Wireless Networks
https://doi.org/10.1007/s11276-022-03068-9

ORIGINAL PAPER

Memetic algorithm applied to topology control optimization of a wireless sensor network

Jorge A. G. de Brito[1] · Diego R. M. Totte[1] · Fábio O. Silva[1,2] · Jurair R. de P. Junior[1] · Felipe da Rocha Henriques[1] · Luís Tarrataca[1] · Diego Barreto Haddad[1] · Laura S. de Assis[1]

Accepted: 29 June 2022

Abstract
In most applications of wireless sensor networks the specification of the corresponding topology can be useful for the optimization of some important features, such as: node energy consumption, connectivity and coverage area. This is known as the Sensor Allocation Problem (SAP). Our work proposes an approach based on memetic algorithm concepts to find high-quality solutions. In our approach, each node can be associated with one of four operation modes (classified according to its maximum range). The algorithm optimizes the position of each node and produces solution clusters. In order to evaluate the efficiency of the method, we analyze case studies with different coverage areas that are then compared against results previously found in the literature. Our experiments show that in order to achieve a smaller energy consumption and an increase in network coverage area, one needs to operate with a sizeable number of sensors, but with few nodes operating in larger transmission power modes (which require an increased energy expenditure).

Keywords Wireless Sensor Network · Combinatorial Optimization · Memetic Algorithms

1 Introduction

Wireless Sensor Networks (WSNs) are self-configured entities composed by a set of sensors (distributed along an area of interest to be monitored) which can freely communicate over short distances [1]. The popularity of WSNs is in part attributed to their advantages of distributed control, scalability and self-organization [2, 3]. Accordingly, they are an important technology for a wide range of real-time applications [4–6]. In a WSN each sensor can perform the following basic tasks: *(i)* sensing environmental factors; *(ii)* data storage; *(iii)* processing; and *(iv)* wireless communication. Each node can collect environmental signals, such as temperature, humidity, pressure, and illumination [7], which can be used for distributed inference purposes [8–10]. These pieces of information are forwarded from the sensor nodes to a sink node, that is, in general, computationally more powerful and has unrestricted access to energy.

A large-scale WSN may consist of hundreds of sensor nodes able to transmit the information gathered without requiring complicated communication backbones [11]. The nodes are usually battery-powered, employ low-priced and

✉ Jorge A. G. de Brito
jagb90@gmail.com

✉ Laura S. de Assis
laura.assis@cefet-rj.br

Diego R. M. Totte
diego.totte@aluno.cefet-rj.br

Fábio O. Silva
fabio.oliveira@ifba.br

Jurair R. de P. Junior
jurair.junior@cefet-rj.br

Felipe da Rocha Henriques
felipe.henriques@cefet-rj.br

Luís Tarrataca
luis.tarrataca@cefet-rj.br

Diego Barreto Haddad
diego.haddad@cefet-rj.br

[1] Federal Center of Technological Education of Rio de Janeiro - CEFET/RJ, Petropolis, Rio de Janeiro, Brazil

[2] Federal Institute of Bahia, Eunapolis, Bahia, Brazil

Published online: 12 August 2022

computationally constrained hardware, have a limited energy budget (which may be irreplaceable [12]) and may be organized in a determined or random manner [13–15]. Due to their limited power supply, energy management is an important issue for WSN [7, 16, 17]. Energy-constrained operation has become one of the major bottlenecks holding back further dissemination and development of WSNs [18]. WSNs can be, in theory, deployed with low costs, due to the fact that they do not require fixed infrastructures [19]. However, this can only be achieved through a judicious sensor allocation strategy. In short, one of the most important issues in WSNs is the achievement of better system performance (*e.g.*, network lifetime) by sensor deployment strategies[1] [1].

