Received: 15 January 2021 | Revised: 24 March 2021 | Accepted: 11 May 2021

DOI: 10.1002/int.22515

RESEARCH ARTICLE

WILEY

# Optimization of electric vehicles charging station deployment by means of evolutionary algorithms

**Alessandro Niccolai** | **Leonardo Bettini** | **Riccardo Zich**

Department of Energy, Politecnico di Milano, Milan, Italy

**Correspondence**
Alessandro Niccolai, Department of Energy, Politecnico di Milano, Via La Masa 34, Milan 20156, Italy.
Email: alessandro.niccolai@polimi.it

**Abstract**

Due to the growing importance of electric vehicles, charging stations (CS) deployment is becoming an important issue in many cities. The aim of this paper is to introduce a novel evolutionary-based approach for solving the CS deployment problem. This study investigates many aspects of the formulation of this approach, such as the design variables selection and the definition of a feasibility function, to improve both effectiveness and flexibility. In particular, the latter is a key factor compared to many other state-of-the-art approaches: in fact, it can be used with most of the available Evolutionary Algorithms (EAs) and can manage different quality-of-service performance parameters. The proposed approach is successfully compared with a greedy optimization on the case study of the City of Milan (Italy) using four different EAs. Two different performance parameters have been defined and used to prove the flexibility of the proposed approach. The results show its very good convergence rate and the quality of the obtained solutions.

All the authors have equally contributed to this study.

**KEYWORDS**
biogeography-based optimization, chargins station deployments, electric vehicles, evolutionary algorithms, genetic algorithm, particle swarm optimization, social network optimization

## 1 | INTRODUCTION

The increasing presence of electric vehicles (EV) opens up new horizons on the city's public infrastructure necessary to support and encourage this phenomenon. Many interests can drive public administrations to promote EV diffusion: in fact, their impact on society is varied and multiple.[1] In the first place, great attention is paid to this theme for environmental and pollution effects. Although the global environmental impact of EV depends very much on the specific mix of sources used for the electricity production, certainly the local impact on cities is beneficial as the possible source of pollution is moved away from city centres. Second, the increase in EV significantly reduces the noise impact within cities compared to internal combustion engine vehicles.[2]

For this reason, many administrations have begun to provide financial incentives for the purchase of EV.[3] Although these incentives are a fundamental aspect, it is nevertheless necessary to consider that in many cities there are limits to the possibility of the citizens to manage their own charging point of electric vehicle. For this reason, the development of a public EV charging infrastructure is crucial aspect. In this context, evolutionary computation (EC) algorithm can provide a decision-making system that can take into account different social requirements, creating an effective and intelligent public infrastructure of charging stations (CS).

Computational intelligence (CI) is growing its importance in many fields, like performing effective forecasts exploiting surrogate models, improving risk management,[4] performing effective data mining,[5] clustering,[6] developing effective controllers,? and many other applications. Among all the CI tools, EC and, thus, evolutionary optimization algorithms (EAs) play a fundamental role in many typical engineering applications, such as optimal design,[7] parameter identification,[8] modelling,[9] portfolio management,[10] decision making,[11] and management systems.[12] Many different Evolutionary Algorithms (EAs) have been developed and are available for solving these problems. The most traditional algorithm is the genetic algorithm (GA)[13] that has been widely applied and developed during years showing very good performance. Another well-established algorithm is the particle swarm optimization (PSO)[14]: several variations of this algorithm have been proposed to avoid the problem of early convergence that highly affects the original implementation of this algorithm. Another important EA is the differential evolution,[15] that has been studied and applied widely in literature.[16,17]

EAs show very good performance in a fast and effective solving of different NP problems. As an example, Kóczy et al.,[18] a new discrete Bacterial Memetic EA in the solution of the Travelling Salesman Problem; the results of this algorithm have been compared with other effective solvers of this problem on 15 benchmarks. Elsisi et al.,[19] implement and test an optimal design for the nonlinear model predictive control based on Opposition-based Learning (OBL). The proposed method results to be very effective in tracking linear and nonlinear trajectories; moreover, it results to be very robust to parameter variations. Massobrio et al.,[20] describe the application of a multiobjective EA for locating roadside infrastructure for vehicular communication networks over realistic urban areas. The proposed multiobjective EA is able to find several trade-off solutions between the installation costs and the Quality-of-Service.

EAs have been exploited also for solving problems related to the optimal deployment of EV CS. For example, Ezhilarasi et al.,[21] propose a cluster-based EA to find a multipath energy-efficient routing protocol to optimize network lifetime and efficiency. This method has been compared against traditional routing protocols on a 200 nodes network. Kuwahara et al.,[22] the authors analyse the problem of finding the optimal way to equip carsharing system with electric chargers for EVs. The proposed method has been tested using practical operational data for Tokyo. Another problem that has been faced with EAs is the deployment of CS for UAV: Hu et al.[23] faced this problem with GAs and Mirzaeinia et al.[24] with Particle Swarm Optimization.

The aim of this paper is to design, implement, and test a new CS deployment approach for urban environment based on EAs: in particular, this study defines design variables, objective function, and a new unfeasible solution management procedure for achieving the maximum flexibility and effectiveness. The proposed method is capable of managing different social requirements by means of the introduction of weighting maps that can drive the optimization solution toward different Quality-of-Service performance indexes. The flexibility of the approach is also given by the possibility to use different EAs within the same framework: in fact, the design variables have been designed to maximize the integration of the proposed approach with most of the state-of-the-art EAs.

The tests are performed with four different optimization algorithms and two weights maps to assess the flexibility of the approach. The performance is proven by comparing the proposed method with a greedy optimization algorithm, that provide the same flexibility, but with lower performance and longer optimization time. Finally, the optimal solutions with the two weight maps are compared to analyse the impact on different social requirements. All these tests are performed on the case study of the city of Milan (Italy).

Compared to deterministic approaches, the one presented here guarantees the possibility of varying performance parameters without having to change the method of solving the problem. Many of the evolutionary-based methods used in literature and described in the following section are based on hypotheses to reduce the research space dimensionality: on the contrary, the approach used in this study is of a general nature, and the discretization choices made can be modified without changing the remaining parts of the approach.

The rest of the paper is structured as follows: in Section 2 the related works are described, analysing similarities and differences with the proposed approach. Section 3 describes the problem of CS deployment, analysing the possible optimization goals, and describing a greedy approach to this problem and the case study used. In Section 4 the proposed method is described and in Section 5 the optimization results with two different objectives are presented. Finally, the conclusions are presented in Section 6.

## 2 | RELATED WORKS

The problem of CS deployment has been addressed in literature under several working hypothesis and conditions, and exploiting different algorithms.

Deterministic approaches, like linear programming, are used in many cases: they provide and guarantee the optimal solution, while they are very often rigid and require a lot of work to be adapted to new conditions.

Xu et al.[25] developed a tailored branch-and-price approach has been developed for determining the optimal deployment of CS for EV. The methods aim to maximize the covered paths knowing the typical flows, being robust to path deviation and nonlinear elastic demand; the proposed systems have been tested on a benchmark 25-node network modelling a real-word

California State road network. As far as the technique is concerned, our method focuses the attention on CS infrastructure in urban environment, where the paths definition is harder and less important for charging purposes.

Linear programming has been used by Bouguerra et al.[26] to find the optimal deployment of CS, limiting the search space under the hypothesis that CSs can be placed only in parking and gas stations. The optimization process takes into account the availability of CS within an acceptable driving range. In all these methods, the search space for the deployment is reduced with an a priori analysis of possible charging zones, as it is done for traditional gas stations. Out approach avoids these hypothesis because the charging process for EV is relatively slow and it is usually performed during long stays, for example, during night or during working time.

EAs can be applied for CS deployment to have a flexible and adaptable tool for different working conditions. Zhang et al.[27] adopt a Multiobjective Particle Swarm Optimization to optimize the deployment of the CS. The available positions for the CSs are obtained overlapping the traffic system and the power system maps and identifying the intersections. Two objectives are here considered: the total cost, that includes both the investment and the expenditure costs, and the average distance between two adjacent CSs. The methods proposed Zhang et al.[27] work well for highways and in nonurban roads.

An approach similar to the one used by Bouguerra et al.[26] has been implemented by Pan et al.[28] where the deployment problem is targeted to taxis. In particular, the authors proposed a multiobjective optimizing model. The problem is faced with a modified GA and models the search space as a graph where the nodes are traffic nodes.

As pointed out before, this method is limited by the definition of the traffic nodes, that requires information that could highly vary in the time horizon of an investment such as the CS deployment.

Luo et al.[29] jointly analysed the design CS and photovoltaic (PV) power plants for creating a CS infrastructure for buses. They consider both the electric infrastructure and the main modes of the bus lines as candidate solutions of the optimization problem. A Surrogate-based Optimization has been implemented here.

This paper combines many targets in the optimization process (charging costs, fees, bus networks), with a search space for the installation locations that is limited by the nodes of the bus network and electrical network. The complexity of this method grows with increasing the number of nodes.

Akbari et al.[30] used GA for the optimal deployment of CS. In this study, the authors defines a priori some settlements in the urban environment. By means of settlements it is intended the locations in which EVs are parked for the most of time. CS positions are defined by GA minimizing the distances to the settlements.

The main limitation of this study is the definition of the settlements: in fact, it requires many statistical information that nowadays are not available due to the relatively low share of EVs.

Finally, in Vazifeh et al.[31] managed again the deployment problem with GA. This study first introduces a discretization of the geographical map for increasing the flexibility of the proposed approach. The paper focus on the minimization of the total driving distance to reach the closest CS. The authors compared the GA with the same greedy algorithm implemented in this study.

