IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 16, 2020, accepted January 18, 2021, date of publication January 28, 2021, date of current version February 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055295

Partial Evaluation and Efficient Discarding for the Maximal Covering Location Problem

CYNTHIA PORRAS[1], JENNY FAJARDO[2], ALEJANDRO ROSETE[1], AND ANTONIO D. MASEGOSA[2,3]

[1]Facultad de Informática, Departamento de Inteligencia Artificial e Infraestructura de Sistemas Informáticos, Universidad Tecnológica de La Habana "José Antonio Echeverría," La Habana 11500, Cuba
[2]Deusto Institute of Technology, Universidad de Deusto, 48007 Bilbao, Spain
[3]IKERBASQUE, Basque Foundation for Science, 48011 Bilbao, Spain

Corresponding author: Jenny Fajardo (fajardo.jenny@deusto.es)

The work of Jenny Fajardo and Antonio D. Masegosa were supported in part by the European Union's Horizon 2020 Research and Innovation Programmes under Grant 769142 and Grant 815069, and in part by the Spanish Ministry of Science and Innovation through research project under Grant PID2019-109393RA-I00.

ABSTRACT The maximal covering location problem attempts to locate a limited number of facilities in order to maximize the coverage over a set of demand nodes. This problem is NP-Hard and it has been often addressed by using metaheuristics, where the execution time directly depends on the number of evaluations of the objective function. In this article, the principles of efficient discarding and partial evaluation are applied to obtain more efficient versions of the objective function of this problem, i.e. not-approximate surrogate objective functions. An experimental study is presented to compare the surrogate functions in terms of number of distance comparisons and runtime. The results show that (on average) the best surrogate function is more than 5 times faster than the original function in general, and more than 8 times faster in the largest instances. This proposal allows for a more efficient metaheuristic solution based on swap operators.

INDEX TERMS Efficient discarding, partial evaluation, maximal covering location problem, objective function.

I. INTRODUCTION

Metaheuristics have proved to be flexible methods than can be used to find good solutions for optimization problems [1]. However, they must perform a large number of the objective function evaluations which implies a high computational cost. One way to reduce this computational cost is by using surrogate functions [2], which may be either approximate or exact versions of the original objective function of the problem. Two approaches to obtain equivalent (not approximate) versions of objective function are: partial evaluation [3] and efficient discarding [4]. Partial evaluation takes into account the influence of the operators on the quality of the solutions. It only evaluates the part of the objective function that was modified by the action of the operators [3]. On the other hand, efficient discarding consists in stopping the evaluation when it is known that the remaining part of the solution will not affect the final value of the objective function. Both approaches have been successfully used in several problems, such as: aggregation of rankings [3], graph drawing [5], task planning [4] and binary problems [6]. These examples indicate the growing interest in the objective function cost reduction in metaheuristics.

The maximal covering location problem (MCLP) is an optimization problem that attempts to find the best location of several facilities (p) in order to maximize the coverage of a set of demand nodes [7]. This problem has many applications such as: location of patrol stations and police distribution [8], ambulances organization and distribution [9], location of fire stations [10] and location of cameras in urban traffic networks [11]. Unfortunately, MCLP is NP-Hard [12], and the solution space of the problem is given by the binomial coefficient $\binom{|J|}{p}$ [13], when $|J|$ is the total of facilities and p is the number of facilities to be located.

Many papers have focused on solving MCLP, both accurately and approximately [14], thus producing efficient applications of this problem in real-world contexts. Murray [14] summarized the main research areas concerning the problem. CPLEX [15], Gurobi [16], LINGO [17], among others, are the main tools that have been used to solve the MCLP

The associate editor coordinating the review of this manuscript and approving it for publication was Xujie Li.

in an exact way for small instances. Heuristics (such as Greedy-Add [18] and Lagrangian Relaxation [13]), and metaheuristics (such as Genetic Algorithms [19], GRASP [20], Simulated Annealing [21] and Tabu Search [22]) and Hybrid approaches [23] have been used to solve the MCLP in an approximate manner. These approximate methods have obtained good results for large-scale instances.

When a MCLP instance is solved in a specific domain by the previous algorithms, the quality of the best solution obtained is related to the time and computational resources that are available for execution. If some additional time is allowed, the search may continue, and an improved solution may be obtained. This is particularly important when the MCLP is applied to situations where the response time is critical, such as emergency attention [9] or surveillance [8].

Previous papers have focused on the efficient exploration of the search space, but the way to implement the objective function has not yet been analyzed in detail. As the objective function of MCLP has quadratic complexity [13] (as detailed below), the impact of using surrogate functions in order to reduce the computational cost in MCLP is an interesting aspect to be studied.

This article is focused on obtaining more efficient versions of the objective function of MCLP, i.e. not approximate surrogate functions. In particular, the impact of partial evaluation and efficient discarding on the objective function of MCLP is studied. The experimental study presented demonstrates the notable reduction obtained in the execution time. The article is organized as follows: Section II presents the MCLP, emphasizing the computational cost of the objective function. Section III proposes the inclusion of efficient discarding and partial evaluation in order to obtain surrogate functions. Section IV presents an experimental study with 170 instances that shows how the best surrogate functions can significantly reduce the execution time. Section V presents an analysis of the complexity of the best surrogate functions focusing on when their efficiency is affected. Section VI presents and discuss the importance of the results obtained, as well as the limitations and implications. Finally, the conclusions of the article are presented.

II. THE MAXIMAL COVERING LOCATION PROBLEM

The maximal covering location problem (MCLP) was originally introduced by Church and ReVelle in [7]. The objective of MCLP is to find the best location of a limited number of facilities in order to maximize the coverage over a set of the demand nodes. MCLP considers a discrete set of facilities and demand nodes. Each demand node has an associated demand, representing its level of importance. The mathematical model of MCLP is presented below.

The parameters and variables that define the MCLP are:

- i, I: the index and set of demand nodes.
- j, J: the index and set of facility sites.
- a_i: population or demand of the node i.
- d_{ij}: the shortest distance (or time) from demand node i to the facility j.
- p: number of facilities to be located.
- S: coverage distance. It is the minimum distance (or time) required between a demand node and facility in order to be considered to be covered.
- $N_i : \{j | d_{ij} \leq S\}$ set of potential facilities that can cover the demand generated in i.
- $X_j : \{0, 1\}$ a binary variable which equals 1 if the facility is placed at node j, 0 otherwise.
- $Y_i : \{0, 1\}$ a binary variable which equals 1 if the demand node i is covered by one or more facilities located within a distance of S, 0 otherwise.

The objective function:

$$\text{Maximize } Z = \sum_{i \in I} a_i Y_i \tag{1}$$

Subject to:

$$Y_i \leq \sum_{j \in N_i} X_j, i \in I \tag{2}$$

$$\sum_{j \in J} X_j = p \tag{3}$$

The objective function (1) aims to maximize the sum of the demand of the covered nodes. Constraint (2) implies that a demand node i is covered only when there is one or more open facilities that belongs to the set N_i. Constraint (3) implies that the number of facilities to be located is p. Fig. 1 shows a solution for a MCLP instance with $p = 4$, which is a binary vector where each one (1) indicates an open facility, and 0 otherwise. It can be see that open facilities are located in nodes 2, 3, 4 and 9.

	Facilities								
j	node 1	node 2	node 3	node 4	node 5	node 6	node 7	node 8	node 9
X_j	0	1	1	1	0	0	0	0	1

FIGURE 1. Representation of a solution, where locations 2, 3, 4 and 9 are open ($p = 4$).

Fig. 2 presents a graphical representation of the solution of Fig. 1, where each rectangle represents a facility and each circle represents a demand. It is worth noting that in order to evaluate each solution, it is necessary to determine the Y_i values for each demand node, which are not inputs to the problem. Each value Y_i is assigned by determining whether the demand nodes are covered by one or more open facilities ($X_j = 1$), that are located at a distance equal to or less than the defined radius S. Then, all Y_i values must be multiplied by the a_i values and summed up to obtain Z. This implies that $(I * p)$ operations are needed [13] in terms of distance comparisons between demand nodes (I) and open facilities (p).

MCLP is NP-Hard [12] and the solution space is given by the binomial coefficient $\binom{J}{p}$ [13]. To solve MCLP exactly, it is necessary to evaluate the solution space given by $\binom{J}{p}$ solutions. As $(I * p)$ is the number of distance comparisons (operations) that are necessary to evaluate one solution, the complexity of solving MCLP (exactly) is $(I * p * \binom{J}{p})$ [13].

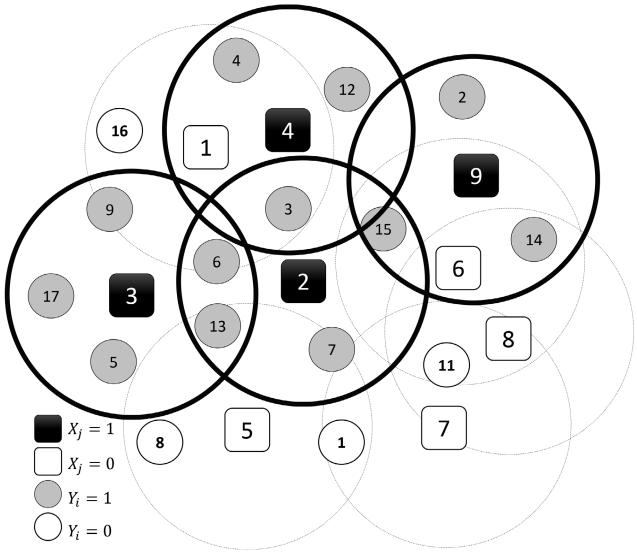

FIGURE 2. Graphic representation of the solution presented in Fig. 1.