Furthermore, WSNs should be capable of preventing problems, such as a sensor failure [14]. Namely, when a sensor node fails, the WSN should establish a new route to the sink, whilst keeping connectivity and preserving autonomy. WSNs commonly use a cluster topology, that divides the sensing area into small groups. Each cluster has a data fusion node, denominated the cluster–head. The nodes belonging to a cluster can collect data and send information directly to the cluster–head, which is responsible for storing, processing and providing data gathered by its one–hop neighboring nodes. Generally, this type of topology has the following objectives: *(i)* to save energy; *(ii)* to be fault-tolerant; *(iii)* to ensure efficient network communication; and *(iv)* to ensure efficient data dissemination [20]. In addition, in order to ensure good area coverage sensing the sensor nodes must be placed in such a way as to cover the whole interest region. Guaranteeing every node in the WSN has connectivity, whilst keeping a low energy consumption, are thus important issues for network topology [21].

The aforementioned concepts of energy consumption, connectivity and coverage area have real-world applications. Namely, in [22], the authors investigated the effects of signal propagation on WSNs applied to landslide management. Transmitting and receiving antennas were placed in order to perform real-world experiments. In [23], a healthcare system based on WSNs was described. The authors proposed a real-time heart pulse monitoring system, collecting data from distinct people, whose data was forwarded to smartphones. In [24] the authors implemented a novel method for Structural Health Monitoring (SHM) applications as a mechanism to perform structural damage detection. The proposed system is wireless and works in real-time. The work also describes the set of laboratory tests that were performed as a way to evaluate the proposal. The work presented in [25] detailed and implemented a Software Defined Network-based energy efficient routing protocol for WSN applications. The work was evaluated through a real test bed, which made use of a Raspberry Pi. The results showed that this strategy outperformed other routing protocols found in the literature with respect to metrics such as network lifetime, energy consumption, packet delivery ratio and average delay.

[1] In the literature, such techniques are also known as placement or coverage schemes [6].

1.1 Objectives and contributions

This paper proposes a mathematical model and the implementation of a methodology based on memetic algorithm (MA) concepts [26–28]. Our method aims to define the best configuration of operation modes for the sensor nodes belonging to a WSN. Our objective is to optimize network efficiency in terms of coverage area and energy consumption. This is a challenging combinatorial optimization problem, one where complexity increases exponentially as the instance (measured in terms of coverage area) grows. Solving this problem allows for greater monitoring autonomy since each operation mode defines the maximum range of a given sensor, which is based on its transmission power. This is directly related to network energy consumption, since higher transmission power requires larger energy expenditure.

In order to evaluate the performance of the MA–based proposed methodology, our work considers several case studies with cover areas of different sizes (in grids). This approach is based on [29–31]. In addition to extending the analysis and the results described in these papers, the main contributions of our work are the following:

- Proposal of a mathematical model of integer linear programming for the aforementioned SAP. This extends the works [29, 31], which only presented the objective functions of the considered optimization problem;
- Proposal of MA-based approach that can be reproduced and adapted to other applications with different objective functions (OF). This is in contrast to the works [29, 31], which only considered a genetic algorithm (GA) method;
- Proposal evaluation in grid-based instances with distinct areas, reflecting more realistic scenarios. This differs from the works [29, 31], which considered only small 10×10 grid-based instances.

The experimental results obtained indicate that the SAP solution proposal can allocate sensor nodes in optimal positions so as to minimize energy consumption whilst retaining network connectivity. It is important to emphasize that there is a trade-off between energy consumption and sensing coverage, which calls for the establishment of an operational choice during network design [6].

1.2 Structure

The remainder of this paper is organized as follows. Section 2 presents the related works and also highlights the contributions of the proposed method against the existing literature. The problem studied and respective mathematical model are presented in Sect. 3. Section 4 describes the methodology for the proposed solution. Simulation results are analyzed and discussed in Sect. 6. Concluding remarks and future works are presented in Sect. 7.