With respect to this paper, our work deepens the analysis of the optimization environment, mainly the definition of different performance parameters and the management of unfeasible solutions. Moreover, here the flexibility of the method is assessed by testing different EAs.

## 3 | CS DEPLOYMENT

### 3.1 | Problem description

The problem analysed in this study consists in optimally positioning the CS for EV within the urban area. It can be interpreted as a minimization problem where the design variables are the CS coordinates on the city map and the cost value is inversely proportional to the coverage.[32] It can be formulated as follows:

$$\min_{\underline{x}\in\mathcal{X}} c(\underline{x}, W) \tag{1}$$

where $\underline{x}$ is a vector containing all the CS coordinates, $\mathcal{X}$ is the feasible search space, $W$ is the weight map, and $c(\underline{x}, W)$ is the cost function that should be minimized.

The cost value represents the weighted coverage, and it is function of the design variables vector and of the coverage weighting map $W$: this last is introduced to manage different social requirements of real city management, because it allows to modify the performance parameter and, therefore, to find a different coverage solution.

The simplest $W$ map weights each point of the city by a unitary factor: in this way, the optimization process will distribute the CS to cover as uniformly as possible the city territory. Alternatively, the map can weight the city cells according to the expected distribution of EVs when they need to be charged: in fact, it can take into account the residential population density as well as the density of offices and of jobs.

Both the city and all the weighting maps used in this study have been discretized to manage the computational burden of the optimization process. This assumption does not reduce the generality of the proposed work as the method, presented in the next Section, is able to manage any type of discretization.

### 3.2 | Case study

The case study used in this paper is the city of Milan (Italy). This is a well representative case because, due to historical and hydrogeological reasons, most of the houses do not have a private garage in which they can have their own charging point. Due to this reason, the public infrastructure represents a fundamental point for the diffusion of EV.

Figure 1 shows the city map: the urban area is represented in grey. This picture highlights the jagged city boundary that will highly affect the CS deployment. The white lines in the figure are the main urban streets: they denote that the population density is highly uneven inside the city.

The city map has been discretized with an equally distributed grid in which each cell is 100 m $\times$ 100 m size. This map guarantees a good accuracy of the analysis, in addition the city blocks are usually smaller than a cell and, thus, they do not affect negatively the optimization results. With this discretization, the map is composed of 130 $\times$ 132 cells.

The first weighting map adopted in this study gives an uniform value to all the point of the city. This map is shown in Figure 2A where the city area represented in grey. In this map the white parts are outside the city boundaries: in these cells CS cannot be deployed and the weighted coverage is not here computed. Solving the deployment problem with this map

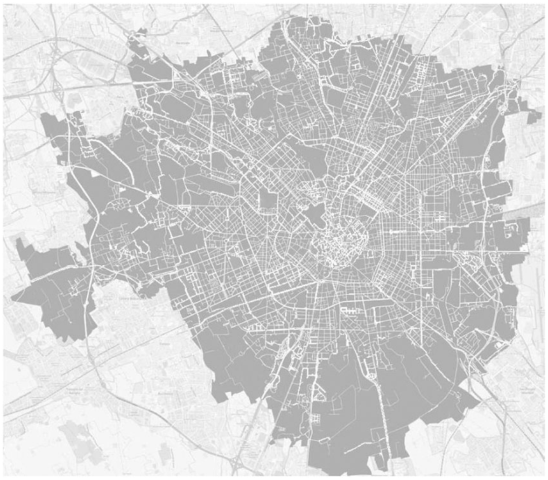


FIGURE 1 City map of Milan (Italy): the urban area is highlighted in the dark grey, while the main streets are indicated in white

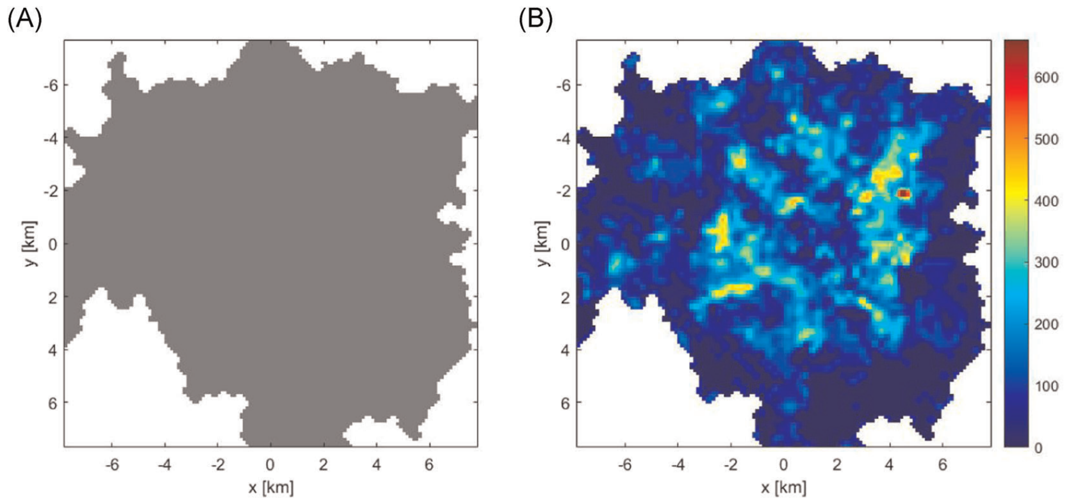


FIGURE 2 Weighting maps implemented: (A) all the points are equally weighted, while in (B) the weights correspond to the city residential density. The colours in (B) are proportional to the estimated number of inhabitants in the discretized cell [Color figure can be viewed at wileyonlinelibrary.com]

corresponds to equally distributing the CS infrastructure within the urban area: on average, all city districts have the same Quality of Service.

The second weighting map implemented is shown in Figure 2B, and it is the city residential density. In the figure, the colours represent the estimated number of inhabitants in the cell. Solving the deployment problem with this map means placing more CS where there are more residents: the average perceived Quality of Service will be higher; however the currently sparsely populated areas risk being completely excluded from the infrastructure.

## 3.3 | Greedy solution

The proposed deployment problem can be solved with a greedy algorithm: this provide a suboptimal solution that will be used as reference value for analysing the results of the optimization method.[31] The greedy solution described here is capable to handle different type of weighting maps.

The greedy solution consists in placing all the CS one at time such that each one is locally optimal. This means that the first CS is deployed in the position that minimize the selected cost function; then its location is fixed, and the second CS is deployed in a new optimal position.

At each step of the procedure, shown in Figure 3 for three CS, all the points of the map are evaluated as possible location of the CS, and the cost value is computed. In this way, a cost surface is obtained (Figure 3A, Figure 3C, and Figure 3E show the cost surface for a uniformly weighted map) and the minimum cost point of this surface is selected as the new location of the CS (red dots in the figures). This position is then fixed, and the weighed coverage distance can be computed for all the points (Figure 3B, Figure 3D, and Figure 3F).

This iterative procedure is very flexible because it allows to change the weighting maps and it finds at each step the global optimum for that specific CS; however, the final solution found is suboptimal. The computational time required by this method depends on the number of samples in the grid ($N_x$ and $N_y$ for the two axis) and the number of CS ($N_{CS}$):

$$t_{greedy} \propto N_x \cdot N_y \cdot N_{CS} \tag{2}$$

The proportionality constant is the time required to compute the cost function for each cell in the map. The computational time required by this approach is relatively high, and it grows when the map discretization is improved.

## 4 | PROPOSED OPTIMIZATION ENVIRONMENT

In this section, the novel Evolutionary-based optimization environment for CS deployment is presented in all the details that are very important to obtain the required flexibility and, at the same time, to fully exploit the optimization capabilities of many state-of-the-art EAs. These two objectives are achieved by a proper definition of the so-called *optimization environment*, that is, the definition of the design variables, the decodification procedure from optimization to design variables, the implementation of an effective feasibility function, and the identification of the cost function.

The proposed method is based on EAs. They are a class of population-based optimization methods that are able to provide a global optimization tool.[33] These algorithms are growing importance in many engineering fields due to their capability to solve multimodal problems and their natural suitability to parallel computing.[34] All these methods are based on a population of candidate solutions that is evolved during the iterations exploiting the best features found to improve the entire population. The performance of these algorithms depends on the specific operators implemented and on the user-defined parameters.[35]

The optimization environment designed in this study is aimed to be integrated with most of the EAs developed in literature; thanks to this characteristic, the improvements in the optimization algorithms can be directly and effectively introduced in this method.

