal Pareto frontier. This metric is presented in columns labeled *HVR*. The *hypervolume ratio* is calculated as $HVR = HV(H)/HV(P)$ where $HV(H)$ and $HV(P)$ are the volume of the heuristic Pareto set and the optimal Pareto set, respectively. Clearly, this ratio indicates the proportion of coverage reached by the heuristic method, i.e., the proximity of the heuristic frontier to the exact Pareto frontier. The processing runtime measured in seconds is represented in columns labeled *T*.

In Table 1, we can observe, for example, in the first row, that gapA332 has a Pareto-optimal front with 20 points and NSGA-

Please cite this article as: J. Alcaraz, M. Landete and J.F. Monge et al., Multi-objective evolutionary algorithms for a reliability location problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2019.10.043

Table 2
AUGMECON vs. SPEA2_RFLP. Comparison of frontiers for GapA instances.

	AUGMECON		SPEA2_RFLP						
Instance	#FP	T.	#FP	#Non-dom.	#Dom.	% Success	% Dom.	HVR	T.
gapA332	20	9521	16	12	4	60.00	25.00	0.9995	79
gapA432	18	5455	17	11	6	61.11	35.29	0.9987	101
gapA532	23	17,145	21	13	8	56.52	38.10	0.9971	141
gapA632	18	11,713	13	9	4	50.00	30.77	0.9971	122
gapA732	26	31,265	22	19	3	73.08	13.64	0.9928	181
gapA832	26	17,657	21	15	6	57.69	28.57	0.9983	99
gapA932	17	6207	19	11	8	64.71	42.11	0.9985	153
gapA1032	22	22,412	18	13	5	59.09	27.78	0.9968	113
gapA1132	26	5784	22	19	3	73.08	13.64	0.9989	76
gapA1232	17	10,554	16	11	5	64.71	31.25	0.9997	59
gapA1332	26	29,582	23	16	7	61.54	30.43	0.9995	194
gapA1432	26	33,035	22	16	6	61.54	27.27	0.9980	149
gapA1532	23	20,850	22	16	6	69.57	27.27	0.9988	156
gapA1632	24	10,835	20	14	6	58.33	30.00	0.9964	110
apA1732	26	12,780	20	15	5	57.69	25.00	0.9992	85
gapA1832	24	16,659	19	15	4	62.50	21.05	0.9988	135
gapA1932	14	8492	14	10	4	71.43	28.57	0.9995	61
gapA2032	19	7434	15	12	3	63.16	20.00	0.9985	87
gapA2132	26	39,674	19	13	6	50.00	31.58	0.9958	96
gapA2232	20	21,582	18	11	7	55.00	38.89	0.9991	85
gapA2332	19	12,578	16	11	5	57.89	31.25	0.9986	132
gapA2432	16	3655	15	10	5	62.5	33.33	0.9997	139
gapA2532	21	12,359	17	12	5	57.14	29.41	0.9930	81
gapA2632	22	27,943	19	13	6	59.09	31.58	0.9984	89
gapA2732	21	19,263	16	8	8	38.10	50.00	0.9981	90
gapA2832	23	13,501	18	12	6	52.17	33.33	0.9989	125
gapA2932	19	27,535	15	11	4	57.89	26.67	0.9992	78
gapA3032	17	6211	15	11	4	64.71	26.67	1.0000	72
gapA3132	24	14,420	17	12	5	50.00	29.41	0.9977	78
gapA3232	18	7947	17	11	6	61.11	35.29	0.9999	98

II_RFLP gives a frontier with 19 points, of which 15 are non-dominated and the other 4 are dominated by any point of the exact frontier. This gives a percentage of success of $15/20 \equiv 75\%$, i.e. three quarters of the points of the metaheuristic frontier belong to the Pareto optimal front and $4/19 \equiv 21.05\%$ of the solutions found by NSGA-II_RFLP are dominated points and do not belong to the exact frontier.We can observe that there are several cases in which the metaheuristic obtains all the points of the Pareto-optimal front as in instances *A*2432, *A*3032, *A*1232 and *A*3232. The worst case reported is instance *A*2132 with 30.77% of solutions coincident with the Pareto frontier, and 66.67% of the solutions dominated. The *HVR* column shows the large proximity between the frontiers given that it is near 100% in all the instances. The computational times needed by the two approaches, exact and metaheuristic, differ considerably. The time needed by the exact method to find the optimal Pareto frontier varies from 1 to 11 hours while the times employed by the metaheuristic to find a good approximation of the frontiers varies between 43 seconds and 2.6 minutes. Although employing other exact approaches could lead to the optimal Pareto front in lower computation times, these are for large instances, in most cases, excessive. Although employing other exact approaches could lead to the optimal Pareto front in lower computation times, these are for hard and large instances, in most cases, excessive.