2 Related works

Connectivity is crucial for the functionalities and capabilities of WSNs [32]. For example, if all wireless devices have the same transmission radius r in homogeneous ad hoc networks, one has r-disk graphs [19]. Target coverage and connectivity are critical issues in WSNs [33]. For instance, reference [34] proposes the utilization of a local search approximation algorithm for connectivity-aware deployment algorithms able to generate a reasonable node placement. Two centroid-based iterative deployment schemes (*i.e.,* blind-zone centroid-based and disturbed centroid-based schemes) for sensor coverage in WSNs are proposed in [35]. None of these schemes require manual adjustments of their parameters.

A low-cost and connectivity-guarantee grid-based deployment mechanism that employs ant colony optimization metaheuristics is presented in [36]. Network topology control and network coding are jointly considered in [37] in order to reduce WSN energy consumption. Despite the deployment problem being NP-complete [38] it was mitigated in [39] by making use of a weighted sampling scheme that matches the requested detection probabilities. The minimization of the number of sensor nodes is performed by using the local event occurrence rate while employing the sensors detection capabilities [1].

It is important to emphasize which features this paper does *not* consider. Note, for example, that the impact of routing protocols is not taken into consideration. This is an important issue in when modelling the cascading feature present in some WSNs (see [40]). Another feature that is not considered is the implementation of behavioral monitoring of sensor nodes, by selecting some of them as watchdogs. These may be considered as a possible countermeasure to attacks such as selective forwarding, that are able to detect misbehaving nodes [41]. Furthermore, specific strategies to reduce packet transfers are not taken into account. As an example, one may cite duty-cycling schemes (which merges sleep intervals with wake-up time intervals [42]) and cluster node division, with the respective cluster-heads sending the average values directly to the sink [43].

In order to address the lack of adaptability to event dynamics, several papers exploit the mobility of sensor nodes [44]. Sink mobility is exploited in [45] as a way to better balance the traffic load across the sensor nodes and increase network lifetime. Hybrid sensor networks (*i.e.,* which include both static and mobile sensor nodes) are addressed in [46]. Both sensor faults and sensor noise are considered in [47], which advocates the employment of a *weighted* average of sensor measurements. The combination of WSNs with free space optical communication technology is addressed in [3]. Since a good understanding of the propagation impairments affecting the wireless links may contribute to a successful design of WSNs, some papers propose models for the involved path losses (e.g., [48]). In [49] the authors address the problem of coalition formation among devices when considering energy availability, communication interest and physical ties. The work proposes a distributed power control framework for determining the optimal power characteristics.

Evolutionary strategies have been employed to address challenges in WSNs. Namely, in [50] the authors present an evolutionary game and respective evolution strategy for addressing the joint spectrum sensing and access problem. A review of computational intelligence techniques applied to the challenges that arise from WSNs was presented in [51]. The authors describe, categorize and present taxonomies for the state of the art in terms of fuzzy systems, neural networks, evolutionary computation, swarm intelligence, learning systems and their respective hybridizations (amongst others). An optimization strategy based on brain storm optimization was proposed in [52]. The authors analyze the factors influencing this mechanism and then improve on it by proposing an orthogonal learning framework to improve its learning mechanism. A framework for solving large-scale multiobjective and many-objective optimization problems was presented in [53]. In this work, the authors propose new algorithms that: *(i)* incorporate reference vectors into the control variable analysis; and *(ii)* optimize the decision variable by making use of an adaptive scalarization strategy. The offload of computationally intensive tasks and associated challenges from WSNs to accessible servers was addressed in [54].

MAs have been explored to optimize network lifetime and fault tolerance. Simulation results from [55] showed that the MA outperformed other heuristic methods in terms of WSN lifetime. The MA-TOSCA proposed in [56] aims to optimize network topology as a way to provide tolerance against cascading failures. For this purpose, a local search operator was designed based on a new network balancing metric.