The proposed optimization environment consists mainly in three important features that have been properly designed to achieve the desired performance. First, the design variables

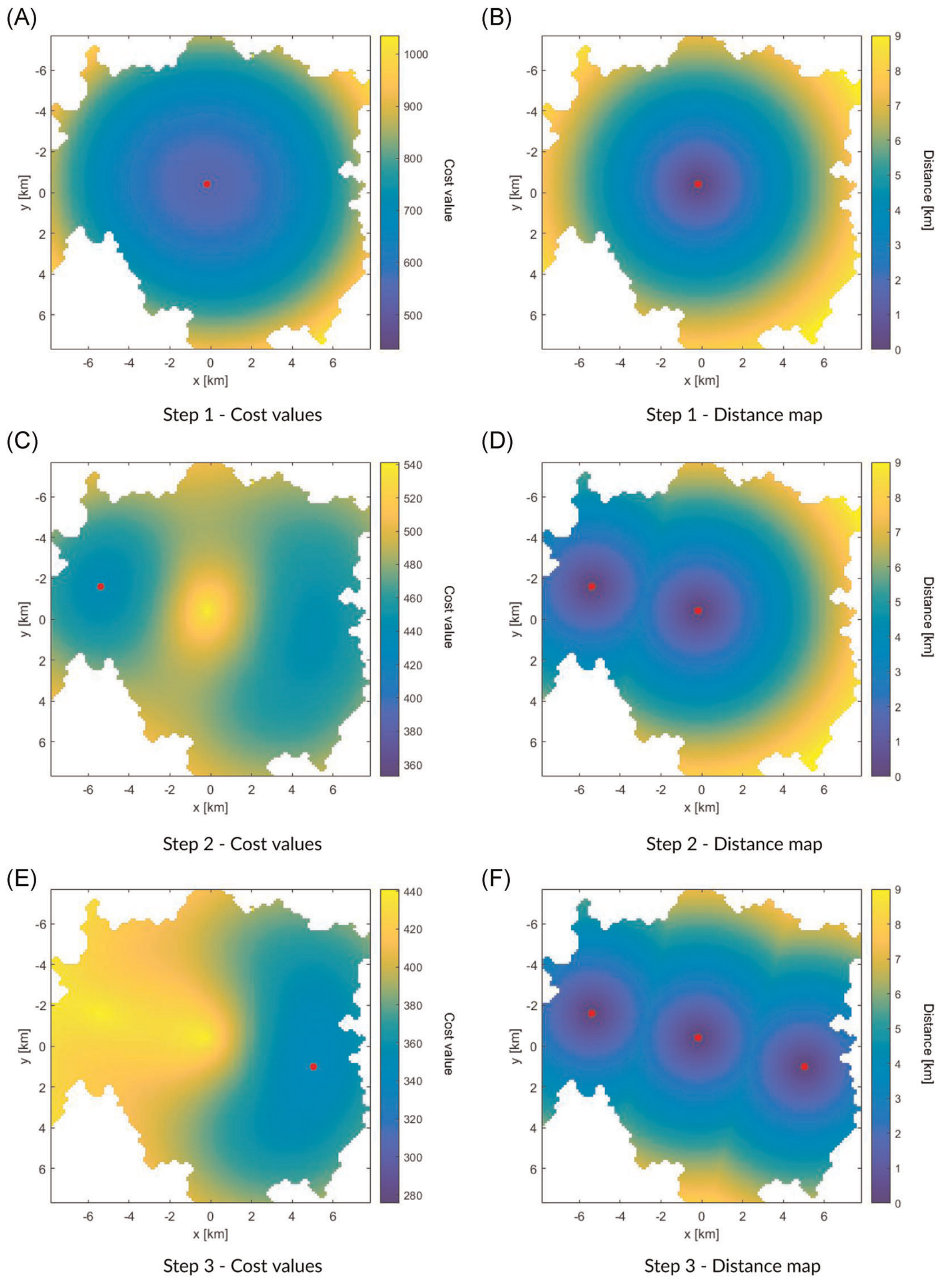


FIGURE 3 First three steps of the greedy approach: (A), (C), and (E) show the cost values for each step of the algorithm and the red dot is the locally optimal position; (B), (D), and (F) show the obtained configuration at each step and the distance from each point in the city to the closest CS. At each step all the points are evaluated as possible location of the CS, and the one that minimize the cost is selected. CS, charging station [Color figure can be viewed at wileyonlinelibrary.com]

have been defined, as well as their relation with the optimization variables of the EAs. This choice is important for the problem scalability, the integration with different EAs, and the reduction of nonlinearities. The second feature is the management of unfeasible solutions: while in many problems it is faced with penalization approaches, here a more sophisticated procedure has been implemented for improving the convergence rate of the optimization process. Finally, the cost function has been defined for taking into account different aspects of the coverage problem.

The design variables selected of this problem are the *x*- and *y*- coordinates of all the CS: in fact, their number depends on the number of CS and not on the city map discretization, allowing a better scalability of the proposed approach. Moreover, they automatically embed the constraint on the required number of CSs. Due to the grid discretization of the city map, the design variables are the integer number representing the matrix indexes.

To enable the use of a larger set of optimization algorithms in this procedure, the variables of the optimizer (*optimization variables*) were considered as real numbers in the range $[0, 1]$: real-valued algorithms are the most of the EAs in literature, and the use of the $[0, 1]$ interval is a common practice because it allows an a priori identification of the user-defined parameters of each algorithm. The optimization variables are mapped into the design variables using a scaling and rounding procedure. The latter does not negatively affect the optimization process thanks to the dimensions of the map matrix.

Each pair of design variables thus obtained represents the position of a CS in the map: in this way, the number of design variables is double the number of CS:

$$N_{DV} = 2 \cdot N_{CS} \tag{3}$$

The three-step procedure implemented for obtaining the candidate solution (position of all the CS) from the optimization variables is summarized in Figure 4.

While this procedure enables the use of a large set of optimization algorithms, it raises to problem of managing unfeasible solutions; by this it is intended solutions in which one or more CS are placed outside the urban area. Even if this issue it can be solved automatically during the optimization process because all these unfeasible solutions are suboptimal, however, in this study they are managed by a feasibility function to improve convergence speed. In addition, the impact of unfeasible solutions would grow with the increase in the number of CS, if not properly managed.

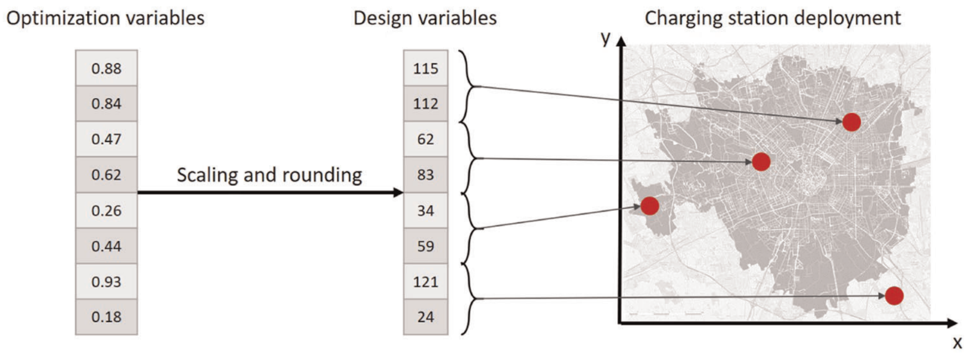


FIGURE 4 Optimization process: the optimization variables are scaled and rounded to obtain the design variables. Each couple of these represents a position of a charging station [Color figure can be viewed at wileyonlinelibrary.com]

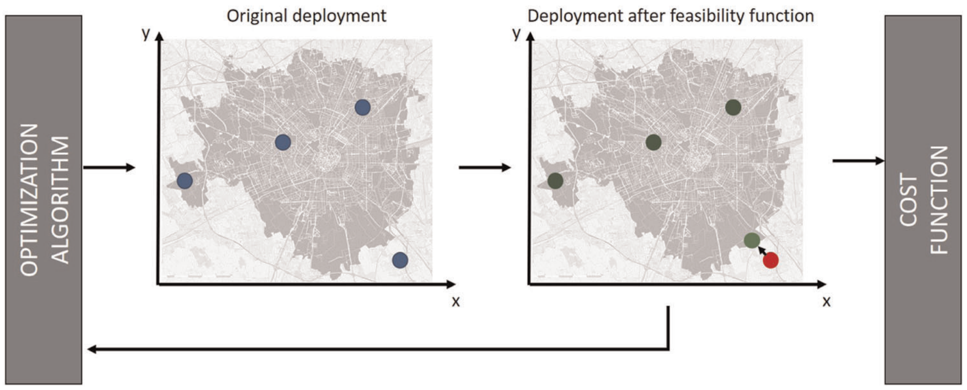


**FIGURE 5** Optimization process: effect of the feasibility function. All the charging stations outside the urban area are mapped to the closest feasible position: the obtained solution is used for the cost function computation and it is feedback in the optimization loop [Color figure can be viewed at wileyonlinelibrary.com]

To face the problem of unfeasible solutions, in this optimization environment an ad hoc approach has been implemented. It consists in analysing unfeasible solutions and in moving the critical CS in the closest point that belongs to the urban area. This feasibility function has many advantages: compared with other approaches as penalization methods or solution elimination, it improves the optimization convergence. Second, it does not impact on the computational time required by the optimization: in fact, the closest feasible point can be computed a priori for each unfeasible cell in the map. Finally, maps in which some unfeasible areas are placed within the urban environment (historical city centres, parks, ...) can be used without loosing in optimization performance.

The feasible solution obtained from the feasibility function is both used in the cost function computation and feedback into the optimization algorithm modifying the original candidate solution: in this way, the modifications are well integrated in the optimization loop improving even more the convergence rate. All this procedure is schematized in Figure 5.

Algorithm 1 shows the entire process from the optimization variables (the vector $x$) to the final feasible positions of the CSs (the two vectors $x_{CS}$ and $y_{CS}$). This decodification procedure is based on three input information: the number of cells in the discretized map in $x$- and $y$-direction (respectively $N_x$ and $N_y$) and the coordinates of the closest feasible point (the two matrices $x_{closest}$ and $y_{closest}$).