A direct way to perform the evaluation of the objective function is: for each facility where $X_j = 1$, the I demand nodes must be visited, in order to obtain the Y_i values. For each demand node, if $Y_i = 0$ and $d_{ij} \leq S$, then $Y_i = 1$ and a_i is added to Z. As this comparison is made between each facility and each demand node, the complexity of the calculation of Z is $\mathcal{O}(I * p)$ [13], in terms of the number of distance comparisons against the reference radius S. Algorithm 1 shows the procedure for calculating the objective function of MCLP. In the rest of the article, this form of computing the objective function will be called the ''original'' objective function.

Algorithm 1 Original Objective Function

```
1:  procedure objFunctionMCLP(J, I, p)
2:      Z = 0
3:      j = 0
4:      repeat
5:          if X_j = 1 then
6:              i = 0
7:              repeat
8:                  if d_ij ≤ S and Y_i = 0 then
9:                      Z = Z + a_i
10:                     Y_i = 1
11:                 i++
12:             until i = |I|, all I nodes are visited
13:         j++
14:     until j = |J|, all J facilities are visited
15:     return Z
```

Daskin [13] showed that if $I = 100$, $J = 50$ and $p = 1$, only 5000 operations are needed, but if $p = 10$, $1.027 * 10^{13}$ operations are needed. Assuming that a computer can perform 10^7 operations per second then 12 days are needed to solve this instance of the problem, making any cost reducing option very attractive.

In order to solve the MCLP for large instances, the use of heuristic algorithms has gained interest [14]. Feasible initial solutions are often constructed and operators that maintain the feasibility are used when trying to solve the MCLP with metaheuristics.

In this sense, as the number of open facilities (p) must be kept constant, one of the most used operators is the 2-swap operator [24]. This operator closes a facility (setting its value X_j to 0) and opens another (setting its value X_j to 1). The selection of the facility to close and to open is at random. Fig. 3 shows an example of the application of this operator with $J = 4$ and $p = 2$. In this example, the facility $j = 1$ is closed and the facility $j = 2$ is opened.

Before applying the operator

j	1	2	3	4
X_j	**1**	**0**	1	0

After applying the operator

j	1	2	3	4
X_j	**0**	**1**	1	0

FIGURE 3. Example 2-swap operator application.

Several heuristics (such as Greedy-Add [18] and Lagrangian Relaxation [13]) and metaheuristics (such as Genetic Algorithms [19], Simulated Annealing [21] and Tabu Search [22]) have been used to solve MCLP in an approximate way. They have obtained good results for large instances. To guide the search in the approximate algorithms, the objective function of the MCLP must be calculated many times. Finding a more efficient way to calculate it would have a very positive impact on the use and application of these algorithms. Indeed, if the time taken to evaluate each solution is reduced, it could be used to obtain either a better solution in the same amount of time or a similar solution in less time.

III. SURROGATE FUNCTIONS BASED ON PARTIAL EVALUATION AND EFFICIENT DISCARDING

In this section, two different ways of implementing the objective function of MCLP by using efficient discarding and partial evaluation are presented. Both approaches can be applied independently, which implies that there are three versions of surrogate functions with respect to Algorithm 1 (the original objective function). The use of efficient discarding, partial evaluation and combination of both approaches are considered. Table 1 shows the new symbols used in the algorithms pseudo-code presented in this section

As in the previous section, the complexity analysis will take into account the number of distance comparisons between a facility and a demand node, which verifies the coverage restriction. The time complexity of the introduced surrogate functions is $\mathcal{O}(I * p)$ in the worst case scenario,

TABLE 1. Notations Used in the Algorithms.

Symbol	Definition
A	Set of uncovered demand nodes
G	Set of covered demand nodes by the closed facility
H	Set of covered demand nodes by the opened facility
Z_{last}	Value of the objective function in the previous solution
C	Index of the closed facility
O	Index of the opened facility

i.e. the same as the original objective function. However, in Section IV, the advantages of the proposals in terms of their average case time complexity are showed.

A. EFFICIENT DISCARDING

The principle of efficient discarding [4] is similar to some classic algorithms such as the Alpha-Beta algorithm [25] or the Branch and Bound approach in Operations Research [26]. The idea is to avoid unnecessary calculations that will not affect the result.

Applying this to Algorithm 1, the time wasted by visiting all the demand nodes is remarkable when considering points where $Y_i = 1$ because they were covered by previously analyzed open facilities. In addition, in the case where all nodes have already been covered or all open facilities have been analyzed, it would be unnecessary to continue visiting the remaining facilities because they would not change the final value of Z. This means that the rest of the comparisons can be discarded.

Algorithm 2 Objective Function With Efficient Discarding

```
1:  procedure objFunctionMCLPDisc(J, I, p)
2:      A = I
3:      Z = 0
4:      p' = 0
5:      j = 0
6:      repeat
7:          if X_j = 1 then
8:              i = 0
9:              p' + +
10:             repeat
11:                 if d_ij ≤ S then
12:                     Z = Z + a_i
13:                     Y_i = 1
14:                     A = A − {i}
15:                 i + +
16:             until i = |A| or |A| = ∅
17:         j + +
18:     until p' = p or |A| = ∅
19:     return Z
```

Algorithm 2 describes the variant of the objective function with efficient discarding. The process is based on controlling the set A of the demand nodes that have not yet been covered and the number of open facilities already visited p'. Based on these variables, the evaluation is stopped conveniently.

In many cases, this new function may allow a reduction in terms of distance comparisons in the objective function. However, the worst case scenario for this algorithm occurs when all demand nodes are covered by a single facility and this facility is visited last. In this case, the time complexity is $\mathcal{O}(I * p)$. This implies that, in the worst case scenario, this version is similar to the original. The average case will be analyzed in the experiments conducted in Section IV. This function can be used for any search algorithm because it does not depend on the way it is used (metaheuristic and operator) to obtain the solution to be evaluated by the objective function.

B. PARTIAL EVALUATION

Partial evaluation is a principle that only evaluates the part of the objective function that was modified by the action of the operators [3]. This implies that partial evaluation depends on the operator used to obtain the solution to be evaluated. Based on the usual 2-swap operator (explained in Section II and in Figure 3), partial evaluation can save the positions of the facilities that were changed; thus, it is only necessary to evaluate the demand nodes that were affected by the change.

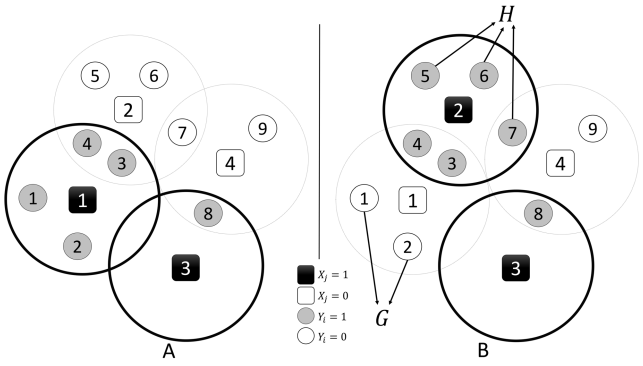

FIGURE 4. Elements of the objective function of the MCLP with partial evaluation.

For example, the demand nodes that are covered by the facilities that remain open ($j = 3$ in Fig. 4) will remain covered. In general, the objective function can only vary in the coverage of the demand nodes that were covered by the facility that was closed and the demand nodes that will be covered by the facility that was opened. The demand nodes covered by both affected facilities (those changed by the operator) and the demand nodes that are not covered by any of them do not change their coverage. Consequently, they do not affect the evaluation of the new solution.

To implement partial evaluation, it is necessary to save some information for each solution: the position of the facility that was closed; the one that is now opened; and in addition, for each state a relation of $I - nodes$ with its Y_i value. This increases the use of internal memory and computational resources, and the next section will show how that can significantly reduce the execution time. With this information, partial evaluation will only evaluate the part of objective

function Z that was modified when applying the mutation operator. Based on the positions that underwent changes, two sets of demand nodes G and H are created ($G \subseteq I$, $H \subseteq I$, $G \cap H = \emptyset$):

- G includes the nodes covered by the closed facility that are not within the radius S of the new open facility. The sum of the demand associated with the nodes of the set G is subtracted from the previous Z value, leaving Z_G. It is worth noting that this demand will not necessarily remain uncovered since they could be covered by one of the other facilities that remain open.
- H includes the nodes that were not previously covered and that now can be covered by the new opened facility. The demand value of the set H is added to Z_G, and the value Z_{G+H} is obtained.
- Finally, the nodes in set G are evaluated $p-1$ times in order to check their coverage: each a_i is added if they are covered ($d_{ij} \leq S$) by the facilities where $X_j = 1$ in the new state. This provides a new covered demand $Z_{G'}$. The final value of Z is calculated as the sum $Z_{G+H} + Z_{G'}$.

Fig. 4 shows an illustrative example of the different elements involved in the evaluation of the objective function of MCLP. Taking into account the state of Fig. 4, the facility $j = 1$ and $j = 3$ are initially open (Fig. 4 (**A**)), and the demand nodes 1, 2, 3, 4 and 8 were covered. When applying the 2-swap operator, the facility $j = 1$ has been closed and the facility $j = 2$ has been opened (Fig. 4 (**B**)). Thus, the demand nodes 1 and 2 belong to set G and the demand nodes 5, 6 and 7 belong to set H.