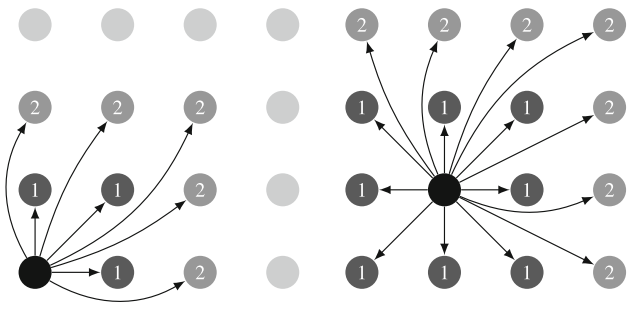
In the context of this work, it is important to note that a finite number of sensor modalities are employed, each of them with a specific communication range (see Sect. 3). Such a sensing model implies that the considered WSNs are heterogeneous (as in a heterogeneous network [57]), making their topology control and deployment more difficult [6, 58]. This advanced node deployment can be classified as grid-point based, which presents several advantages against continued-point alternatives [59].

3 Mathematical model

Using graph theory terminology [60], a two-dimensional area for sensor allocation can be modelled as a connected graph, where V is the set of n nodes and A is the set of m arcs. A node $i \in V$ represents a possible location to place a sensor. The arc $(i,j) \in A$, $i \in V$ and $j \in V$, represents the possible communication link between two sensors.

Figure 1 exemplifies the set of existing arcs between the black-node and the other nodes of the network reachable by it. In order to facilitate the visibility, only the arcs that are incident to one specific node are depicted. The same process must be repeated for each node of the graph in order to obtain a comprehensive description of the network. According to the sensors type considered in this work, the maximum achievable range is two nodes (or two-hop) distant. Thus, each node of the graph is adjacent, at best, to a node that is at this transmission range (distance) from it, as depicted by the number presented in each node (Fig. 2).

The SAP proposed in this article was based on the simplified problem presented in [29]. However, this work proposes a mathematical formulation able to provide clear specifications to the solution search procedure. Namely, the model uses a set of binary decision variables s_i^t, with $t \in S$ and $i \in V$, defined as follows:



(a) Sample 1. **(b)** Sample 2.

Fig. 1 Partial Graph representation of a two-dimensional area with size $L \times L = 4 \times 4$, exemplifying the maximum range for the black-nodes

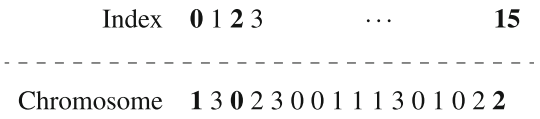


Fig. 2 Chromosome representation of solution depicted in Figure 3

$$s_i^t = \begin{cases} 1, & \text{if a sensor of operation mode t is placed at the } i\text{-th node} \\ 0, & \text{otherwise,} \end{cases}$$

where the set $S = \{x, y, z, w\}$ represents all sensor types used in the study, defined by their communication range, and $i \in \{1, \ldots, n\}$, being $n = L \times L$. Nodes with z-sensors (one-hop) and y-sensors (two-hops) communicate with x-sensor nodes, and w-sensor nodes represent inactive operation sensors, which may be in sleep mode or even be absent. L is the one-dimension size of the considered square field. The SAP can be formulated as the following integer linear programming problem:

$$\min \sum_{j=1}^{o} \alpha_j f_j \tag{1}$$

$$st.$$

$$\sum_{t \in S} s_i^t \leq 1 \qquad \forall\, i \in V \tag{2}$$

$$\sum_{i \in V} s_i^x \geq 1 \tag{3}$$

$$\sum_{i \in V} \sum_{t \in S} s_i^t = L \times L \tag{4}$$

$$s_i^t \in \{0, 1\} \qquad \forall\, i \in V, \quad \forall\, t \in S \tag{5}$$

where $|S|$ is the number of sensor types (operation mode) considered, o is the number of OFs considered, α_j is the parameter that weighs the j-th OF and f_j is the j-th OF.