If we observe the results given by the metaheuristic based on SPEA2 when solving the instances in GapA shown in Table 2 and we compare them with those given by NSGA-II_RFLP we quickly realize that, for those instances, the genetic metaheuristic improves the performance of SPEA2_RFLP. In most instances *%Success*, *%Dom* and the computation time is better for NSGA-II_RFLP than for SPEA2_RFLP. However, SPEA2_RFLP obtains good results because the percentage of success varies from 38.10 to 73.08, and HVR is always near 100%. For reasons of space, we have not represented the details for instances of types GapB and GapC, given that their results are very similar to those in GapA type.

The results for the three types of instances have been summarized in Tables 3 and 4 that show, on average, the number and percentage of dominated and non-dominated solutions on the frontier, HVR value and processing number of seconds when solving the instances by the exact and metaheuristic methods. As we previously mentioned, rows GapA, GapB and GapC have been obtained by processing 30 instances, so each cell in these rows represents the average of 30 instances; the last row of both tables represents the total averages obtained from 90 instances. We want to highlight the high percentage, on average, of successful solutions versus the low percentage of dominated solutions, thus clearly showing the good convergence of the metaheuristic algorithms.

In Table 3 we can observe that, on average the percentage of success, that is, the percentage of points of the Pareto frontier given by NSGA-II_RFLP is always greater than 70%, regardless of the group of instances solved and the *hypervolume ratio* metric, disaggregated by type of instances, is always close to 100%, reporting on average a value of 99.85%, which indicates that the reduced number of points given by the heuristic that do not belong to the Pareto frontier are really very close to it. The processing time employed by the techniques for solving each group of instances is extremely different. For example, in instances type GapA, NSGA-II_RFLP needs, on average, less than 1.5 minutes to solve an instance against the near 4.5 hours needed by the exact approach. Regarding the total average, the exact method needs 14290.50 seconds to solve each instance versus an average of 95.98 seconds needed by the metaheuristic based on NSGA, which is three orders of magnitude difference.

Table 4 shows the summarized results given by SPEA2_RFLP. We can observe that the metrics are very similar in the three groups of instances: the percentage of success is near 60%, the *%Dom* is

Table 3
AUGMECON vs. NSGA-II_RFLP. Comparison of frontiers by group of instances.

	AUGMECON		NSGA-II_RFLP						
Instances	#FP	T.	#FP	#Non-dom.	#Dom.	% Success	% Dom.	HVR	T.
GapA	21.37	16134.93	19.70	15.23	4.47	71.3	22.7	0.9983	89.43
GapB	18.14	10351.55	16.79	13.31	3.48	73.4	20.7	0.9985	91.72
GapC	21.45	16321.41	19.72	15.86	3.86	73.9	19.6	0.9986	107.00
Overall	20.32	14290.50	18.74	14.81	3.93	72.9	21.0	0.9985	95.98

Table 4
AUGMECON vs. SPEA2_RFLP. Comparison of frontiers by group of instances.

	AUGMECON		SPEA2_RFLP						
Instances	#FP	T.	#FP	#Non-dom.	#Dom.	% Success	% Dom.	HVR	T.
GapA	21.37	16134.93	18.06	12.73	5.33	59.7	29.8	0.9982	108.80
GapB	18.14	10351.55	15.87	11.03	4.84	60.8	30.5	0.9984	93.63
GapC	21.45	16321.41	18.13	13.03	5.10	60.7	28.1	0.9985	105.90
Overall	20.32	14290.50	17.35	12.26	5.09	60.41	29.47	0.9984	102.78