At this point it can be concluded that demand nodes of sets G and H may modify the value of Z. The demand nodes 3 and 4 do not modify the value of Z, since in previous state they were covered by facility $j = 1$ and they are still covered by facility $j = 2$. As the demand nodes 8 and 9 are completely independent of the analyzed facilities, they do not affect the value of Z.

Algorithm 3 presents the procedure the objective function of the MCLP with partial evaluation. It has three new inputs: Z_{last} (value of the objective function of the previous state), C (position of the facility that was closed) and O (position of the facility that was opened). It is worth noting that when the first solution is generated there is no previous state, thus Algorithm 1 or Algorithm 2 must be used to compute the objective function.

The worst case scenario for Algorithm 3 occurs when the $G \cup H = I$. This occurs when all demand nodes are covered by the facility that was closed or by the facility that was opened. Thus, the complexity of this function is $\mathcal{O}(I * p)$ in the worst case scenario. The average case is analyzed in the experiments in Section IV. It worth noting that the set H is not used in the computation of the objective function in the Algorithm 3, but this step was included in order to be able to conduct the analysis conducted in the following sections.

This surrogate objective function can be used both in exact and approximate search algorithms. As requirement, to use this function, it is necessary to guide the search by simple swaps, i.e. by swapping one open facility with another closed facility as shown in Figure 3 (2-swap operator) and to save the $j-$ value of these facilities. As similar idea may be applied to several swaps, if the information related to the demand covered in each solution is updated. Also, it is necessary to save the value of the objective function of the solution before the application of the operator. However, this surrogate function can not be applied to metaheuristics based on crossover mechanism, e.g. genetic algorithms.

Algorithm 3 Objective Function With Partial Evaluation

```
1:  procedure objFunctionMCLPartialEval(J, I, p, Z_last, C, O)
2:      Z = Z_last
3:      G = ∅
4:      i = 0
5:      repeat
6:          if d_iC ≤ S and d_iO > S then
7:              Z− = a_i
8:              G = G + {i}
9:              Y_i = 0
10:         else if d_iC > S and d_iO ≤ S then
11:             if Y_i = 0 then
12:                 H = H + {i}
13:                 Z+ = a_i
14:                 Y_i = 1
15:         i++
16:     until i = |I|, all the nodes in I are visited
17:     k = 0
18:     j = 0
19:     repeat
20:         if X_j = 1 and j! = O then
21:             repeat
22:                 if d_G[k]j ≤ S then
23:                     Z+ = a_G[k]
24:                     Y_G[k] = 1
25:                 k++
26:             until k = |G|, all the nodes in G are visited
27:         j++
28:     until j = |J|, all the facilities in J are visited
29:     return Z
```

C. COMBINATION OF EFFICIENT DISCARDING AND PARTIAL EVALUATION

It is also possible to combine partial evaluation with efficient discarding in order to reduce the number of distance comparisons. In this case, it may not be necessary to take into account all open facilities, since all the demand nodes of set G may have been covered, nor may it be necessary to continue visiting more facilities if all the open facilities have already been visited. Again, it is worth noting that when the first solution is generated there is no previous state, thus Algorithm 1 or Algorithm 2 must be used to compute the objective function.

Algorithm 4 Objective Function With Partial Evaluation and Efficient Discarding

```
1:  procedure objFunctionMCLPEvalPartDisc(J, I, p, Z_last, C, O)
2:      Z = Z_last
3:      G = ∅
4:      i = 0
5:      repeat
6:          if d_iC ≤ S and d_iO > S then
7:              Z− = a_i
8:              G = G + {i}
9:              Y_i = 0
10:         else if d_iC > S and d_iO ≤ S then
11:             if Y_i = 0 then
12:                 H = H + {i}
13:                 Z+ = a_i
14:                 Y_i = 1
15:         i++
16:     until i = |I|, all the nodes in I are visited
17:     p' = 1, the new opened facility
18:     j = 0
19:     repeat
20:         if X_j = 1 and j! = O then
21:             p'++
22:             k = 0
23:             repeat
24:                 if d_G[k]j ≤ S then
25:                     Z+ = a_G[k]
26:                     Y_G[k] = 1
27:                     G = G−{k}
28:                 k++
29:             until k = |G| or G = ∅
30:         j++
31:     until p' = p or G = ∅
32:     return Z
```

Algorithm 4 shows the procedure of the objective function of MCLP with partial evaluation and efficient discarding. The requirements are the same than the function with only partial evaluation.

The worst case scenario for this algorithm is the combination of the worst cases scenarios of the previous algorithms. Similarly, the worst case scenario occurs when $G \cup H = I$ and the facility that can cover them is the last one. Thus, the time complexity in the worst case scenario is $\mathcal{O}(I * p)$. The average case is analyzed in the experiments in Section IV.

IV. EXPERIMENTAL STUDY

This section presents an experimental study that aims to evaluate the performance of the proposed versions of the objective function, in terms of the number of distance comparisons between facilities and demand nodes, and the overall computational time. Consequently, this experiment provides an average case analysis of the different versions of the objective function.

First, the data for the experimentation should be defined. In order to design the set of instances, three data set of the Traveling Salesman Problem, available in [27] (ar6723, pm4951 and fi10639), were selected. Different instances were derived as result of the combinations of I values between 1500 and 9000, and J values between 80 and 850. It has selected p values by using 30, 40, 50, 60 and 70% of each J value.

Two values of S (small and large radius) were used. In each instance, the small values of S can obtain a solution with a coverage about 60 to 95% for the highest value of p. The large value of S permits a solution with coverage of about 90 to 99% to be obtained for the highest value of p. In summary, 170 instances for the experiments are used, obtained from the previous combinations of factors. The description of the instances is shown in Table 2.

TABLE 2. Description of the 170 Instances Used in the Experiments.

Data set	I	J	p					S small	S large
pm8079	1500	80	24	32	40	48	56	156	556
fi10639	1800	100	30	40	50	60	70	102	402
ar9152	2000	150	45	60	75	90	105	1068	1568
pm8079	2500	200	60	80	100	120	140	136	336
fi10639	3000	250	75	100	125	150	175	182	300
ar9152	3500	300	90	120	150	180	210	1068	1468
pm8079	4000	350	105	140	175	210	245	50	150
pm8079	4500	400	120	160	200	240	280	50	150
ar9152	5000	450	135	180	225	270	315	650	1050
ar9152	5500	500	150	200	250	300	350	700	900
ar9152	6000	550	165	220	275	330	385	450	650
fi10639	6500	600	180	240	300	360	420	250	400
fi10639	7000	650	195	260	325	390	455	190	350
fi10639	7500	700	210	280	350	420	490	102	302
fi10639	8000	750	225	300	375	450	525	280	380
fi10639	8500	800	240	320	400	480	560	200	320
fi10639	9000	850	255	340	425	510	595	182	402

The previous data were generated based on a semi-random style that is commonly used in the context of MCLP [28] and without the intention of causing bias in any of the functions. The results obtained are then analyzed, based on the characteristics of the instances used.

In the experimental process, a Local search algorithm is used in order to obtain several solutions to the MCLP instances and to be able to evaluate them with the objective functions. Local search methods [1] have been used before in the context of MCLP [14]. The simplest version can be based on a random initial solution and the application of mutation operators, accepting a new solution when it is better than the current one.

For this reason, Hill-climbing as the metaheuristic was selected to conduct the experiments with the mutation operator described in Figure 3. This algorithm does not have any other operation that could interfere with the performance of the functions, such as auxiliary lists or selection/replacement operators. However, it should be noted that the time of the proposed surrogate functions does not depend on the

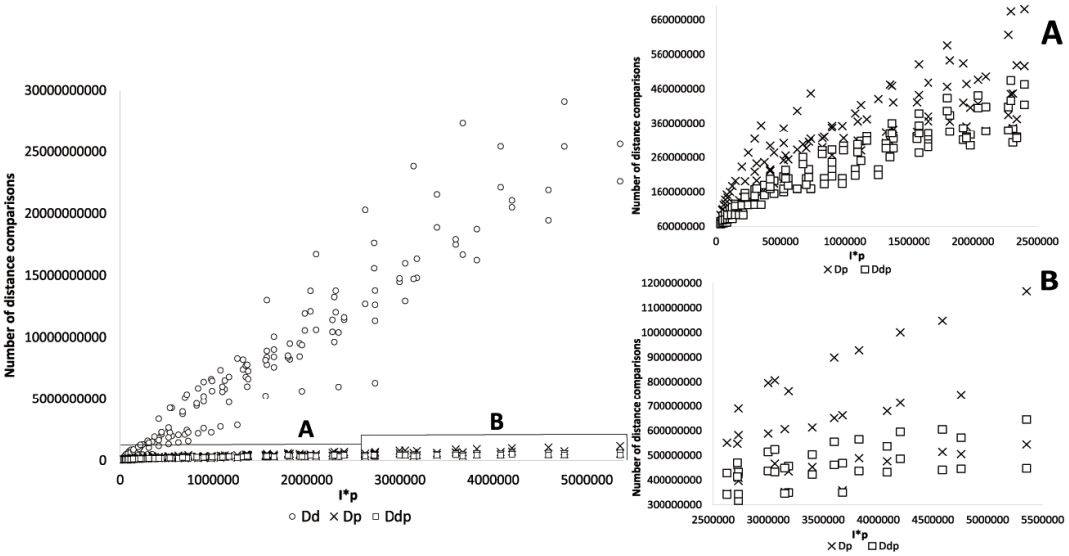

FIGURE 5. Average of distance comparisons.

metaheuristic but on the operator used. Thus, it could be used in other metaheuristics based on mutations such as Evolution Strategies, Simulated Annealing or Tabu search. The Hill-climbing was executed 30 times with 10000 evaluations of each the objective function in independent form. With 30 executions and 10000 evaluations for each execution, the bias that the randomness can produce in the results is reduced, thus obtaining an average behavior.