We decided to map the textual descriptions of the metrics employed in [29, 30] into the following set of formulas that can be employed into the proposed linear programming problem:

1. $f_1 = FC$: this OF represents the coverage and communication between the nodes (maximization function).

$$FC = \frac{\left(\sum_{i \in V} s_i^x + \sum_{i \in V} s_i^y + \sum_{i \in V} s_i^z\right) - \left(\sum_{i \in V} s_i^w + N_{OR}\right)}{n}, \tag{6}$$

   where each individual sum is responsible for presenting the number of x-, y-, z- and w-sensors allocated in the solution, and the N_{OR} term represents the number of sensors of types y and z that are out of range of a sensor of type x (see Eq. 11).
2. $f_2 = OpCiE$: this OF controls whether a cluster in charge (x-sensor) is well distributed to minimize overlaps.

$$OpCiE = \frac{N_{overlaps}}{\sum_{i \in V} s_i^x}, \tag{7}$$

where $N_{overlaps}$ (number of overlaps, see Eq. 14) represents the number of y- and z-sensors, reachable for more than one sensor of mode x (minimization function).

3. $f_3 = SORE$: this OF verifies how many sensors are out of range of a x-sensor (cluster in charge), which depends directly on the communication range of each sensor and its position in the network. It is assumed that the y-mode sensors have a circular cover equal to $2\sqrt{2}$ units in length and the z-mode sensors have a circular cover equal to $\sqrt{2}$ units in length (minimization function).

$$SORE = \frac{N_{OR}}{\sum_{i \in V} s_i^w}. \tag{8}$$

Figure 3 exemplifies an instance of the positions for a set of different sensors. Node $a_{1,1}$, corresponding to an x-sensor, distances one horizontal unit from node $a_{1,2}$, allocated to an z-sensor. In the same manner, node $a_{1,1}$ is two vertical units and two horizontal units apart from node $a_{3,3}$.

4. $f_4 = SpCi$: this OF verifies that the y- and z-sensors are in accordance with the traffic capacity, data management and the physical communication capacity of the x-sensors (maximization function).

$$SpCi = \frac{\sum_{i \in V} s_i^y + \sum_{i \in V} s_i^z - N_{OR}}{\sum_{i \in V} s_i^x}. \tag{9}$$

5. $f_5 = NE$: this OF is a numerical measure of energy consumption that depends directly on the network topology and operating mode of each sensor. It is assumed that the x-sensor consumes twice what y-sensor does and four times more than the z-sensor [29] (minimization function).

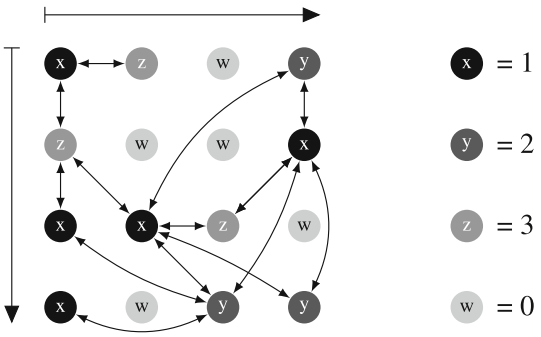


Fig. 3 Example of solution with $L = 4$

$$NE = \frac{4\sum_{i \in V} s_i^x + 2\sum_{i \in V} s_i^y + \sum_{i \in V} s_i^z}{n}. \tag{10}$$

6. N_{OR} (number of sensors that are out of range), can be defined as:

$$N_{OR} = N_{OR}^y + N_{OR}^z, \tag{11}$$

where N_{OR}^y and N_{OR}^z are the number of y-sensors and z-sensors that are out of range of a sensor of type x, respectively. They are defined in Eqs. 12 and 13.

$$N_{OR}^z = \sum_{i \in V} 1 - \min\left(1, \sum_{k \in N_i^z} s_k^x\right), \tag{12}$$

$$N_{OR}^y = \sum_{i \in V} 1 - \min\left(1, \sum_{k \in N_i^y} s_k^x\right), \tag{13}$$

where N_i^z is the set of nodes in the neighborhood of a z-sensor placed in the i-th node of the network, and N_i^y is the analog for y-sensor. Both neighborhoods are specified in detail by Eqs. 17-19.