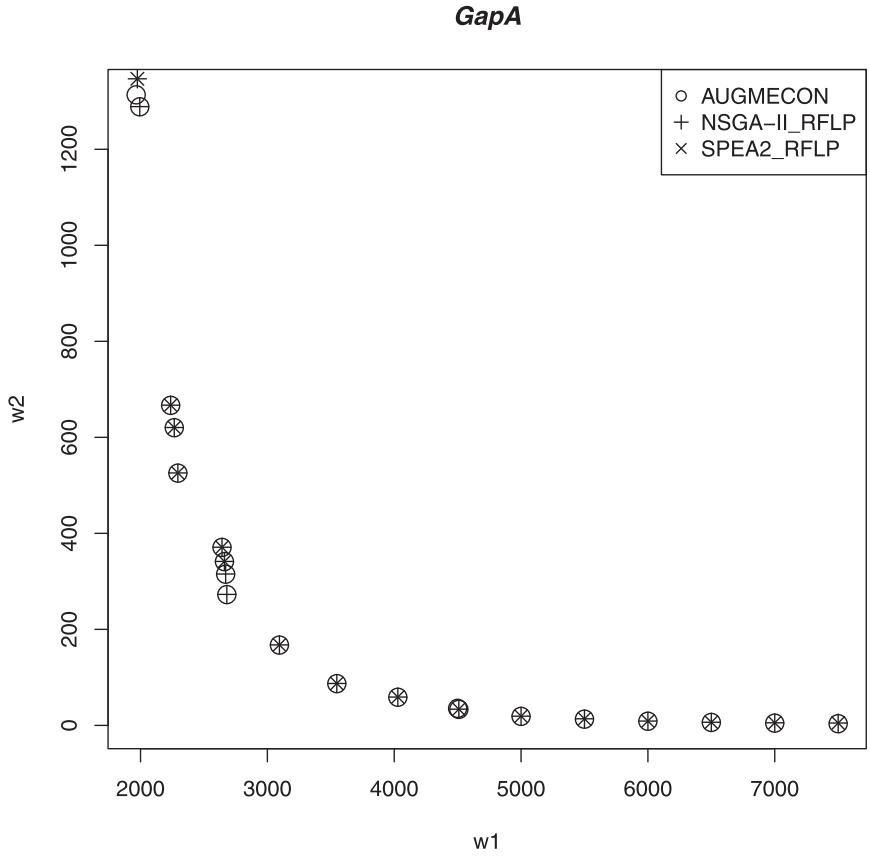

Fig. 1. Exact and approximated Pareto frontiers for GapA332.

also stable and rounds 30%, computation times are just over one minute and a half and the HVR is very near 100%. Comparing the results in Tables 3 and 4 we can deduce that, in this computational experiment, the performance of NSGA-II_RFLP is clearly superior to SPEA2_RFLP. In all the metrics the results of the first are better than those of the second. The success rate of NSGA-II_RFLP is ten points higher than that of SPEA2_RFLP and the *%Dom* is almost 8 points lower for the first method. However, the values of metric HVR are very similar for both metaheuristics, which indicates that although the genetic metaheuristic finds better frontiers than that based on SPEA2, the points that do not belong to the exact frontier are at the same distance from this in both cases. The computation times are smaller for the first algorithm, although they are very similar in both cases, around one minute and a half.

In order to compare the frontiers given by the metaheuristics proposed and the optimal Pareto-front in a graphic way, we have also attached Figs. 1–3 in which the points belonging to each one of the three frontiers are drawn for the first instance of each type, i.e., for instances gapA332, gapB331 and gapC333, respectively.. The rest of instances produce similar patterns. We can observe the close similarities between the three frontiers in the three instances presented. Most of the points given by the proposed techniques coincide with the exact points and, in the cases that it does not occur, the points are really very close. In the third instance, we can observe that there are more exact points that have not been found by the metaheuristics that have found points that are a little further away. However, as we have seen before, there do not seem to be large differences between instances of the three groups.