Finally, experimental conditions were established. The model was implemented in Java by using BiCIAM class library [29] for the implementation of optimization problems and the use of metaheuristic algorithms. To establish a fair comparison in terms of implementation in Java, the following aspects were taken into consideration. First, the original function and discarding function only need a temporary list of covered and uncovered nodes, which is left unreferenced (Java terms) upon completion of the calculation. Also, in the functions with partial evaluation it was necessary to have (for each solution) a list of covered demand nodes. The experiments were conducted on a personal computer with 8GB of RAM and an Intel (R) Core (TM) i7 CPU at 2.4 GHz.

In the following subsections, the results that correspond to the number of distance comparisons that are computed in each version of the objective function will be shown, as this is the fundamental operation that it intends to reduce. Then, the average execution time is analyzed. The results will be shown using graphs, where the behaviors of the similar functions are given in more detail.

A. REDUCTION OF THE FUNDAMENTAL OPERATION

In this section, the number of distance comparisons computed by each algorithm (objective function) is analyzed. In the original function, this operation is performed for the p facilities with respect to all the demand nodes, so the distance is calculated $p * I$ times [13].

Fig. 5 shows the average of the total number of distance comparisons for all iterations (10000) in the 30 executions (on the $y - axis$) in relation to the corresponding $p * I$ value (on the $x - axis$). The discarding function (Dd, circles), the function with partial evaluation (Dp, crosses) and with both aspects (Ddp, squares) are shown. The representation of distance comparisons in the original objective function is obviated, since it is always equal to $(p*I)*10000$ (iterations).

It can be observed that Dd made more distance comparisons than Dp and Ddp for all the instances. Section A and B of Fig. 5 show, in more detail, the behavior of the Dp and Ddp functions. They differ by about 20% on average (Dp performs more distance comparisons).

Fig. 6 shows the percent of savings that could be obtained in terms of distance comparisons. This percent was calculated as the average of distance comparisons for each surrogate function with respect to the original function.

A considerable decrease can be observed in the number of distance comparisons in the surrogate functions with respect to the original objective function. The functions with partial evaluation imply greater reductions in the number of distance comparisons for larger instances. Section A and B of Fig. 6 show that Dp and Ddp compute less than 30% of the number of distance comparisons made in the original function in all cases. Also, Ddp and Dp perform less than 10% of the number of distance comparisons for the largest instances, i.e. with partial evaluation, the saving of comparisons increases for the largest instances. On average, Dd performs approximately 50% of the distance comparisons performed by the original function, while Dp (and Ddp) only made 5% (4%, respectively).

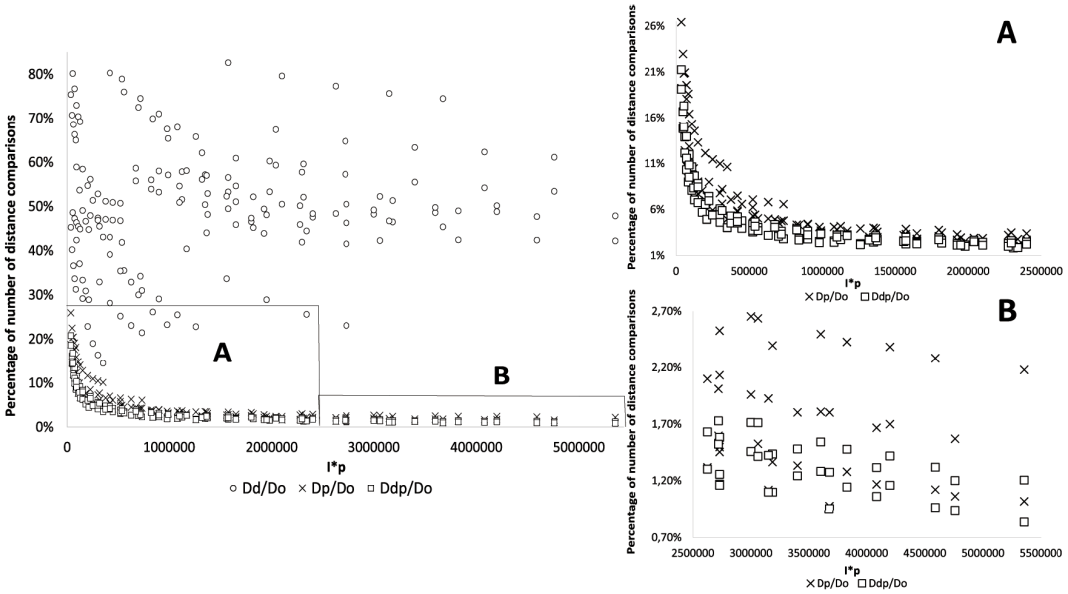

FIGURE 6. Proportion of distance comparisons with respect to the original function.

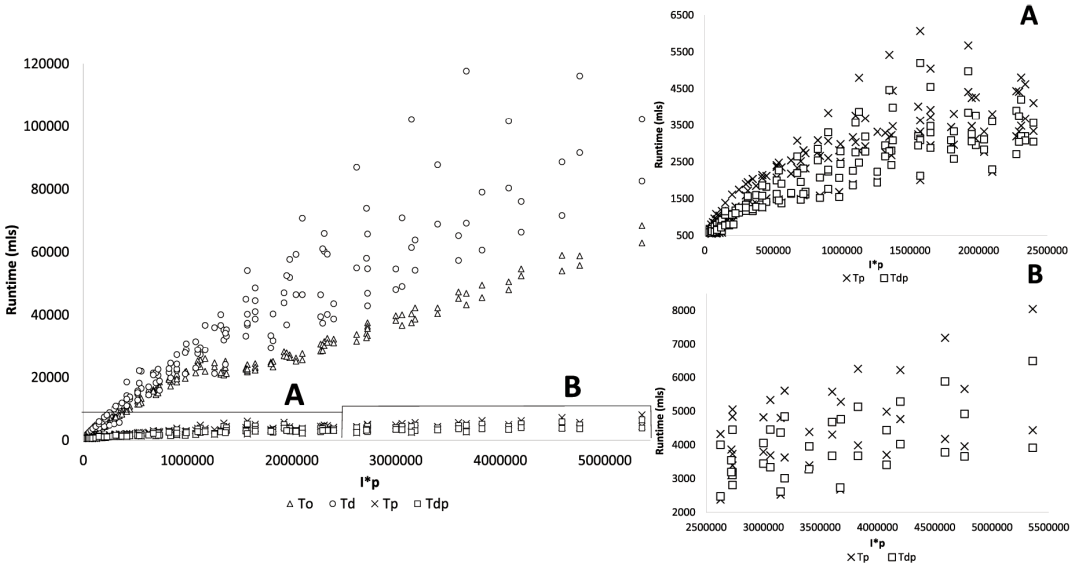

FIGURE 7. Average runtime.

In summary, the three variants of surrogate functions allow a remarkable reduction in the number of distance comparisons, but the best function is the combination of partial evaluation and efficient discarding. Later, in Subsection IV-C, statistical tests are performed in order to confirm the advantage of the proposals (see Table 3).

B. TIME REDUCTION WITH SURROGATE FUNCTIONS

As indicated above, the surrogate functions allow a remarkable reduction in the number of distance comparisons. The reduction in time is now specifically analyzed taking into account that in order to obtain these reductions other operations had to be carried out as part of the objective function. Fig. 7 presents the average runtime (in milliseconds) of the original objective function (To, triangles), the surrogate function with efficient discarding (Td, circles), with partial evaluation (Tp, crosses) and the surrogate function with both aspects (Tdp, squares). Each time value (on the $y - axis$) is averaged over 30 executions of each problem and it is associated with the $I * p$ value (on the $x - axis$).

It can be seen that, for most instances, the time taken by Td is greater than that of the original objective function. This happens because very few nodes are discarded. In terms of the implementation of the efficient discarding, set A

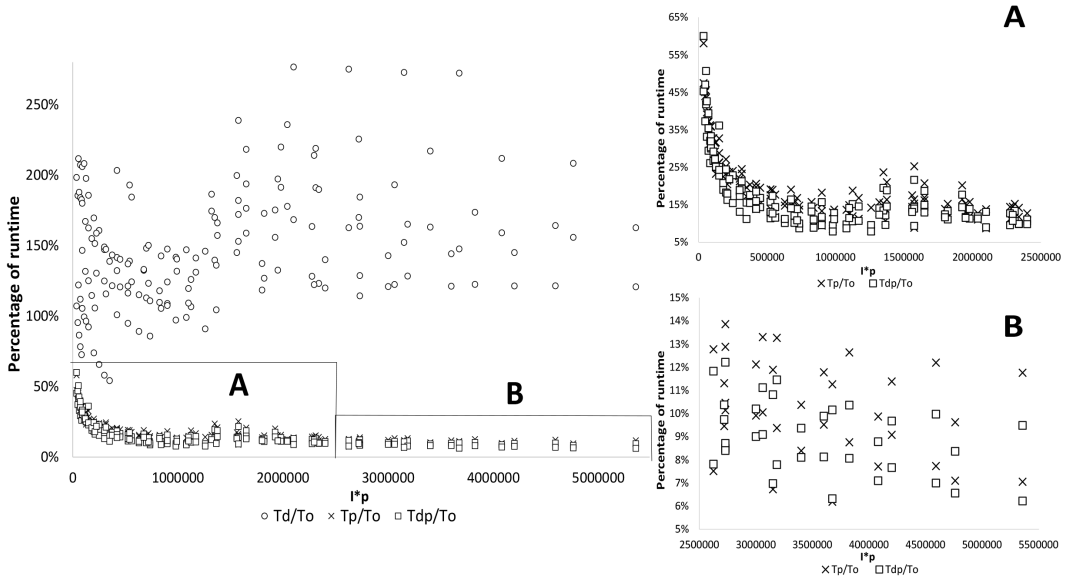

FIGURE 8. Proportion of runtime with respect to the original function.