7. $N_{overlaps}$ is the number of z- and y-sensors that have more than one x-sensor in their range. The number of overlaps can be defined as:

$$N_{overlaps} = N_{overlaps}^z + N_{overlaps}^y, \tag{14}$$

where $N_{overlaps}^z$ (Eq. 15) and $N_{overlaps}^y$ (Eq. 16) are the number of overlapped z- and y-sensors, respectively:

$$N_{overlaps}^z = \sum_{i \in V} \left[\left(|S_i^z - N^x(N_i^z)| \right) \times S_i^z \times \left\lceil \frac{N^x(N_i^z)}{N^x(N_i^z) + 1} \right\rceil \right], \tag{15}$$

$$N_{overlaps}^y = \sum_{i \in V} \left[\left(|S_i^y - N^x(N_i^y)| \right) \times S_i^y \times \left\lceil \frac{N^x(N_i^y)}{N^x(N_i^y) + 1} \right\rceil \right], \tag{16}$$

$$N^x(N_i^z) = \sum_{k \in N_i^z} S_k^x, \tag{17}$$

$$N^x(N_i^y) = \sum_{k \in N_i^y} S_k^x. \tag{18}$$

Equations 17 and 18 define the number of x-sensors that are present in a z- and y-sensor range, where N_i^z and N_i^y are, respectively, the neighborhood of a z-sensor and y-sensor placed in the i-th node. In this paper, the neighborhood of a z-sensor placed in the i-th node is the set of nodes located one-distance from such a node (see Fig. 1, nodes with flag 1). Similarly, the neighborhood of a y-sensor allocated in the i-th node is the set of nodes placed two-distance from such a node (see Fig. 1, nodes with flag 2). For a two-

dimensional area, such as the one considered in the present work, the neighborhood of size one can be defined as:

$$N_i^z = (((q-1)L) + u) \qquad \forall \;\; q \neq R \; or \; u \neq C \;, \tag{19}$$

where:

- $u \in \{\max\{1, \cdots, C-1\}, \cdots, \min(C+1, L)\}$,
- $q \in \{\max\{1, \cdots, R-1\}, \cdots, \min(R+1, L)\}$,
- $C = (i-1) \bmod L$,
- $R = \lfloor (i-1)/L \rfloor$ and $i = \{1, 2, \cdots, L \times L\}$, *i.e.,* $i \in V$.

N_i^z in Eq. 19 is the neighborhood of size 1 from a node i, *i.e.*, one-hop distance. From this definition it is possible to calculate the neighborhood of any size for a given node. According to graph theory, we can use the concept of reachability with the recursive process shown in Eqs. 20-23 to determine the neighborhood of size n of a given node v [61]:

$$N^1(v) = N(v), \tag{20}$$

$$N^2(v) = N(N^1(v)), \tag{21}$$

$$\vdots \quad \vdots$$

$$N^n(v) = \; N(N^{(n-1)}(v)) \tag{22}$$

$$N^n(v) = \bigcup_{i=0}^{n} N^i(v), \tag{23}$$

where $N^1(v)$ is the neighborhood of size 1 of a node v, thus $N^n(v)$ is the neighborhood of size n of a node v, *i.e.,* the union of all neighbor sets of size smaller than n (Eq. 23). For example, in this work, it is possible to find the neighborhood of a y-sensor placed in the i-th node, through its z-sensor neighborhood when it is allocated in the i-th node. Thus $N_i^y = N(N_i^z)$ is the neighborhood of size 2 from a node i, *i.e.,* the set of neighboring nodes of size 1 and the neighbors of its neighbors.

Furthermore, the of (1) minimizes the network optimization function described above[2]. Constraint (2) ensures that only one sensor can be installed on each node (possible position). Equation 3 guarantees that at least one x-sensor will be allocated in the solution. Equation 4 assures that the number of sensors (considering the inactive ones) will be equal to the coverage area size. Constraint (5) defines all variables as binaries.