Please cite this article as: J. Alcaraz, M. Landete and J.F. Monge et al., Multi-objective evolutionary algorithms for a reliability location problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2019.10.043

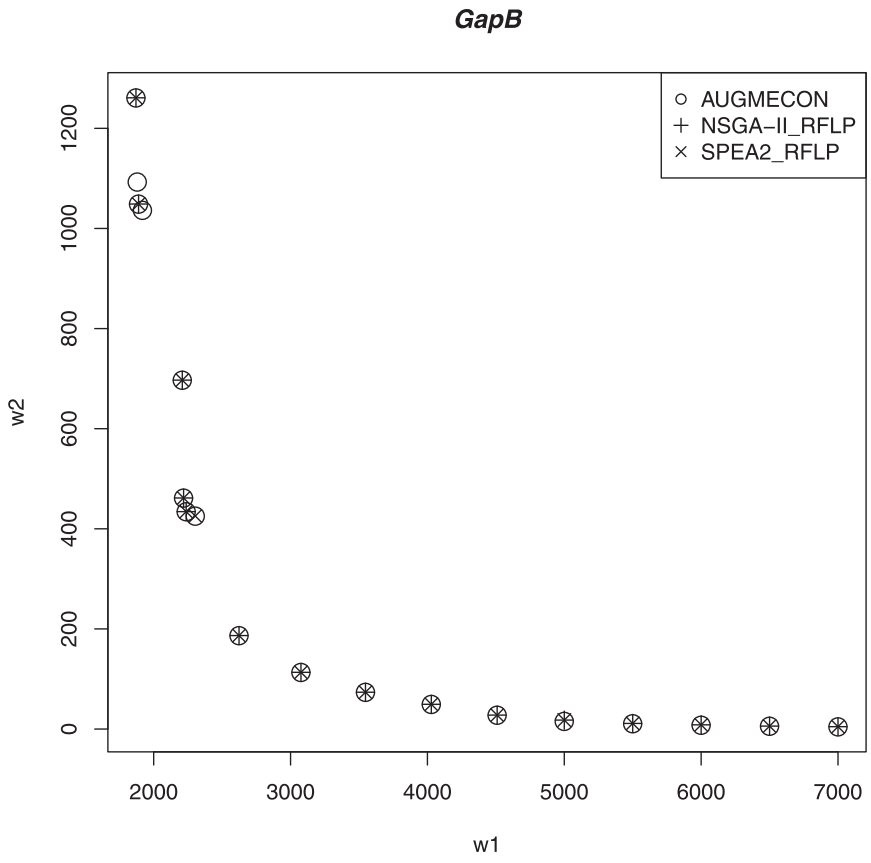

Fig. 2. Exact and approximated Pareto frontiers for GapB331.

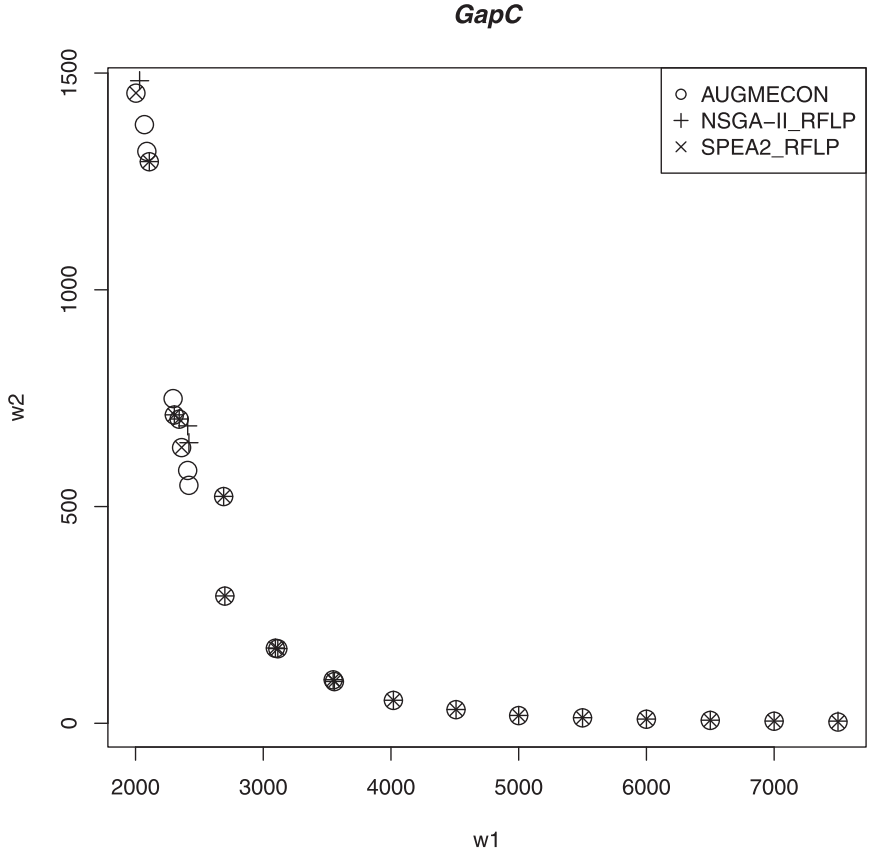

Fig. 3. Exact and approximated Pareto frontiers for GapC333.