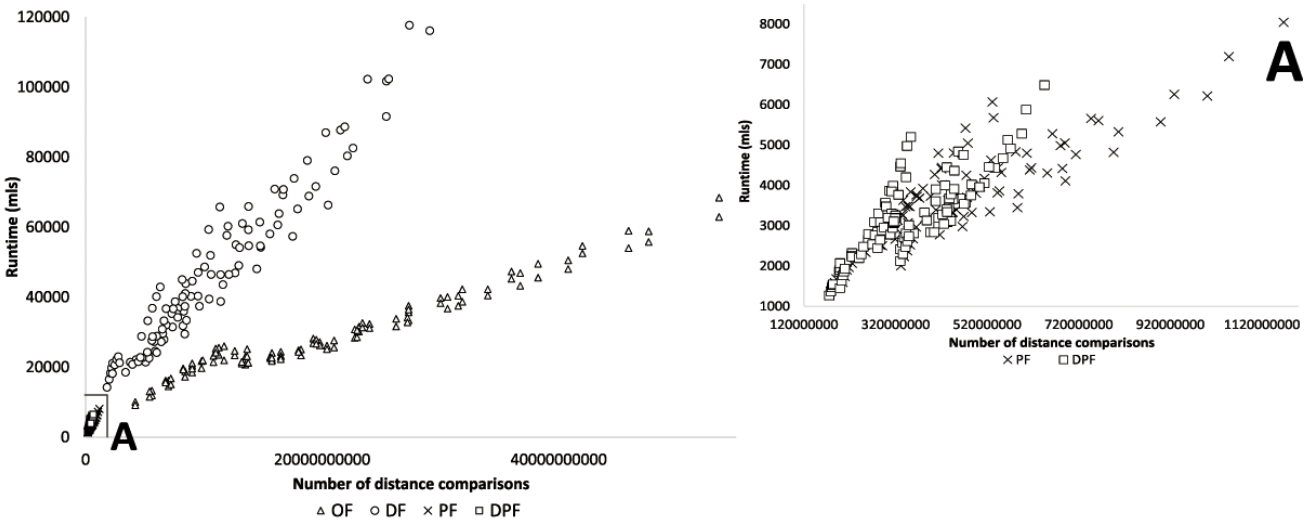

FIGURE 9. Proportion time/distance comparisons for $I \geq 4000$.

(see Algorithm 2) is a Java linked list, and when $Y_i = 1$ this demand node must be removed from the list. A linked list was used for A because deletion is more efficient in this data structure than it is in array lists. Although the search for the nodes is slower in linked lists, it is not as slow as a deletion in an array list [30], making it more efficient to use a linked list to implement efficient discarding. However, none of these extra computations are compensated with a significant reduction in terms of distance comparisons. Maybe other data structures should be studied in the future.

Fig. 8 shows the percentage of the average time taken by surrogate functions with respect to the average time taken by the original function. Again, it can be observed that Td (in general terms) is slower than To. Indeed, Td was only better than To in some small instances.

However, the functions with partial evaluation (Tp and Tdp) reduce the execution time in all instances. In the smallest instances (that is, $I < 3000$) Tp (respectively, Tdp) uses on average 32% (respectively, 29%) of the original runtime. In the largest instances (that is, $I \geq 3000$), it is worth noting that Tp (respectively, Tdp) uses on average 14% (respectively 12%), of the original time. Indeed, both functions with partial evaluation are more than 5 times faster in general. In the largest instances, Tp (respectively, Tdp) is more than 7 (respectively, 8) times faster than the original function.

The execution time, as observed, was remarkably reduced using the functions with partial evaluation, and this reduction is even more noticeable for the largest instances. Fig. 9 shows the runtime (on the $y-axis$) and number of distance comparisons (on the $x-axis$) for the instances with $I \geq 4000$. All the objective functions analyzed are detailed below: original function (OF, triangles), surrogate function based on efficient discarding (DF, circles), with partial evaluation (PF, crosses), and with both aspect (DPF, squares).

It can be observed the great difference that exists between the PF and DPF with respect to OF in terms of time and number of distance comparisons. The most remarkable reductions are observed for larger instances. Fig. 9 (**A**) shows that the DPF performs a smaller number of distance comparisons than PF and executes in less time. In spite of the advantage of DPF with respect to PF, this advantage is not as significant as the advantage of DPF and PF with respect to OF and DF. This means that partial evaluation (used in PF and DPF) is the most important aspect that can influence the reduction of the execution time.

C. STATISTICAL TESTS

In order to detect significant differences among the objective functions, the Friedman test [31] and the post-hoc Holm [32], Finner [33] and Li [34] were applied, for the number of distance comparisons and the execution time, with $\alpha = 0.05$. Keel [35] was used to conduct the tests. Table 3 shows the Friedman ranking and the results of the post-hoc Holm, Finner and Li. The $p-$ values coincide in all three post-hoc with respect to α for each analysis (distance comparisons and runtime). Thus, this consensus in only one column (Post-hoc) is shown.

TABLE 3. Friedman Ranking and Post-Hoc Results for Distance Comparisons and Execution Time. The "∗" Symbol Indicates the Control Method and "<" Means That the Control Method was Significantly Better Than the Others.

	Distance	comparisons	Runtime	
Objective functions	Ranking	Post-hoc	Ranking	Post-hoc
Ddp	1	∗	1.14	∗
Dp	2	<	1.86	<
Dd	3	<	3.90	<
Do	4	<	3.10	<

Table 3 shows that there were significant differences between the functions in both aspects (comparisons and runtime), since $p - value = 0$. The best function in both cases was Ddp and the slowest was Dd, for the reasons explained above.

In order to confirm the direct relationship between the execution time and the number of distance comparisons, the Pearson correlation coefficients [36] were obtained: between Do and To is 0.98, between Dd and Td is 0.98, between Dp and Tp is 0.94, and between Ddp and Tdp is 0.94. This proves that it was a good decision to focus on the reduction of the number of distance comparisons in order to reduce the execution time of the objective function of MCLP.

V. ANALYSIS OF THE BEST OBJECTIVE FUNCTION

In the previous sections, it is shown that the best surrogate objective function is based on the combination of the principles of partial evaluation and efficient discarding. Given its small difference with respect to the function that only has partial evaluation, the principle of partial evaluation has more influence on the reduction of distance comparisons and execution time. For this reason, it considers that is important the analyzes of the complexity of the objective function of MCLP when the partial evaluation is applied.

It is previously defined that the worst case scenario for partial evaluation occurs when $G \cup H = I$. In this section, three scenarios where the function with partial evaluation made more distance comparisons than the original function are explained. Fig. 10 shows the conditions where the first scenario occurs.

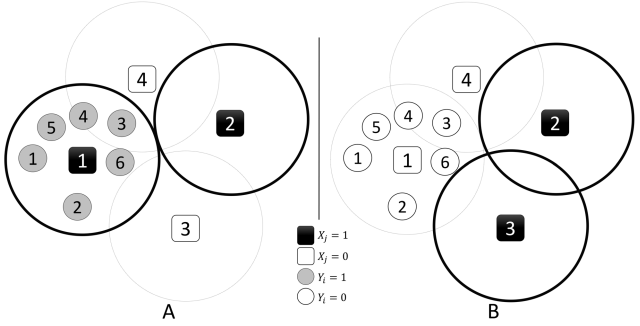

FIGURE 10. Worst case: Scenario 1.

At first (Fig. 10 (**A**)), all demand nodes are only covered by the open facility, e.g. $j = 1$ and $X_1 = 1$. After applying the 2-swap operator (Fig. 10 (**B**)), facility $j = 3$ is now open and $j = 1$ is closed, the demand nodes of set $G = \{1, 2, 3, 4, 5, 6\}$ and set $H = \emptyset$. Then $G \cup H = \{1, 2, 3, 4, 5, 6\} = I$. Hence, the worst case scenario in partial evaluation has occurred, since the I demand nodes should be evaluated $p - times$ (for each open facility).

In order to better explain the situation, an example in Fig. 10 is given:

- In Scenario 1, $I = \{1, 2, 3, 4, 5, 6\}$, $J = \{1, 2, 3, 4\}$ and $p = 2$, the original objective function (see Algorithm 1) will perform $I * p$ distance comparisons. If $I = 6$ and $p = 2$ then $6 * 2 = 12$ distance comparisons.
- In the partial evaluation, as result of the first cycle, if all the nodes belong to set G, it will perform $2 * 6 = 12$ distance comparisons (see Algorithm 3 and 4), where 2 represents the number of distance comparisons needed to know if each demand node belongs to G, and 6 is $|I|$.
- In the second cycle $1 * 6 = 6$ comparisons must be added where 6 is $|G|$ and 1 represents the other facilities that have not been analyzed $(p - 1)$ and could cover the demand nodes of set G.
- Finally, a total of $12 + 6 = 18$ distance comparisons would be made. Thus, the function with partial evaluation is worse than the original function in Scenario 1.