**Algorithm 1** Design variable decodification and feasibility analysis

```
1: procedure Decodification $x$                          ▷ x is the vector representing the optimization variables
2:   for $i \leftarrow 1, N_{CS}$                              ▷ $N_{CS}$ is the number of CSs
3:     $x_{CS}(i) \leftarrow \mathrm{ceil}(x(2 \cdot (i-1)+1) \cdot N_x)$      ▷ $N_x$ is the number of cells in $x$ direction
4:     $y_{CS}(i) \leftarrow \mathrm{ceil}(x(2 \cdot i+1) \cdot N_y)$          ▷ $N_y$ is the number of cells in $y$ direction
5:     if $incity(X_{CS}(i), Y_{CS}(i)) = 0$ then               ▷ If the CS is deployed out of the city
```

(Continues)

```
6:        $x_{CS}(i) \leftarrow x_{closest}(x_{CS}(i), y_{CS}(i))$        ▷ The closest feasible point is assigned
7:        $y_{CS}(i) \leftarrow y_{closest}(x_{CS}(i), y_{CS}(i))$
8:      end if
9:    end for
10:   return $x_{CS}, y_{CS}$        ▷ The position of the CSs is returned for cost calculation
11: end procedure
```

The computation of these last two matrices is a key point for the proposed procedure. In fact, this improves the convergence of the optimization process and they can be calculated only once at the beginning of the optimization. The definition of this matrices is shown by the pseudocode in Algorithm 2. The procedure is quite computationally expensive, but it has been optimized: the search of the closest point is performed with a circular search around each unfeasible cell of the map, thus the first feasible point identified is the closest one and the process can be stopped. The first two for loops in the pseudocode correspond to the analysis of all the points of the map. Then, the third loop performs the circular search. This entire procedure, for the analysed test case, lasts only 0.89 s.

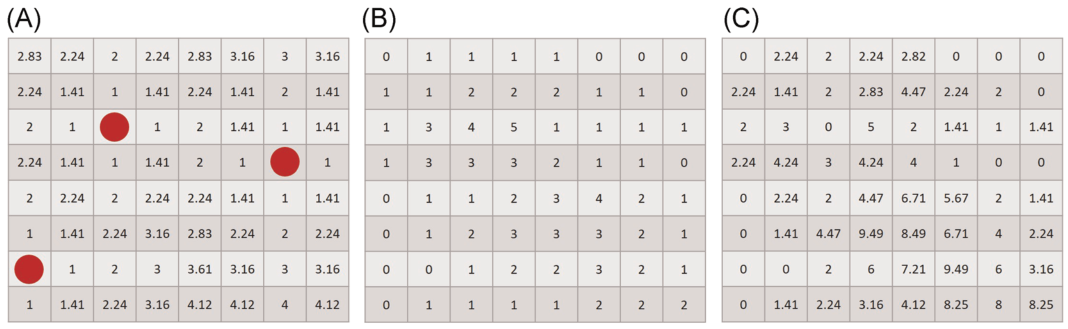
**Algorithm 2** Calculation of closest feasible points

```
1: procedure FindClosest incity        ▷ incity is a matrix representing the urban feasible area
2:   for $i_0 \leftarrow 1, N_x$ do        ▷ $N_x$ is the number of cells in x direction
3:     for $j_0 \leftarrow 1, ..., N_y$ do        ▷ $N_y$ is the number of cells in y direction
4:       $x_{closest}(i_0, j_0) \leftarrow 0$        ▷ Initialize to zero
5:       $y_{closest}(i_0, j_0) \leftarrow 0$
6:       if $incity(i_0, j_0) = =0$ then        ▷ If the point out of the city
7:         stopFlag ←0
8:         $k \leftarrow 1$
9:         while not stopFlag do        ▷ While the closest point is not found
10:          for $z \leftarrow 0, k$ do        ▷ Start the round search
11:            if not $z = =k$ then        ▷ Identify the search directions
12:              $D_i \leftarrow [-k, -k, k, k, -z, -z, z, z]$
13:              $D_j \leftarrow [-z, z, z, -z, -k, k, k, -k]$
14:              $n_D \leftarrow 8$
```

(Continues)

```
15:             else
16:                 $D_i = [-k, -k, k, k]$
17:                 $D_j = [-z, z, z, -z]$
18:                 $n_D \leftarrow 4$
19:             end if
20:             $l \leftarrow 1$
21:             while $l \geq n_D$ and not stopFlag do    ▷ Search in the identified directions
22:                 $i_n = i_0 + D_i(l)$
23:                 $j_n = j_0 + D_j(l)$
24:                 if $i_n > 0$ and $j_n > 0$ and $i_n \leq N_x$ and $j_n \leq N_y$ then    ▷ If it is inside the map
25:                     if not $incity(i_n, j_n)$ then    ▷ If the point belongs to the city
26:                         $x_{closest}(i_0, j_0) = i_n$    ▷ Set it as the closest one
27:                         $y_{closest}(i_0, j_0) = j_n$
28:                         stopFlag $\leftarrow 1$    ▷ Stop all the search procedure
29:                     end if
30:                 end if
31:                 $l \leftarrow l + 1$
32:             end while
33:         end for
34:         $k \leftarrow k + 1$
35:       end while
36:     end if
37:   end for
38: end for
39: return $x_{closest}, y_{closest}$
40: end procedure
```

When the candidate solution has been correctly mapped into feasible CS positions, the coverage values are computed. The proposed approach allows the use of different weighting matrices, as described before, to face a large number of coverage problems and to be highly effective in real life.



(A)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2.83 | 2.24 | 2 | 2.24 | 2.83 | 3.16 | 3 | 3.16 |
| 2.24 | 1.41 | 1 | 1.41 | 2.24 | 1.41 | 2 | 1.41 |
| 2 | 1 | ● | 1 | 2 | 1.41 | 1 | 1.41 |
| 2.24 | 1.41 | 1 | 1.41 | 2 | 1 | ● | 1 |
| 2 | 2.24 | 2 | 2.24 | 2.24 | 1.41 | 1 | 1.41 |
| 1 | 1.41 | 2.24 | 3.16 | 2.83 | 2.24 | 2 | 2.24 |
| ● | 1 | 2 | 3 | 3.61 | 3.16 | 3 | 3.16 |
| 1 | 1.41 | 2.24 | 3.16 | 4.12 | 4.12 | 4 | 4.12 |

(B)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 2 | 2 | 2 | 1 | 1 | 0 |
| 1 | 3 | 4 | 5 | 1 | 1 | 1 | 1 |
| 1 | 3 | 3 | 3 | 2 | 1 | 1 | 0 |
| 0 | 1 | 1 | 2 | 3 | 4 | 2 | 1 |
| 0 | 1 | 2 | 3 | 3 | 3 | 2 | 1 |
| 0 | 0 | 1 | 2 | 2 | 3 | 2 | 1 |
| 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |

(C)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 2.24 | 2 | 2.24 | 2.82 | 0 | 0 | 0 |
| 2.24 | 1.41 | 2 | 2.83 | 4.47 | 2.24 | 2 | 0 |
| 2 | 3 | 0 | 5 | 2 | 1.41 | 1 | 1.41 |
| 2.24 | 4.24 | 3 | 4.24 | 4 | 1 | 0 | 0 |
| 0 | 2.24 | 2 | 4.47 | 6.71 | 5.67 | 2 | 1.41 |
| 0 | 1.41 | 4.47 | 9.49 | 8.49 | 6.71 | 4 | 2.24 |
| 0 | 0 | 2 | 6 | 7.21 | 9.49 | 6 | 3.16 |
| 0 | 1.41 | 2.24 | 3.16 | 4.12 | 8.25 | 8 | 8.25 |

**FIGURE 6** Weighted coverage computation: (A) the euclidean distance between each cell and the closest charging station is calculated; (B) represent the weighting matrix; (C) with an element-wise multiplication the weighted distance is calculated [Color figure can be viewed at wileyonlinelibrary.com]

For each cell of the discretized map, the Euclidean distance to the closest CS is weighted as indicated by the weighting matrix $W$. From this procedure a weighted distance matrix is obtained; thus, two different performance parameters can be extracted: the average distance, that represent the global feeling of the quality of service, and the maximum distance, that takes into account the worst-case scenario in the urban environment. Both these values are used in the optimization process, resulting in the following cost equation:

$$c = \frac{1}{n_x \cdot n_y} \sum_{i}^{n_x} \sum_{j}^{n_y} (w_{i,j} \cdot d_{i,j}) + 10^{-2} \cdot \max_i \max_j (w_{i,j} \cdot d_{i,j}) \tag{4}$$

where $c$ is the cost value, $n_x$ and $n_y$ are the map dimensions, $w_{i,j}$ is the weighting matrix and $d_{i,j}$ is the distance matrix, that contains in each cell the Euclidean distance between the cell itself and the closest CS.

Figure 6 exemplifies the procedure of weighted distance calculation: in (6A) the red dots represent the position of the CS and the numbers are the Euclidean distances to the closest CS; (6B) represents a weight matrix, and (6C) shows the weighted distance, obtained from an element-wise multiplication.

## 5 | RESULTS

In this section, the proposed method has been applied to the test case of the city of Milan using two different weighting maps, that is, on two optimization objectives. For both of these, the four tested algorithms have been compared and the best has been selected for the further analysis. Its results have been compared in terms of performance indexes and achieved deployment with the greedy approach. Finally, the two optimal solutions with 100 CS have been deeply analysed.

To prove the flexibility and effectiveness of the proposed method with different algorithms, for EAs has been here tested: the first two are classical and well-established algorithms, the particle swarm optimization (PSO)[36] and the GA.[37] Even if these algorithms are often outperformed by more recent ones, they have been introduced in this analysis due to the fact that many other methods in literature exploit them and, thus, the improvements introduced with the proposed optimization environment can be better investigated.