Please cite this article as: J. Alcaraz, M. Landete and J.F. Monge et al., Multi-objective evolutionary algorithms for a reliability location problem, European Journal of Operational Research, https://doi.org/10.1016/j.ejor.2019.10.043

These figures also illustrate the antagonism between both objectives w_1 ad w_2. The slope of the points with $2000 \leq w_1 \leq 3000$ indicates that small increments in w_1 lead to large decrements in w_2. We can also observe that even in the case where the number of open facilities was fixed, the multi-objective problem would make sense: points with similar values for w_1 would probably have the same number of open plants while their values for w_2 considerably differ.

6. Conclusions

In this paper, we have proposed two new metaheuristic methods to solve the Reliability Fixed-Charge Location Problem where two objectives are considered simultaneously, one based on non-dominated sorting and the other on the idea of Pareto strength. The operators included in the algorithms have been carefully designed, incorporating problem-specific knowledge in order to obtain quality solutions in a reasonable computation time. Some of these operators are shared by both algorithms and others are specific. In order to demonstrate the efficiency of our proposals, we have compared our results with those given by an exact method which has also been implemented, testing the approaches in a standard library of instances through extensive computational experiments. The results presented are highly satisfactory, given that our approaches find many of the points belonging to the Pareto frontier and provide a very good approximation of the exact Pareto frontier, given by a *HVR* value very close to 100%, employing computation times near 150 times lower than the exact approach. Comparing the two metaheuristics, NSGA-II_RFLP obtains better results than SPEA2_RFLP for all the metrics considered in the instances evaluated and it is clearly more efficient.

There are several interesting lines for future research given, that multi-objective problems in location are very common in practice. The proposed metaheuristics could be adapted to solve different bi-objective location problems or they could even be extended in order to manage problems with more than two objectives, problems in which the difficulty of obtaining an exact Pareto frontier increases considerably.

Acknowledgments

We are grateful for the financial support from the Spanish Ministry for Economy, Industry and Competitiveness (Ministerio de Economía, Industria y Competitividad), the State Research Agency (Agencia Estatal de Investigacion) and the European Regional Development Fund (Fondo Europeo de Desarrollo Regional) under Grants MTM2016-79765-P (AEI/FEDER, UE) and MTM-2015-68097(P) (MINECO/FEDER).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2019.10.043.

References

Alcaraz, J., Landete, M., & Monge, J. F. (2012). Design and analysis of hybrid metaheuristics for the reliability p-median problem. *European Journal of Operational Research, 222*, 54–64.

Alcaraz, J., Landete, M., Monge, J. F., & Sainz-Pardo, J. L. (2015). Strengthening the reliability fixed-charge location model using clique constraints. *Computers & Operations Research, 60*(1), 14–26.

Antunes, C. H., & Dolores, M. (2016). Sensor location in water distribution networks to detect contamination events: A multiobjective approach based on NSGA-II. Proceedings of the 2016 IEEE congress on evolutionary computation, (pp. 1093–1099).

Baev, I., & Rajaraman, R. (2001). Approximation algorithms for data placement in arbitrary networks. Proceedings of the 12th Annual ACM-SIAM symposium on discrete algorithms, (pp. 661–670).

Bashiri, M., & Rezanezhad, M. (2015). A reliable multi-objective p-hub covering location problem considering of hubs capabilities. *International Journal of Engineering Transactions B: Applications, 28*(5), 717–729.

Chen, W., Wiecek, M. M., & Zhang, J. (1999). Quality utility - a compromise programming approach to robust design. *Journal of Mechanical Design, 121*, 179–187.

Coello, C. A., Lamont, G. B., & Veldhuizen, D. A. V. (2007). Evolutionary algorithms for solving multi-objective problems. In D. E. Goldberg, & J. R. Koza (Eds.), *Genetic and evolutionary computation series*. Springer.