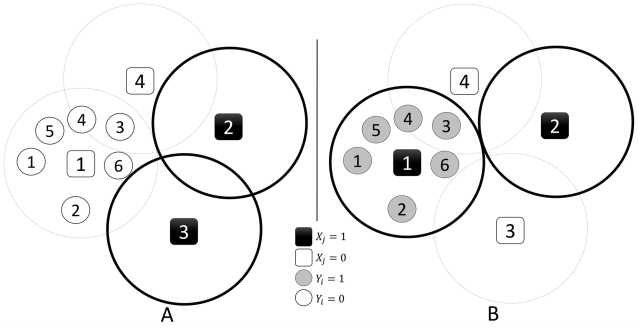

FIGURE 11. Worst case: Scenario 2.

The second scenario is described in Fig. 11. This scenario may take place after the occurrence of scenario 1 (Fig. 11 (**A**)). Now, Fig. 11 (**B**) shows that the facility $j = 3$ was closed and the facility $j = 1$ was opened, then the set $G = \emptyset$, the set $H = \{1, 2, 3, 4, 5, 6\}$ and $G \cup H = I$.

An example in Fig. 11 is given:

- As in the previous example, $6 * 2 = 12$ distance comparisons will be performed using the original function.
- In the first cycle of the partial evaluation, as all the nodes belong to set H, $4 * 6 = 24$ distance comparisons will be performed (see Algorithm 3 and 4). The first two comparisons are carried out in order to verify if they are in set G, and the other two comparisons are necessary in order to verify if they are in set H.
- In the second cycle $1*0 = 0$ comparisons must be added, since the set $G = \emptyset$ and the 1 represent the other facilities that have not been analyzed $(p - 1)$.
- Finally, a total of $24 + 0 = 24$ distance comparisons would be made. Thus, the function with partial evaluation is worse than the original function in Scenario 2.

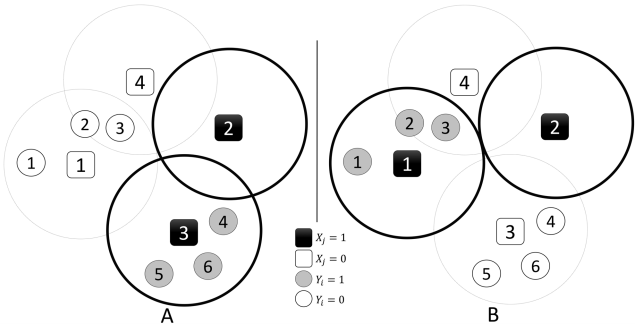

FIGURE 12. Worst case: Scenario 3.

The third scenario occurs when the $I - nodes$ belong the sets G and H, where $G \cap H = \{\emptyset\}$. Initially, Fig. 12 (**A**) shows that the facility $j = 3$ is opened. Then, this facility is closed and the facility $j = 1$ is now opened (Fig. 12 (**B**)). Finally, all demand nodes are covered by the facilities that were modified. There is also an example to explain this scenario, see Fig. 12:

- As in the previous examples, $6 * 2 = 12$ distance comparisons will be carried out, using the original function.
- In the first cycle, the sets $G = \{4, 5, 6\}$ and $H = \{1, 2, 3\}$ will be constructed. Then $3 * 2 = 6$ comparisons will be made for the set G, where 3 is $|G|$ and 2 is the number of distance comparisons required to determine set G. Then, $3*4 = 12$ comparisons for set H are made, where 4 is the number of distance comparisons in order to determine set H, and 3 is $|H|$. Finally, a total of $6 + 12 = 18$ distance comparisons are obtained in the first cycle.
- In the second cycle $1*3 = 3$ comparisons must be added, since set G has three demand nodes, and 1 is the number of facilities that have not been analyzed $(p - 1)$.
- Finally, a total of $18 + 3 = 21$ distance comparisons would be made. Thus, the function with partial evaluation is worse than the original function in Scenario 3.

In the scenarios explained above, the function with partial evaluation performs more distance comparisons than the original function. This demonstrates that the number of comparisons depends on the characteristics of each instance and on the facilities that are modified by the mutation.

In the experiments presented in the Section IV, the sizes of G and H for each iteration and instances were saved, and the average size of G and H were calculated over 10000 iterations and 30 executions. Fig. 13 shows the percentage that represents the size of set G and H with respect to the size of set I for each instance (on average).

It can be observed that the behavior of the percentage that represents G with respect to I is similar to the behavior of the number of distance comparisons and execution time, i.e. it decreases for the largest instances. Set H has almost no influence on the number of distance comparisons, since its size (on average) was smaller than the size of set G.

It is worth noting that the worst case scenario (where the new objective function with partial evaluation does not improve the original function) only occurs when $G \cup H = I$. According to Figure 13, the average size of G is only over 5% in the smallest instances (it never reaches 12%), while the average size of H is smaller than 1% of I in all instances.

This implies that the extreme cases where the proposals do not speed-up the objective function are very unlikely. In addition, the situation where the combined sizes of G and H is over 5% of I only occurs for several small instances (and this value is less than 2% in the largest instances). In the greatest instances (where it is more important to reduce the computational cost) the new proposals are useful with a very high probability. With this result it can be concluded that the worst case scenario of the surrogate objective function with partial evaluation occurs if $G = I$, but this almost never happens.

VI. DISCUSSION

Based on the results presented previously, several points can be highlighted. In general terms (and particularly with respect to the configuration of the parameters), it is worth mentioning the following:

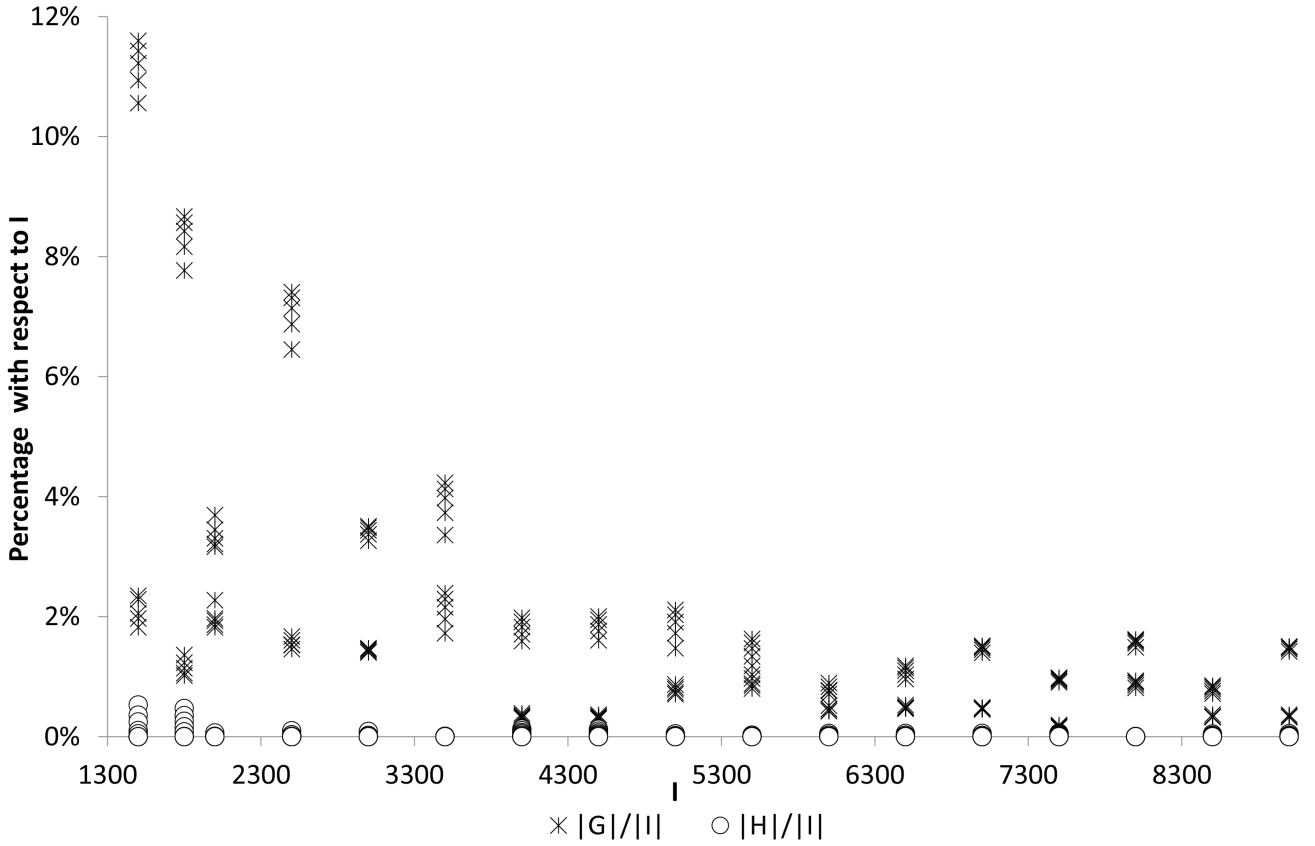

FIGURE 13. Percent of set *G* and *H* with respect to set *I*, average case.