The second group consists in two more recent and performative algorithms: the biogeography-based optimization (BBO)[38] and the social network optimization (SNO).[39] Both

**TABLE 1** Parameters for all the four tested evolutionary algorithms: (a) particle swarm optimization, (b) genetic algorithm, (c) biogeography-based optimization, and (d) social network optimization

| Parameter | Value |
|---|---|
| (a) Particle swarm optimization | |
| Population size | 20 |
| Inertia | 0.95 |
| Acceleration coefficients | 0.45 |
| Velocity clamping | 0.35 |
| (b) Genetic algorithm | |
| Population size | 25 |
| Crossover probability | 0.98 |
| Mutation rate | 0.1 |
| Mutation amplitude | 0.06 |
| (c) Biogeography-based optimization | |
| Population size | 20 |
| Immigration coefficient | 500 |
| Emigration coefficient | 1 |
| Mutation rate evolution | 0.01 |
| (d) Social network optimization | |
| Population size | 40 |
| Attraction rate | 0.64 |
| Preserve rate | 0.52 |
| Linguistic error probability | 0.08 |

of them have been widely tested in literature over many engineering problems and their performance have been well assessed.

Performance of EAs are highly affected by the definition of their internal parameters. In this study, they have been defined by means of a sensitivity analysis for improving the performance. The values found to be optimal are reported in Table 1.

The number of iterations of the algorithms ($n_{iter}$) is set in each test based on the number of objective function calls ($n_{call}$) and the population size of the algorithms ($n_{pop}$):

$$n_{call} = n_{iter} \cdot n_{pop} \tag{5}$$

In fact, in all test performed the number of objective function calls is used as termination criterion for all EAs. This ensure a fair comparison between the algorithms because this last parameter is proportional to the total computational time required to perform the entire optimization process since the self-time of the algorithms is negligible compared to the total optimization time. More details regarding the optimization times are provided in the following of the section.

Furthermore, the total number of objective function calls was fixed by the greedy approach; for this method, the computational effort depends only on the discretization of the city map and the number of CS deployed:

$$n_{call} = N_x \cdot N_y \cdot N_{CS} \tag{6}$$

where $N_{CS}$ is the number of CS, and $N_x \times N_y$ is the map discretization size; in the analysed case study it is $130 \times 132$.

## 5.1 | Space coverage objective

The first group of optimization tests was carried out by optimizing the space coverage objective, that is, weighting each point of the map with an unitary factor.

First, the four EAs used in this study (GA, PSO, BBO, and SNO) were compared. In this comparison, a test case with 100 CS was used. The maximum number of calls to the cost function was set, coherently to what said before, at 1,030,000 and 40 independent runs were made to reduce the impact of the stochasticity of EAs.

Figure 7 shows the results of this comparison. In particular, in Figure 7A the convergence curves in semilogarithmic scale of the four algorithms have been represented: the line corresponds to the average of the 40 independent runs, while the coloured section represents the confidence interval at 90%.

The algorithms have several convergence characteristics: PSO has the slowest convergence rate, however the convergence continues with a good gradualness throughout the final part of the optimization. This algorithm has the largest *SD* of independent trials. On the contrary, the other three algorithms manage to have a rapid convergence phase after which the dispersion of the results is very low. In particular, the BIMBO has a very low dispersion of trials throughout the optimization process: this is due the strong pressure ratio of its selection operators. The GA exhibits good convergence, however slightly slower and less effective than the BBO. Finally, the convergence of SNO is characterized by a limited initial speed and by a fair dispersion of the launches due to the initial exploration capability of the algorithm; however, starting from 50,000 calls to the cost function (5% of the total optimization time), the launches return to having very similar cost values.

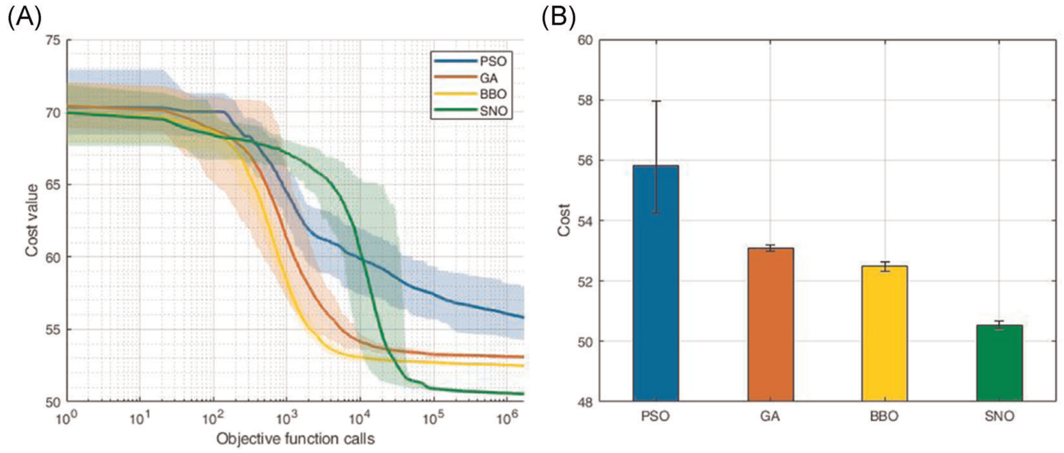


FIGURE 7 Comparison between four different EAs on the space coverage objective with 100 charging stations. In (A) the convergence curves are reported: the thick line is the average of 40 independent trials, while the coloured bands correspond to the 90% confidence value. In (B) the comparison on the final values is presented with the 90% confidence value. BBO, biogeography-based optimization; EA, evolutionary algorithms; GA, genetic algorithm; PSO, particle swarm optimization; SNO, social network optimization [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Final results of the optimization process of the four tested algorithms: the average and the standard deviation of the 40 independent trials are reported with the average and maximum distance for the best solution found

| Algorithn | Average cost | *SD* | Average distance (km) | Maximum distance (km) |
|---|---|---|---|---|
| PSO | 55.8 | 1.06 | 0.54 | 1.21 |
| GA | 53.1 | 0.07 | 0.53 | 1.34 |
| BBO | 52.49 | 0.09 | 0.52 | 1.31 |
| SNO | 50.55 | 0.07 | 0.5 | 1.08 |

Abbreviations: BBO, biogeography-based optimization; GA, genetic algorithm; PSO, particle swarm optimization; SNO, social network optimization.

TABLE 3 Average running time for the four tested algorithms

| Algorithn | Average running time (s) |
|---|---|
| PSO | 104.06 |
| GA | 100.94 |
| BBO | 133.14 |
| SNO | 110.68 |

Abbreviations: BBO, biogeography-based optimization; GA, genetic algorithm; PSO, particle swarm optimization; SNO, social network optimization.

Figure 7B shows the detail of the dispersion of the final results of the 40 independent launches: BBO, GA, and SNO are characterized by a low dispersion. This is very important in real applications, and in particular in computational expensive ones, because it allows to reduce the number of independent runs that must be performed.

The details of the optimization results are shown in Table 2: the average value and the standard deviation of all the 40 independent trials are reported for all the algorithms. In addition, the table shows the average distance and the maximum distance achieved by the best solution of each algorithm.

Table 3 shows the comparison between the computational time required by the four algorithms to run on an Intel(R) Core(TM) i9 10900KF CPU. The small differences between the computational times can be due to the different overload required by the algorithm and due to different background operations.

From the results shown, SNO obtains the best performance with a computational time comparable with PSO and GA; due to these reasons, the following analysis on this performance parameter has been limited to this algorithm.

The solutions obtained by SNO are compared with the greedy approach. Also in this case the number of objective function calls of SNO is set by the limits of the greedy approach ($n_{call} = 130 \cdot 132 \cdot N_{CS}$) to have a fair comparison.

Figure 8 shows the comparison between SNO and the greedy approach with a number of CS varying from 10 to 100. The blue line is the cost value of the greedy method, while for SNO it is represented the average final cost of 40 independent trials (green line) and the 95% confidence bound (light green area). The insert shows the details for CS above 50.

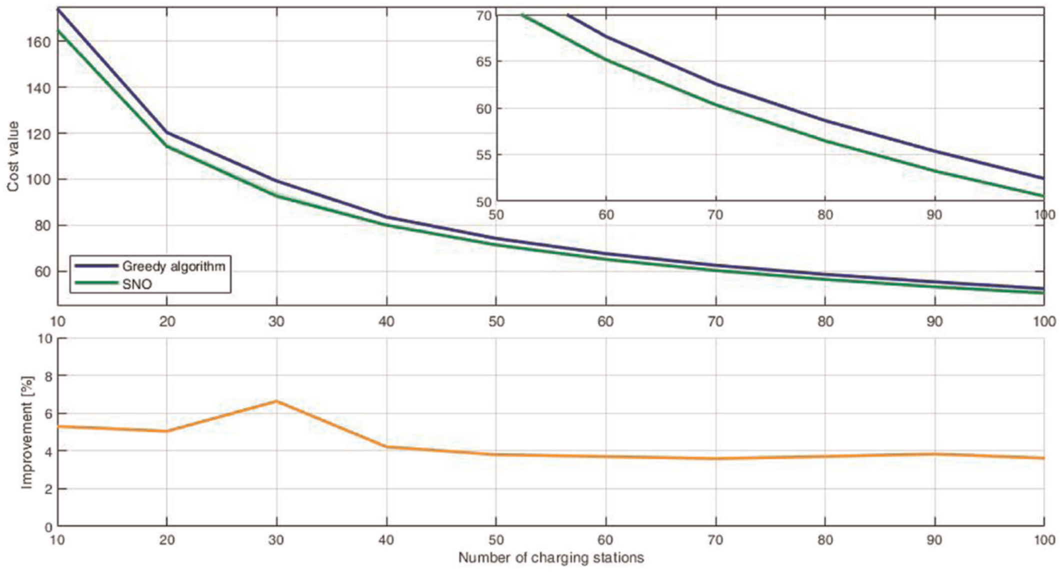


FIGURE 8 Comparison between the greedy approach and the solution of SNO, with a number of charging stations ranging from 10 to 100. The insert shows the details for charging stations above 50, and the bottom graph shows the improvement in per unit. SNO, social network optimization [Color figure can be viewed at wileyonlinelibrary.com]