Das, I., & Dennis, J. (1998). Normal-boundary intersection: A new method for generating Pareto optimal points in multicriteria optimization problems. *SIAM Journal on Optimization, 8*(3), 631–657.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Eghbali, M., Abedzadeh, M., & Setak, M. (2014). Multi-objective reliable hub covering location considering customer convenience using NSGA-II. *International Journal of System Assurance Engineering and Management, 5*(3), 450–460.

Ehrgott, M., & Gandibleux, X. (2004). Approximate solution methods for multiobjective caombinatorial optimization. *TOP, 12*(1), 85–89.

Ehrgott, M., & Ruzika, S. (2008). Improved ε-constraint method for multiobjective programming. *Journal of Optimization, Theory and Applications, 138*, 375–396.

Farahani, R. Z., Seifi, M. S., & Asgari, N. (2010). Multiple criteria facility location problems: A survey. *Applied Mathematical Modelling, 34*(7), 1689–1709.

Fernandez, E., & Landete, M. (2015). Fixed charge facility location problems. *Location Science*. Springer.

Hancock, B. J., & Mattson, C. A. (2013). The smart normal constraint method for directly generating a smart pareto set. *Structural and Multidisciplinary Optimization, 48*(4), 763–775.

Jalalia, S., Seifbarghyb, M., Sadeghia, J., & Ahmadic, S. (2016). Optimizing a bi-objective reliable facility location problem with adapted stochastic measures using tuned-parameter multi-objective algorithms. *Knowledge-Based Systems, 95*, 45–57.

Konak, A., & Smith, E. (2011). Efficient optimization of reliable two-node connected networks: A biobjective approach. *INFORMS Journal on Computing, 23*(3), 430–445.

Levialdi, S. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems Man and Cybernetics, 3*, 296–297.

Li, X., & Ouyang, Y. (2011). Reliable sensor deployment for network traffic surveillance. *Transportation Research Part B: Methodology, 45*, 218–231.

Marler, R. T., & Arora, J. S. (2010). The weighted sum method for multi-objective optimization: new insights. *Structural Multidisciplinary Optimization, 41*, 853–862.

Mavrotas, G. (2009). Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation, 213*(2), 455–465.

Mavrotas, G., & FLorios, K. (2013). An improved version of the augmented e-constraint method (AUGMECON2) for finding the exact Pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation, 219*, 9652–9669.

Messac, A. (1996). Physical programming: Effective optimization for computational design. *AIAA Journal, 34*(1), 149–158.

Messac, A., Ismail-Yahaya, A., & Mattson, C. A. (2003). The normalized normal constraint method for generating the Pareto frontier. *Structural and Multidisciplinary Optimization, 25*(2), 86–98.

Messac, A., & Mattson, C. A. (2004). Normal constraint method with guarantee of even representation of complete pareto frontier. *AIAA Journal, 42*(10), 2101–2111.

Moalic, L., Caminada, A., & Lamrous, S. (2013). A fast local search approach for multiobjective problems. In G. Nicosia, & P. Panos (Eds.), *Learning and intelligent optimization* (pp. 294–298). Berlin Heidelberg: Springer.

Nickel, S., Puerto, J., & Rodríguez-Chía, A. M. (2015). *Location problems with multiple criteria*. Location Science.

Pardalos, P., Žilinskas, A., & Žilinskas, J. (2017). *Non-convex multi-objective optimization* p. 123. Berlin: Springer.

Sainz-Pardo, J. L., Alcaraz, J., Landete, M., & Monge, J. F. (2017). On relaxing the integrality of the allocation variables of the reliability fixed-charge location problem. *Journal of Global Optimization, 67*, 787–804.

Shen, Z. J. M. (2011). The reliable facility location problem formulations, heuristics, and approximation algorithms. *INFORMS Journal on Computing, 23*(3), 470–482.

Snyder, L. V., & Daskin, M. S. (2005). Reliability models for facility location: The expected failure cost case. *Transportation Science, 39*(3). 440–416

Zadech, L. A. (1963). Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control, 8*(1), 59–60.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation, 1*(1), 32–49.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation, 8*(2), 173–195.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). Spea2: Improving the strength Pareto evolutionary algorithm. In K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, & T. Fogarty (Eds.), *Evolutionary methods for design optimization and control with applications to industrial problems* (pp. 95–100). International Center for Numerical Methods in Engineering, Athens, Greece.