[2] To obtain more specific information about the optimization parameters, please refer to [29].

4 Solution methodology

The principle behind heuristics and metaheuristics methods is to provide quality solutions for the problem addressed in a reasonable computational time. The solution method developed for the instance of the SAP being tackled is based on GA concepts and local search optimization procedures. These methods allow for high-quality solutions to be found, albeit not guaranteeing an optimal solution. However, empirical studies have shown that such techniques have good performance in practice [62–65].

4.1 Proposed memetic algorithm

GAs are optimization algorithms that work in a randomly oriented manner according to probabilistic rules which are based on an analogy with the Darwinian principles of species evolution and genetics [66, 67]. MAs were introduced in [68] as a variation of GA, where the term "meme" denotes a cultural evolution and not only a gene evolution.

MA is a search strategy in which a population of individuals that are being optimized cooperate and compete. Each individual represents a solution to the addressed problem. A candidate solution, not necessarily feasible, is characterized by a chromosome representing a gene sequence that encodes a possible solution. This coding is defined by specifying the possible values (alleles) for each chromosome positions (locus), arranging them in an appropriate structure to the problem [67]. In the case of our methodology, each individual of the population is a solution to the SAP, which is represented by a chromosome consisting of integer numbers. The main memetic algorithm concepts are presented below, namely: Sect. 4.1.1 describes the representation employed for each solution; Sect. 4.1.2 details how to measure the intrinsic value of each solution; Sect. 4.1.3 presents the pseudocode for generating a feasible initial population. Section 4.1.4 characterizes the selection operator used to create an intermediate population; Sect. 4.1.5 expands on how to perform gene permutation; Sect. 4.1.6 reveals a mutation operator responsible for introducing gene diversity; Sect. 4.1.7 elaborates on how to make infeasible generated solutions into feasible ones; Sect. 4.1.8 puts forward the local search procedure that is applied to the offspring. Finally, Sect. 4.1.9 introduces the pseudocode for the memetic algorithm implemented.

4.1.1 Solution encoding

A chromosome representation for the SAP can be built using a vector of integer numbers with dimension one and

size $L \times L$. The length of the vector symbolizes the two-dimensional monitoring region (size $L \times L$). Each vector position is associated with a location where a sensor can be installed. The value assigned to each chromosome position is in the interval [0, 3], representing the operating mode (*w*, *x*, *y* and *z*) of each sensor. Value 0 means that the node is inactive (*i.e., w*-sensor); value 1 corresponds to operating mode *x* (cluster in charge), value 2 represents operating mode *y* and value 3 operating mode *z*. Figure 2 exemplifies the chromosome for the solution encoding presented in the Fig. 3, which shows the position and communication between each sensor. Each node presented in Fig. 3 has a specific $a_{i,j}$ position where *i* and *j* correspond, respectively, to a row and column of the $L \times L$ matrix. The chromosome contains the information $a_{1,1}, \cdots, a_{1,L}, a_{2,1}, \cdots, a_{2,L}, \cdots, a_{L,1}, \cdots, a_{L,L}$.

4.1.2 Fitness function

In each generation the chromosome of every individual is evaluated by calculating its fitness value. This measure is computed after the creation of the initial population and recalculated after: *(i)* the genetic operators have been applied; and *(ii)* the local search procedure has been performed. As is typical in minimization problems, the fitness function adopted in this paper is the inverse of the OF, as presented in Eq. 24 (for more details please refer to Sect. 3).

$$fitness = \frac{1}{\sum_{i=1}^{n} \alpha_i F_i}. \tag{24}$$

4.1.3 Initial population

A randomized constructive heuristic is employed to generate a feasible initial population, as described in Algorithm 1. The procedure begins with an empty population (line 1), and each main iteration (lines 2-19) constructs one individual at a time. Each individual (*I*) starts without any sensors (lines 3). In the inner loop, (lines 5-17), trials are performed to allocate a sensor to every possible location. The operation mode of the sensor assigned to a particular position is determined according to a predetermined probability. Namely, *y*-sensors have the highest occurrence probability, this is then followed by *z*-sensors, then *w*-sensors and the least likely *x*-sensors. The procedure returns a population with P_{size} initial individuals. It should be emphasized that population size (P_{size}) and the individual size (c_s, *i.e.,* chromosome size) are parameters of the algorithm. Algorithm 1 executes in $\Theta(P_{size} \times c_s)$ time.