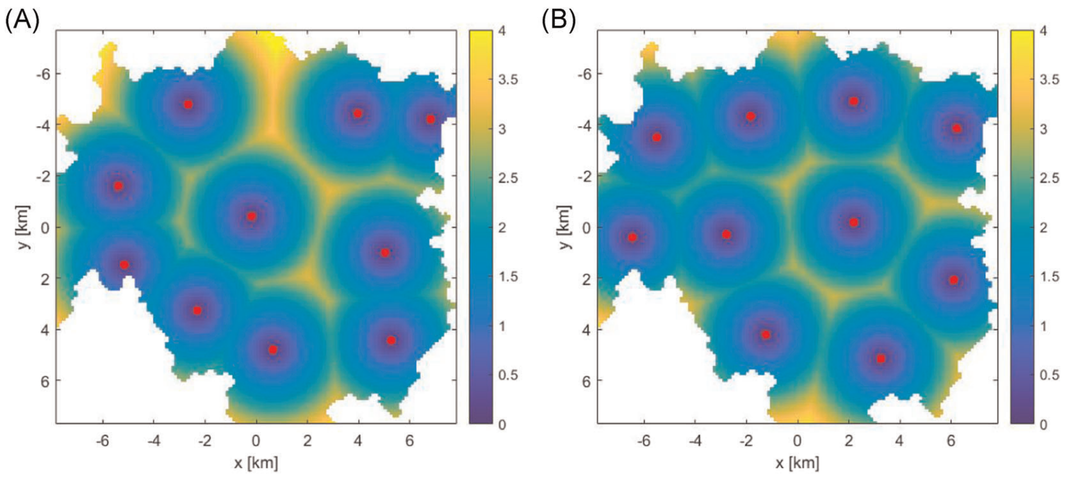


FIGURE 9 Comparison between the optimal solution with 10 charging stations of (A) the greedy algorithm and (B) SNO. The red dots represent the position of the charging stations, while the colormap is representative of the distance to the closest charging stations (in km). SNO, social network optimization [Color figure can be viewed at wileyonlinelibrary.com]

In the figure, it is hard to see the confidence bound of SNO because the reliability of the algorithm is very high. The performance of SNO outperforms for all the tested cases the greedy solution. The improvement introduced by the optimization process ranges from the 3.6% to 6.6%.

**TABLE 4** Comparison between greedy solutions and SNO solutions for all the tested number of charging stations ($N_{CS}$)

| | Greedy | | SNO | |
|---|---|---|---|---|
| $N_{CS}$ | Average dist. (km) | Maximum dist. (km) | Average dist. (km) | Maximum dist. (km) |
| 10 | 1.74 | 4.28 | 1.64 | 3.91 |
| 20 | 1.2 | 3.15 | 1.14 | 2.49 |
| 30 | 0.99 | 2.64 | 0.92 | 2.34 |
| 40 | 0.83 | 1.91 | 0.8 | 1.82 |
| 50 | 0.74 | 1.78 | 0.71 | 1.55 |
| 60 | 0.67 | 1.78 | 0.65 | 1.34 |
| 70 | 0.62 | 1.58 | 0.6 | 1.31 |
| 80 | 0.58 | 1.58 | 0.56 | 1.27 |
| 90 | 0.55 | 1.58 | 0.53 | 1.16 |
| 100 | 0.52 | 1.38 | 0.5 | 1.11 |

Abbreviation: SNO, social network optimization.

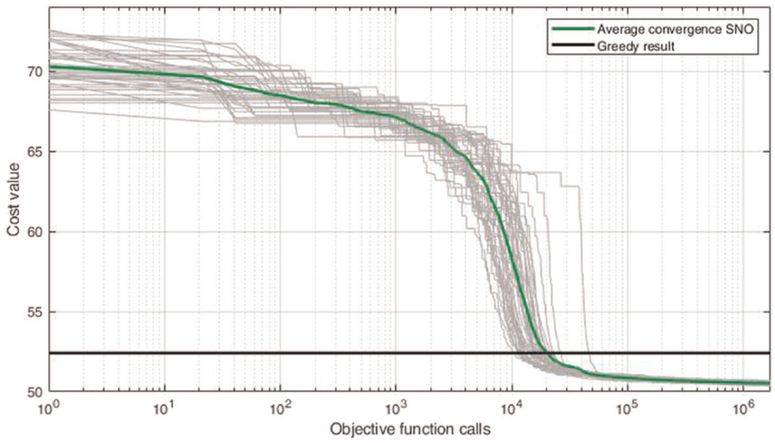


**FIGURE 10** Convergence curves for SNO with 100 charging stations. The thick green line is the average value of the 40 independent runs, represented each with a grey line; the black line is the cost value obtained with the greedy approach. SNO, social network optimization [Color figure can be viewed at wileyonlinelibrary.com]

Finally, the plot in the bottom shows the improvement from the greedy solution to the optimized one. It is interesting to see that this is almost constant with respect to the number of CS and it is around 4%.

The differences between the SNO and the greedy solutions can be seen from Figure 9. Here, the optimal solutions with 10 CS is presented because it allows easy considerations that can be extended to more complex cases.

In this analysed case, the coverage capability is quite limited for both the approaches due to the reduced number of CS. The greedy solution has a limited coverage on the

**TABLE 5** Final results of the optimization process of the four tested algorithms: the average and the standard deviation of the 40 independent trials are reported with the average and maximum distance for the best solution found

| Algorithn | Average cost | *SD* | Average weighted dist. | Maximum weighted dist. |
|---|---|---|---|---|
| PSO | 51.84 | 1.29 | 46.97 | 253.31 |
| GA | 49.43 | 0.15 | 46.54 | 255.88 |
| BBO | 48.2 | 0.14 | 45.62 | 222.15 |
| SNO | 44.5 | 0.17 | 42.34 | 185.93 |

Abbreviations: BBO, biogeography-based optimization; GA, genetic algorithm; PSO, particle swarm optimization; SNO, social network optimization.

external parts of the city due to the approach limitations that force the first placed CS in the centre of the city. This limitation is not present in the optimization approach because it provides a global search that takes into account all the CSs at the same time. The average distance from a cell to the closest CS is 1.73 km for the greedy solution, while it is reduced to 1.65 km for SNO. The difference between these approaches is even higher considering the maximum distance: in fact, it is 4.2 km for the greedy approach and 3.7 km in the SNO solution.

A comparison between the greedy solution and the optimal solution found by SNO is provided in Table 4 for all the analysed numbers of CS. For both the tested approaches, the table shows the average and maximum distances to the closest CS.

These results confirm the superiority of SNO with respect to the greedy approach, and show how much this algorithm is able to properly handle both the average and maximum distances.

Finally, Figure 10 shows the details of the convergence curves of SNO with 100 CS. Each independent trial is represented with a grey line; the green thick line is the average convergence and the black line is the cost value obtained with the greedy approach.

All the SNO trials outperform the greedy solution in 1/20 of the computational time. Moreover, this figure clearly shows the reliability of the solutions obtained by SNO: indeed, all independent trials achieve very similar cost values. This is very important in the real application of the proposed technique: having a high reliability allows to perform fewer independent trials and, thus, increasing the number of objective function calls with the same total computational effort.

## 5.2 | Population coverage objective

In this section, the flexibility of the proposed approach is tested analysing the second objective, that is, the coverage in where each cell is weighted by its population density.

In this case the optimization process is slightly more complex, and the differences between SNO and the other tested algorithms are increased. The numerical results reported in Table 5 refer to an optimization process in which for each algorithm 40 independent trials have been performed using as termination criterion 1,030,000 objective function calls.

The cost value for PSO, GA, and BBO is comparable: in particular, the average weighted distance value is very similar between the three algorithms, while the BBO is able to find a solution with a better maximum weighted distance. The SNO results are better than the other

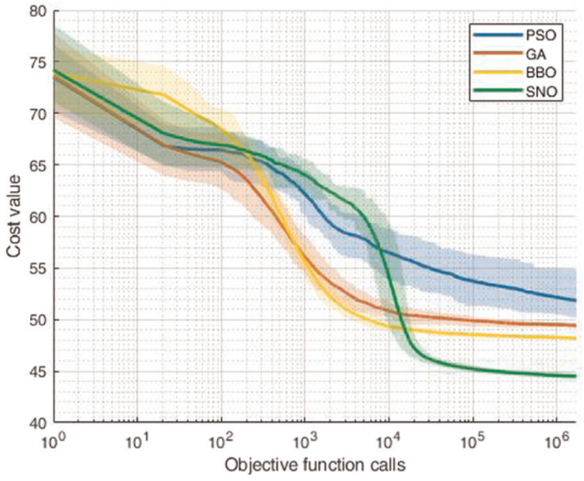


**FIGURE 11** Comparison between the convergence curves of the four different EAs on the population coverage objective with 100 charging stations. The thick line is the average of 40 independent trials, while the coloured bands correspond to the 90% confidence value. BBO, biogeography-based optimization; EA, evolutionary algorithm; GA, genetic algorithm; PSO, particle swarm optimization; SNO, social network optimization [Color figure can be viewed at wileyonlinelibrary.com]