- The beneficial effect of partial evaluation is noteworthy, i.e. the functions with partial evaluation are the fastest. This is due to the reduction that this technique implies in terms of distance comparisons. Indeed, the direct effect of the distance comparisons in the execution time was confirmed.
- In general, the functions without partial evaluation are slower. The maximum (worst) average time taken to obtain a solution was obtained by the discarding function in the instance where $I = 7500$ and $p = 70\%$ with the small radius S (about 2 minutes on average to obtain a solution). However, with the large radius the instance that took the longest time on average was the one where $I = 8500$ and $p = 70\%$(1.5 minutes). The original function obtained the largest average execution time in the largest instance ($I = 9000$ and $p = 70\%$) with both coverage radius S, where it took about one minute.
- In the case of the functions with partial evaluation, the execution time was substantially shorter. Indeed, their longest execution time occurs in the instance with the largest coverage radius $S, I = 9000$ and $p = 70\%$ (about 6 to 8 seconds). With the small coverage radius S, the longest execution time is about 4 to 5 seconds in the instance $I = 8000$ and $p = 70\%$.
- On average, the large radius S increases the execution time of the original function To and the partial evaluation functions Tp and Tdp (by approximately 13%, 43% and 25%, respectively), with respect to the runtime with the small S. However, the large radius S decreases the execution time of the efficient discarding function by approximately 80%. This is due to the fact that a large radius implies that set G and H are larger, thus the execution time increases for the original functions and for the function based on partial evaluation. In the case of the efficient discarding function when the radius S is large the function discards more because it is easier to cover a demand, thus the execution time decreases.
- The analysis conducted in Section III concludes that the new proposals does not change the time complexity of the MCLP fitness objective evaluation in the worst case scenario. However, the analysis conducted in Section V clearly states when this happens, resolving that this extreme situation would be very unlikely. Indeed, based on the results presented in Section V, the introduced versions of the objective function are faster than the original function with a very high probability. This is why the functions with partial evaluation perform better on average.

- The proposed surrogate functions are particularly beneficial in the largest instances where it is more important to reduce the computational cost, since these instances are the ones that demand more computational effort. The beneficial effect of this speed-up in almost all objective function evaluations is demonstrated in the average case analysis conducted in Section IV. The average time of the best objective function (Tdp) is about 12% of the original time for the largest instances ($I \geq 3000$). Indeed, the best surrogate functions are generally more than 5 times faster.
- In general, the results obtained are aligned with the objectives of the work. The proposed techniques (efficient discarding and partial evaluation) allow to reduce the number of distance comparisons that are computed in the objective function of MCLP. This reduction in the comparison produces a direct effect on the reduction of the execution time (as it was confirmed with the computation of the correlation coefficient between distance comparisons and execution time). In the case of the functions with partial evaluation this implies a significant reduction in the execution time.

Taken into account that the best objective function uses the principle of partial evaluation, several limitations and opportunities can be considered:

- Some limitations of the use of the function with partial evaluation are detailed in Section V. These cases may occur when two facilities are enough to cover all the demand nodes, or when there are many facilities that cover almost no nodes. Thus, it would be more convenient to use the original objective function to solve this type of instances. However, it is important to note that this type of instances are very unlikely and they are easy to solve. The main interest of the proposals presented in this article is to solve large instances.
- Another limitation of the presented functions with partial evaluation is their dependency on the use of the 2-swap operator. However, there are some opportunities to extend their use. Indeed, if the mutation operator makes several swaps, the proposal may be adapted in order to be able to still reduce the computational cost. For example, if the operator makes k swaps, partial evaluation may be used repeatedly. Thus, the evaluation of these k-swaps may take approximately $k * 12\%$ of the time of the original function (for large instances). For example, for $k = 4$ the partial evaluation may use 48% of the original time. An experimental evaluation of this situation may be studied in future research.
- The advantages of the results presented in this article may be considered limited when the metaheuristics are based on several mutation operators, i.e. when swap is just one of the available mutations. For example, this happens if there are three mutation operators (2-swap being one of them) and each operator is used with the same probability. In this example, the application of the partial evaluation presented here would allow that 1 out of 3 evaluations to be done faster. Thus, an approximately equal cost would be maintained in 67% of the evaluations but the other 33% of the evaluations may be done in only 12% of the time (on average, for the largest instances). This would imply that the global time would be approximately $67\% + 12\% * 33\% = 67\% + 4\% = 71\%$, thus saving about 30% of the overall execution time.

Therefore, the use of the proposed objective function may represent a benefit in the execution time of the search algorithms. In order to understand the importance of the results presented in this article, it is important to realize what is implied when a metaheuristic-based solution for MCLP uses the new objective functions in real and experimental applications. Two situations are given to exemplify this implication:

- To solve instances of MCLP, if a given execution time is allowed for a metaheuristic, the speed-up induced by the surrogate functions presented in this article allows an extra search to be conducted within the same execution time. Conversely, in the same time the original function can only perform about 20% of the iterations that the proposed surrogate functions can do. Therefore, the results obtained may improve if the proposed functions are used.
- Consider a MCLP application where the desired placement of the drones is obtained in 10 seconds in a drone-based surveillance system. The speed-up induced by the surrogate functions presented in this article allows the desired placement to be obtained in less than 2 seconds (for the case of the largest instances that are more than 5 times faster than the original function). Thus, the proposals may allow 8 extra seconds of surveillance that may be critical in several situations such as identifying suspects before they escape from a crime scene.

Thus, the surrogate functions presented in this article imply a beneficial effect in terms of efficiency that may be translated either into better solutions archived in the same amount of time or into similar solutions achieved in less time.

VII. CONCLUSION

In this article, three surrogate functions were proposed for the MCLP, using a combination of partial evaluation and efficient discarding. As a result of the experiments carried out, the following statements were verified:

- The three surrogate functions reduce the number of distance comparisons with respect to the original objective function, saving (on average) approximately 50% of the comparisons when the efficient discarding is used (Dd), 95% for partial evaluation (Dp), and 96% for the combination of both (Ddp).
- The surrogate function with efficient discarding alone (Td) only produces a small reduction in the execution time (on average), and this advantage only occurs in 17 out of 170 instances.

- The functions with partial evaluation (Tp and Tdp) allow for more than 80% of the original time (To) to be saved, i.e. they are more than 5 times faster.
- The statistical tests proved that there are significant differences between the functions. The fastest function (which is also the function that performs the smallest number of distance comparisons) is the surrogate function that combines efficient discarding with partial evaluation, followed by the function that only includes partial evaluation.
- The advantage of both functions with partial evaluation are quite remarkable when compared to the others. The principle of partial evaluation in the objective function of the MCLP produces a very positive influence in the reduction of distance comparisons and time of execution (5 to 8 times faster than the original function).
- The worst case scenario for the objective function with partial evaluation occurs when $G = I$, but this happens very infrequently.
- The objective function with partial evaluation and efficient discarding can be used in metaheuristics that use the 2-swap operator in several or all iterations.
- To use partial evaluation, it is necessary to update the demand nodes covered by a state after each swap (mutation) is applied. The partial evaluation presented in this article cannot be used if the new states (solutions) have been obtained by using crossover or other operators.
- The use of the best objective functions proposed in real situations of MCLP permits to obtain solutions of the problem in less time.

As future research, the partial evaluation method for the MCLP can be extended to allow the objective function to be calculated when the swap operator makes more than one change to the current state. Also, the version of the best objective function may be extended to other variants of MCLP, such as multi-period environments, where the changes produced by the operators occur in different periods of time.

REFERENCES

[1] E. G. Talbi, *Metaheuristics: From Design to Implementation*, 1nd ed., vol. 14. Paris, France: Wiley, 2009.

[2] J. Y. J. Chow and A. C. Regan, "A surrogate-based multiobjective Metaheuristic and network degradation simulation model for robust toll pricing," *Optim. Eng.*, vol. 15, no. 1, pp. 137–165, Mar. 2014, doi: 10.1007/s11081-013-9227-5.

[3] J. A. Aledo, J. A. Gámez, and A. Rosete, "Partial evaluation in rank aggregation problems," *Comput. Oper. Res.*, vol. 78, pp. 299–304, Feb. 2017, doi: 10.1016/j.cor.2016.09.013.

[4] O. Avalos-Rosales, F. Angel-Bello, and A. Alvarez, "Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times," *Int. J. Adv. Manuf. Technol.*, vol. 76, nos. 9–12, pp. 1705–1718, Feb. 2015, doi: 10.1007/s00170-014-6390-6.

[5] A. Rosete, A. Ochoa, and M. Sebag, "Efficient-discarding fitness functions," presented at the Genetic Evol. Comput. Conf., Orlando, FL, USA, Jul. 1999. [Online]. Available: https://pdfs.semanticscholar.org/7bd1/237b5cc00b14974562299658102f8b6 0199c.pdf

[6] F. Glover, T. Ye, A. P. Punnen, and G. Kochenberger, "Integrating tabu search and VLSN search to develop enhanced algorithms: A case study using bipartite Boolean quadratic programs," *Eur. J. Oper. Res.*, vol. 241, no. 3, pp. 697–707, Mar. 2015, doi: 10.1016/j.ejor.2014.09.036.

[7] R. Church and C. R. Velle, "The maximal covering location problem," *Papers Regional Sci.*, vol. 32, no. 1, pp. 101–118, Jan. 2005, doi: 10.1111/j.1435-5597.1974.tb00902.x.

[8] K. M. Curtin, K. Hayslett-McCall, and F. Qiu, "Determining optimal police patrol areas with maximal covering and backup covering location models," *Netw. Spatial Econ.*, vol. 10, no. 1, pp. 125–145, Mar. 2010, doi: 10.1007/s11067-007-9035-6.

[9] C. S. Lim, R. Mamat, and T. Braunl, "Impact of ambulance dispatch policies on performance of emergency medical services," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 624–632, Jun. 2011, doi: 10.1109/TITS.2010.2101063.