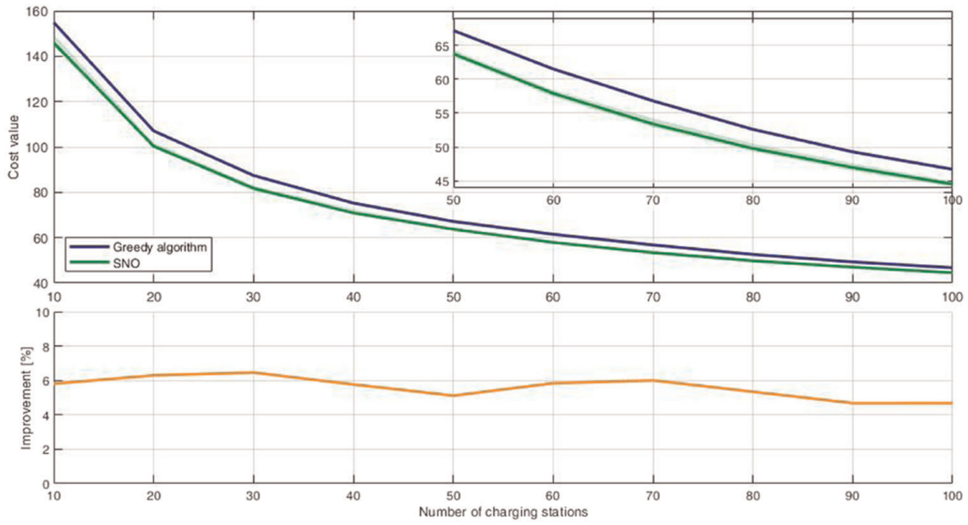


**FIGURE 12** Comparison between the greedy approach and the solution of SNO, with a number of charging stations ranging from 10 to 100. The insert shows the details for charging stations above 50, and the bottom graph shows the improvement in per unit. SNO, social network optimization [Color figure can be viewed at wileyonlinelibrary.com]

algorithms both in terms of average and maximum weighted distances. The dispersion of the results of the PSO remains quite high, while for GA, BBO and SNO it is comparable.

The convergence curves, shown in Figure 11, show a slightly different behaviour than the previous objective. Indeed, in this case it is the BBO that starts with the lowest convergence rate, followed by a rapid increase between 100 and 1000 objective function calls. In this time

**TABLE 6** Comparison between greedy solution and the best SNO solution for 10 different number of charging stations ($N_{CS}$): for both the solutions the average and the maximum weighted distance are compared

| | **Greedy** | | **SNO** | | |
|---|---|---|---|---|---|
| $N_{CS}$ | **Average w. dist.** | **Maximum w. dist.** | **Average w. dist.** | **Maximum w. dist.** | **Break even time (%)** |
| 10 | 145.52 | 933.57 | 137.71 | 668.56 | 1.2 |
| 20 | 100.75 | 631.29 | 94.81 | 474.62 | 1.31 |
| 30 | 82.63 | 478.23 | 77.02 | 360.28 | 1.47 |
| 40 | 71.05 | 417.8 | 67.05 | 330.37 | 1.37 |
| 50 | 63.69 | 345.39 | 60.33 | 298.31 | 1.44 |
| 60 | 58.03 | 345.39 | 55.02 | 235.19 | 1.22 |
| 70 | 53.89 | 290.27 | 50.6 | 230.74 | 1.24 |
| 80 | 50.08 | 252.33 | 47.35 | 199.12 | 1.29 |
| 90 | 46.95 | 234.54 | 44.55 | 194.33 | 1.44 |
| 100 | 44.38 | 234.54 | 42.31 | 183.7 | 1.47 |

*Note*: The last column shows the percentage of optimization time required by SNO to reach the same performance of the greedy algorithm.

Abbreviation: SNO, social network optimization.

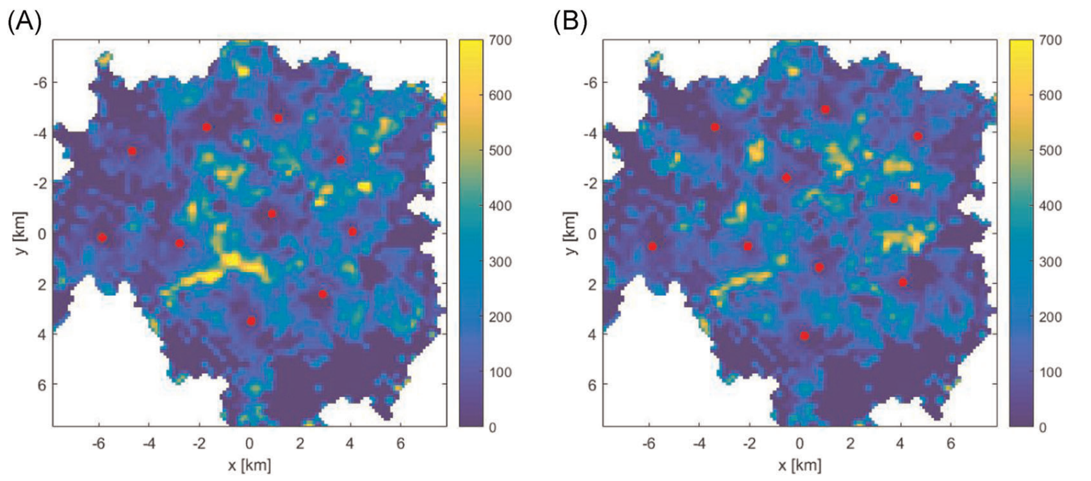


**FIGURE 13** Comparison between the optimal solution with 10 charging stations of (A) the greedy algorithm and (B) SNO. The red dots represent the position of the charging stations, while the colormap is representative of the weighted distance to the closes charging stations. SNO, social network optimization [Color figure can be viewed at wileyonlinelibrary.com]

interval, also the PSO and the GA show an acceleration in converge, while for SNO this occurs in a slightly later phase. Similarly to before, the SNO convergence delay guarantees a better exploration to this algorithm, which therefore manages to converge towards a better minimum. Also in this case, SNO outperforms all the other algorithms, consequently it has been used in the following analyses.

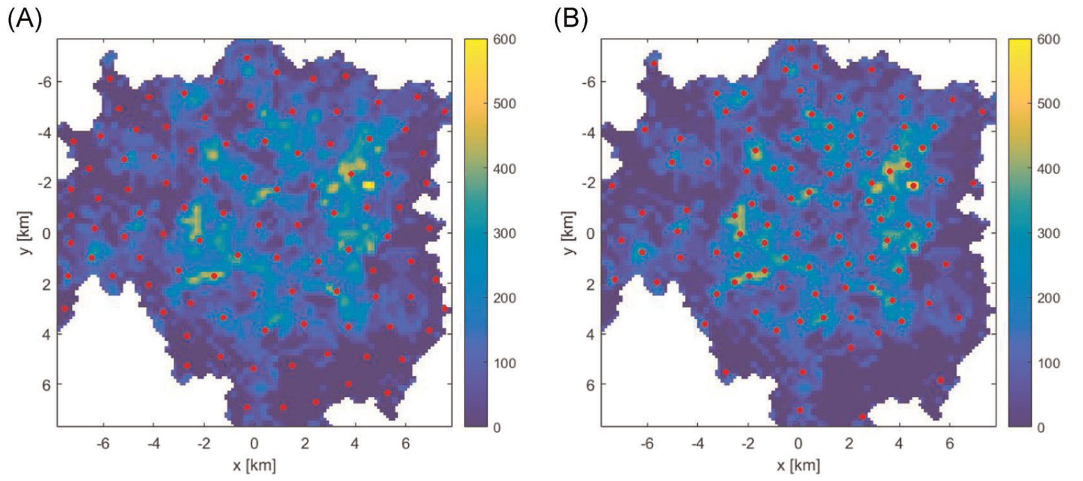


**FIGURE 14** Comparison between the SNO optimal solution with 100 charging stations for (A) the space coverage objective, and (B) the population coverage. The red dots represent the position of the charging stations, while the colormap is proportional to the population density. SNO, social network optimization [Color figure can be viewed at wileyonlinelibrary.com]

Figure 12 shows, as previously done for the space coverage, the comparison between SNO and the greedy approach with a number of CS varying from 10 to 100. With respect to the previous performance indicator, in this case the improvement introduced by the optimized solution is higher; thus, this indicates that the optimization process becomes more convenient increasing the complexity of the target required.

The effectiveness of SNO is finding a good solution both in terms of average and maximum distance. This fact can be seen also in the comparison with the greedy approach shown in Table 6. For the SNO optimization, 40 independent trials have been performed and the termination criterion has been set to $130{\cdot}130{\cdot}N_{CS}$ objective function calls to have the same computational time required by both the two approaches. The table shows the comparison for 10 different values of CS numbers: the best solution found by SNO is compared with the greedy one in terms of average and maximum weighted distances.

The last column of Table 6 shows the percentage of optimization time required by SNO to reach the same performance of the greedy algorithm. This last parameter is very important for understanding all the advantages of the proposed methodology with respect to the greedy approach: in fact, to have the same performance, SNO requires less than the 2% of the computational time.

The distance weighting impacts on the greedy algorithm more than on the EAs. This can be seen from Figure 13 where the greedy and the SNO optimal solutions with 10 CS are compared. The red dots are the charging stations positions, while the colormap represents the weighted distance between each point of the map and the closest CS.

The first CS placed by the greedy approach is located in the barycentre of the density map; this means that it is placed in the city centre where the population density is relatively low, worsening all the other solutions found.

Finally, Figure 14 shows a comparison between the two optimal solutions found by SNO optimizing the space coverage objective (Figure 14A) and the population coverage (Figure 14B). In these figures the coloromap represents the population density, thus it is possible to clearly see the difference introduced by changing the weighting factors in the optimization process.