[10] S. Bao, N. Xiao, Z. Lai, H. Zhang, and C. Kim, "Optimizing watchtower locations for forest fire monitoring using location models," *Fire Saf. J.*, vol. 71, pp. 100–109, Jan. 2015, doi: 10.1016/j.firesaf.2014.11.016.

[11] P. Dell'Olmo, N. Ricciardi, and A. Sgalambro, "A multiperiod maximal covering location model for the optimal location of intersection safety cameras on an urban traffic network," *Procedia Social Behav. Sci.*, vol. 108, pp. 106–117, Jan. 2014, doi: 10.1016/j.sbspro.2013.12.824.

[12] N. Megiddo, E. Zemel, and S. L. Hakimi, "The maximum coverage location problem," *SIAM J. Algebr. Discrete Methods*, vol. 4, no. 2, pp. 253–261, Jun. 1983, doi: 10.1137/0604028.

[13] M. S. Daskin, "The maximum covering location model," in *Network and Discrete Location: Models, Algorithms, and Applications*, 2nd ed. Hoboken, NJ, USA: Wiley, 2013, pp. 146–154.

[14] A. T. Murray, "Maximal coverage location problem: Impacts, significance, and evolution," *Int. Regional Sci. Rev.*, vol. 39, no. 1, pp. 5–27, Jan. 2016, doi: 10.1177/0160017615600222.

[15] M. H. F. Zarandi, S. Davari, and S. A. H. Sisakht, "The large-scale dynamic maximal covering location problem," *Math. Comput. Model.*, vol. 57, nos. 3–4, pp. 710–719, Feb. 2013, doi: 10.1016/j.mcm.2012.07.028.

[16] J. North and F. L. Miller, "Facility location using GIS enriched demographic and lifestyle data for a traveling entertainment troupe in Bavaria, Germany," *Decis. Support Syst.*, vol. 99, pp. 30–36, Jul. 2017, doi: 10.1016/j.dss.2017.05.007.

[17] M. F. Youshanlo and R. Sahraeian, "Dynamic multi-objective maximal covering location problem with gradual coverage," in *Enhancing Synergies in a Collaborative Environment* (Lecture Notes in Management and Industrial Engineering). Cham, Switzerland: Springer, pp. 39–47, Feb. 2015, doi: 10.1007/978-3-319-14078-0_5.

[18] S. Davari, M. H. F. Zarandi, and I. B. Turksen, "A greedy variable neighborhood search heuristic for the maximal covering location problem with fuzzy coverage radii," *Knowl.-Based Syst.*, vol. 41, pp. 68–76, Mar. 2013, doi: 10.1016/j.knosys.2012.12.012.

[19] S. Atta, P. R. S. Mahapatra, and A. Mukhopadhyay, "Solving maximal covering location problem using genetic algorithm with local refinement," *Soft Comput.*, vol. 22, no. 12, pp. 3891–3906, Jun. 2018, doi: 10.1007/s00500-017-2598-3.

[20] J. A. Díaz, D. E. Luna, J.-F. Camacho-Vallejo, and M.-S. Casas-Ramírez, "GRASP and hybrid GRASP-tabu heuristics to solve a maximal covering location problem with customer preference ordering," *Expert Syst. Appl.*, vol. 82, pp. 67–76, Oct. 2017, doi: 10.1016/j.eswa.2017.04.002.

[21] M. S. Jabalameli, B. B. Tabrizi, and M. M. Javadi, "A simulated annealing method to solve a generalized maximal covering location problem," *Int. J. Ind. Eng. Computations*, vol. 2, no. 2, pp. 439–448, Apr. 2011, doi: 10.5267/j.ijiec.2011.01.003.

[22] J. Leigh, S. Dunnett, and L. Jackson, "Predictive police patrolling to target hotspots and cover response demand," *Ann. Oper. Res.*, vol. 283, nos. 1–2, pp. 395–410, Dec. 2019, doi: 10.1007/s10479-017-2528-x.

[23] J. F. Calderín, A. D. Masegosa, and D. A. Pelta, "An algorithm portfolio for the dynamic maximal covering location problem," *Memetic Comput.*, vol. 9, no. 2, pp. 141–151, Jun. 2017, doi: 10.1007/s12293-016-0210-5.

[24] S. Basu, M. Sharma, and P. S. Ghosh, "Metaheuristic applications on discrete facility location problems: A survey," *Opsearch*, vol. 52, no. 3, pp. 530–561, Sep. 2015, doi: 10.1007/s12597-014-0190-5.

[25] J. Schaeffer, "The history heuristic and alpha-beta search enhancements in practice," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 11, pp. 1203–1212, Nov. 1989, doi: 10.1109/34.42858.

[26] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 10th ed. New York, NY, USA: McGraw-Hill, 2014, pp. 1–9.

[27] W. Cook, "The traveling salesman problem (TSP)," Dept. Combinatorics Optim., Univ. Waterloo, Waterloo, ON, Canada, Aug. 2019. [Online]. Available: http://www.math.uwaterloo.ca/tsp/index.html
[28] M. H. F. Zarandi, S. Davari, and S. A. H. Sisakht, "The large scale maximal covering location problem," *Scientia Iranica*, vol. 18, no. 6, pp. 1565–1570, Dec. 2011, doi: 10.1016/j.scient.2011.11.008.
[29] J. Fajardo, "Soft computing en problemas de optimización dinámicos," Ph.D. dissertation, Dept. Comput. Sci. Artif. Intell., Univ. Granada, Granada, Spain, 2015.
[30] C. A. Shaffer, *A Practical Introduction to Data Structures and Algorithm Analysis*, 3nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1997, pp. 100–120.
[31] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Stat. Assoc.*, vol. 32, no. 200, pp. 675–701, Dec. 1937, doi: 10.1080/01621459.1937.10503522.
[32] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandin. J. Statist.*, vol. 6, no. 2, pp. 65–70, 1979. [Online]. Available: http://www.jstor.org/stable/4615733
[33] H. Finner, "On a monotonicity problem in step-down multiple test procedures," *J. Amer. Stat. Assoc.*, vol. 88, no. 423, pp. 920–923, Sep. 1993, doi: 10.1080/01621459.1993.10476358.
[34] J. (David) Li, "A two-step rejection procedure for testing multiple hypotheses," *J. Stat. Planning Inference*, vol. 138, no. 6, pp. 1521–1527, Jul. 2008, doi: 10.1016/j.jspi.2007.04.032.
[35] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization," *J Heuristics*, vol. 15, no. 6, p. 617, May 2008, doi: 10.1007/s10732-008-9080-4.
[36] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*, vol. 2. Berlin, Germany: Springer, 2009, pp. 1–4, doi: 10.1007/978-3-642-00296-0_5.

CYNTHIA PORRAS was born in Havana, Cuba, in 1993. She received the B.S. degree in ingeniería informática and the M.S. degree in informatics applied from the Universidad Tecnológica de La Habana "José Antonio Echeverría" (CUJAE), La Habana, Cuba, in 2016 and 2018, respectively. She is currently pursuing the Ph.D. degree in ciencias técnicas with CUJAE. Since 2016, she has been a Researcher and also a Professor with CUJAE. She has publications in several international conferences such as ICOR 2016, ICOR 2017, FUZZ-IEEE 2018, ESCIM 2019, and IPMU 2020. Her research interests include the optimization and location problems and their techniques of solution like metaheuristics. She has received several Cuban awards in the National Computing Contest, since 2016.

JENNY FAJARDO received the degree in computer engineering and the master's degree in applied informatics from the University of CUJAE (Universidad Tecnológica de La Habana "José Antonio Echeverría"), Cuba, in 2008 and 2010, respectively, and the Ph.D. degree in information and communication technologies in Granada, in 2015. Since 2015, she has been working in both teaching and research. She has more than 10 years' experience teaching both undergraduate and graduate lessons at CUJAE. Since 2019, she has also been a Researcher with the Deusto Institute of Technology, University of Deusto, Bilbao, Spain. In research, she has focused on the field of artificial intelligence, the main research topics are focused on intelligent systems based on soft computing techniques for decision and optimization problems. Her current research interests include artificial intelligence, intelligent systems, soft computing, hybrid metaheuristics, machine learning, and intelligent transportation systems.

ALEJANDRO ROSETE was born in Havana, Cuba, in 1970. He received the B.S. degree in ingeniería informática and the Ph.D. degree from the Universidad Tecnológica de La Habana "José Antonio Echeverría" (CUJAE), La Habana, Cuba, in 1993 and 2000, respectively. Since 1993, he has been a Professor with CUJAE. He has authored of articles in several journals such as *Information Sciences*, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Expert Systems With Applications*, *Decision Support Systems*, and *European Journal of Operational Research*. His research interests include artificial intelligence, optimization with metaheuristics, and soft computing.

ANTONIO D. MASEGOSA received the degree in computer engineering and the Ph.D. degree in computer sciences from the University of Granada, Spain, in 2005 and 2010, respectively. From June 2010 to November 2014, he was a Postdoctoral Researcher with the Research Center for ICT, University of Granada. Since 2014, he has been an IKERBASQUE Researcher with the Deusto Institute of Technology, University of Deusto, Bilbao, Spain. He has published four books, 18 JCR articles, and more than 20 articles in both international and national conferences. He has participated in several research projects at regional, national, and European level. His current research interests include artificial intelligence, intelligent systems, soft computing, hybrid metaheuristics, machine learning, deep learning, intelligent transportation systems, logistic networks, travel demand estimation, and traffic forecasting.