In Figure 14A, the CSs are evenly distributed in the urban area; in the central part the distance between the CSs is almost constant, while on the city boundaries they are located to cover all the irregularities of the city boundary. On the other hand, in Figure 14B the CSs location follows very well the population density: it is interesting to see that a CS is exactly located in the highest density point of the city.

## 6 | CONCLUSIONS

In this paper, a new optimization environment has been introduced to apply in a flexible and effective way EAs to the EV CS deployment problem. The proposed approach avoids many of the hypotheses that are often introduced in other works, providing in this way a very flexible tool capable of managing different requirements.

The presented optimization environment defines design variables with the aim to enlarge the number of EAs that can be integrated in; moreover, the scalability issue is taken into account, as well as the reduction of non-linearities. The selected design variables lead to the problem of unfeasible solutions management, that has been solved with a methodology able to improve the convergence rate with low computational impact on the optimization process. Finally, the cost function has been designed to manage different performance indicators.

This system models the social requirements for CSs by means of weights maps. Two different objectives have been analysed in this paper: the space coverage, where all the points are equally weighted and provides an even distribution of CSs in the urban area, and the population coverage, where the weights correspond to the population density. This last objective leads to solutions in which the CSs are concentrated in the areas with more inhabitants.

The proposed methodology has been compared on both the objectives with the greedy algorithm, showing the superiority of the evolutionary-based approach both in terms of performance and of computational time required to achieve a solution.

Many new social and economical requirements can be introduced in this procedure: the simplest approach is the definition of new maps that take into account different aspects, like the cost of installation, the distance from electrical network, and so on. The definition of the cost function can be suitable for adding other performance metrics, like the average charging time or the overload of the electricity grid. The low computational cost required by the optimization makes this approach feasible also for hierarchical optimization.

## ACKNOWLEDGMENTS

Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement.

## ORCID

*Alessandro Niccolai* https://orcid.org/0000-0002-5840-4222
*Leonardo Bettini* https://orcid.org/0000-0002-7298-7161
*Riccardo Zich* https://orcid.org/0000-0003-1845-0811

## REFERENCES

1. Gandoman FH, Ahmadi A, Van den Bossche P, et al. Status and future perspectives of reliability assessment for electric vehicles. *Reliability Eng Syst Safety*. 2019;183:1-16.
2. Zheng X, Lin H, Liu Z, Li D, Llopis-Albert C, Zeng S. Manufacturing decisions and government subsidies for electric vehicles in China: a maximal social welfare perspective. *Sustainability*. 2018;10(3):672.
3. Lévay PZ, Drossinos Y, Thiel C. The effect of fiscal incentives on market penetration of electric vehicles: a pairwise comparison of total cost of ownership. *Energy Policy*. 2017;105:524-533.

4. Peña A, Bonet I, Lochmuller C, Chiclana F, Góngora M. An integrated inverse adaptive neural fuzzy system with Monte-Carlo sampling method for operational risk management. *Expert Syst Appl*. 2018;98:11-26.
5. Moodley R, Chiclana F, Caraffini F, Carter J. A product-centric data mining algorithm for targeted promotions. *J Retailing Consumer Serv*. 2020;54:101940.
6. Saidala RK, Devarakonda N. Multi-swarm whale optimization algorithm for data clustering problems using multiple cooperative strategies. *Int J Intell Syst Appl*. 2018;10(8):36.
7. De Domenico D, Ricciardi G, Takewaki I. Design strategies of viscous dampers for seismic protection of building structures: a review. *Soil Dynam Earthquake Eng*. 2019;118:144-165.
8. Yang B, Wang J, Zhang X, et al. Comprehensive overview of meta-heuristic algorithm applications on PV cell parameter identification. *Energy Conv Manage*. 2020;208:112595.
9. Pérez LG, Mata F, Chiclana F, Kou G, Herrera-Viedma E. Modelling influence in group decision making. *Soft Comput*. 2016;20(4):1653-1665.
10. Dallagnol V, van den Berg J, Mous L. Portfolio management using value at risk: a comparison between genetic algorithms and particle swarm optimization. *Int J Intell Syst*. 2009;24(7):766-792.
11. Ali R, Muhammad S, Takahashi R. Decision making vivagenetic algorithm for the utilization of leftovers. *Int J Intell Syst*. 2021:1-24.
12. Muthu ABA, Enoch S. Optimized scheduling and resource allocation using evolutionary algorithms in cloud environment. *Int J Intell Eng Syst*. 2017;10(5):125-133.
13. Mirjalili S. Genetic algorithm. In: *Evolutionary algorithms and neural networks*. Springer; 2019:43-55.
14. Banks A, Vincent J, Anyakoha C. A review of particle swarm optimization. Part I: background and development. *Natural Comput*. 2007;6(4):467-484.
15. Iacca G, Neri F, Caraffini F, Suganthan PN. A differential evolution framework with ensemble of parameters and strategies and pool of local search algorithms. In: *European Conference on the Applications of Evolutionary Computation*. Springer; 2014:615-626.
16. Caraffini F, Neri F, Poikolainen I. Micro-differential evolution with extra moves along the axes. In: *2013 IEEE Symposium on Differential Evolution (SDE)*. IEEE; 2013:46-53.
17. Caraffini F, Neri F. A study on rotation invariance in differential evolution. *Swarm Evol Computat*. 2019;50: 100436.
18. Kóczy LT, Földesi P, Tüű-Szabó B. An effective discrete bacterial memetic evolutionary algorithm for the traveling salesman problem. *Int J Intell Syst*. 2017;32(8):862-876.
19. Elsisi M. Optimal design of nonlinear model predictive controller based on new modified multitracker optimization algorithm. *Int J Intell Syst*. 2020;35(11):1857-1878.
20. Massobrio R, Toutouh J, Nesmachnow S, Alba E. Infrastructure deployment in vehicular communication networks using a parallel multiobjective evolutionary algorithm. *Int J Intell Syst*. 2017;32(8):801-829.
21. Ezhilarasi M, Krishnaveni V. An evolutionary multipath energy-efficient routing protocol (EMEER) for network lifetime enhancement in wireless sensor networks. *Soft Comput*. 2019;23(18):8367-8377.
22. Kuwahara M, Yoshioka A, Uno N. Practical searching optimal one-way carsharing stations to be equipped with additional chargers for preventing opportunity loss caused by low SoC. *Int J Intell Trans Syst Res*. 2020:1-10.
23. Hu Y, Gao J, Chen X, Meng F, Wang YH. Distribution planning of UAV automatic charging station based on genetic algorithm. In: *2019 International Conference on Economic Management and Model Engineering (ICEMME)*. IEEE; 2019:446-452.
24. Mirzaeinia A, Mirzaeinia M, Bradfield QA, Bradley S, Hassanalian M. Particle swarm optimization for wireless charging of swarming drones through ambient radio frequencies. In: *AIAA Propulsion and Energy 2019 Forum*; 2019:4463.
25. Xu M, Meng Q. Optimal deployment of charging stations considering path deviation and nonlinear elastic demand. *Trans Res Part B: Methodol*. 2020;135:120-142.
26. Bouguerra S, Layeb SB. Determining optimal deployment of electric vehicles charging stations: case of Tunis City, Tunisia. *Case Studies Transport Policy*. 2019;7(3):628-642.
27. Zhang Y, Zhang Q, Farnoosh A, Chen S, Li Y. GIS-based multi-objective particle swarm optimization of charging stations for electric vehicles. *Energy*. 2019;169:844-853.
28. Pan A, Zhao T, Yu H, Zhang Y. Deploying public charging stations for electric taxis: A charging demand simulation embedded approach. *IEEE Access*. 2019;7:17412-17424.

29. Luo Z, He F, Lin X, Wu J, Li M. Joint deployment of charging stations and photovoltaic power plants for electric vehicles. *Trans Res Part D: Trans Environ*. 2020;79:102247.
30. Akbari M, Brenna M, Longo M. Optimal locating of electric vehicle charging stations by application of genetic algorithm. *Sustainability*. 2018;10(4):1076.
31. Vazifeh MM, Zhang H, Santi P, Ratti C. Optimizing the deployment of electric vehicle charging stations using pervasive mobility data. *Transportation Research Part A: Policy and Practice*. 2019;121:75-91.
32. Priyadarshi R, Gupta B. Coverage area enhancement in wireless sensor network. *Microsystem Technol*. 2020;26(5):1417-1426.
33. Simon D. *Evolutionary Optimization Algorithms*. John Wiley & Sons; 2013.
34. Jansen T. *Analyzing Evolutionary Algorithms: The Computer Science Perspective*. Springer Science & Business Media; 2013.
35. Karafotias G, Hoogendoorn M, Eiben ÁE. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Trans Evol Computat*. 2014;19(2):167-187.
36. Bonyadi MR, Michalewicz Z. Particle swarm optimization for single objective continuous space problems: a review. *Evol Computat*. 2017;25(1):1-54.
37. Sinha A, Malo P, Deb K. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Trans Evol Computat*. 2017; 22(2): 276-295.
38. Ma H, Simon D, Siarry P, Yang Z, Fei M. Biogeography-based optimization: a 10-year review. *IEEE Trans Emerging Topics Computat Intell*. 2017;1(5):391-407.
39. Niccolai A, Grimaccia F, Mussetta M, Gandelli A, Zich R. Social network optimization for wsn routing: analysis on problem codification techniques. *Mathematics*. 2020;8(4):583.

**How to cite this article:** Niccolai A, Bettini L, Zich R. Optimization of electric vehicles charging station deployment by means of evolutionary algorithms. *Int J Intell Syst*. 2021; 36:5359-5383. https://doi.org/10.1002/int.22515