Algorithm 1 GenerateInitialPopulation(P_{size}, c_s)

```
1:  P ← ∅
2:  while (|P| ≤ P_size) do
3:    I ← ∅
4:    i ← 0
5:    for (i < c_s) do
6:      R ← Random(0, 1)
7:      if (R ⩾ 0) and (R ⩽ x_rate) then
8:        I_i ← x_sensor
9:      else if (R > x_rate) and (R ⩽ w_rate) then
10:       I_i ← w_sensor
11:     else if (R > w_rate) and (R ⩽ z_rate) then
12:       I_i ← z_sensor
13:     else
14:       I_i ← y_sensor
15:     end if
16:     i ← i + 1
17:   end for
18:   P ← P ∪ {I}
19: end while
20: return P
```

4.1.4 Selection

The goal of the selection operator is to choose pairs of individuals in the current generation population to perform the crossover. The operator implemented in this paper is based on `Roulette Selection`, ensuring that all individuals have a certain probability of being chosen that is proportional to their fitness. Table 1 shows the fitness of four individuals and Fig. 4 illustrates the respective selection operator implemented. The roulette rotates as many times as necessary to select the required number of individuals to participate of the crossover.

4.1.5 Crossover

The purpose of the crossover operator is to generate an intermediate population through genes permutation of the parents that were produced from the selection operation. A two-point crossover is applied, in which two cutting points

Table 1 Example of fitness and accumulated fitness by individual

Individual i	**Fitness** f_i	**Accumulated fitness** $\sum_{i=1}^{n} f_i$
1	2.0	2.0
2	1.5	3.5
3	4.0	7.5
4	2.5	10.0

are randomly selected to split the chromosomes from two parents. These points determine what genetic information will be copied from each parent to generate an offspring. Each offspring takes one segment, in between adjacent cut points, from each parent, thus each parent contributes with part of its genes to generate a new individual. This procedure is exemplified in Fig. 5.

Algorithm 2 shows the pseudocode for the crossover operation. I_1 and I_2, are the two individuals selected to participate in the crossover, c_s is the chromosome size and I_o is the offspring resulting from the crossover. The algorithm starts by selecting two cut points (lines 1-2). These points are sorted in ascending order (lines 3-7). The offspring is then composed by the first part of individual 1, followed by the second part of individual 2 and, lastly, by the third part of individual 1 (lines 8-16). The algorithm executes in $\Theta(c_s)$ time.

Algorithm 2 Crossover(I_1, I_2, c_s)

```
1:  p_cut1 ← Random(1, c_s)
2:  p_cut2 ← Random(1, c_s)
3:  if p_cut1 > p_cut2 then
4:      temp ← p_cut1
5:      p_cut1 ← p_cut2
6:      p_cut2 ← temp
7:  end if
8:  for i ← 1 to p_cut1 do
9:      I_o[i] ← I_1[i]
10: end for
11: for i ← p_cut1 + 1 to p_cut2 do
12:     I_o[i] ← I_2[i]
13: end for
14: for i ← p_cut2 + 1 to c_s do
15:     I_o[i] ← I_1[i]
16: end for
17: return I_o
```

4.1.6 Mutation

The mutation operator is responsible for introducing diversity in the current population. This is performed by changing the genetic code of the constituent individuals and happens in accordance with a mutation rate, which dictates whether an offspring can mutate. When a locus is selected for mutation the method considers four possibilities:

1. **position contains x-sensor:** a *y*-sensor, *z*-sensor or *w*-sensor is allocated instead of a *x*-sensor;

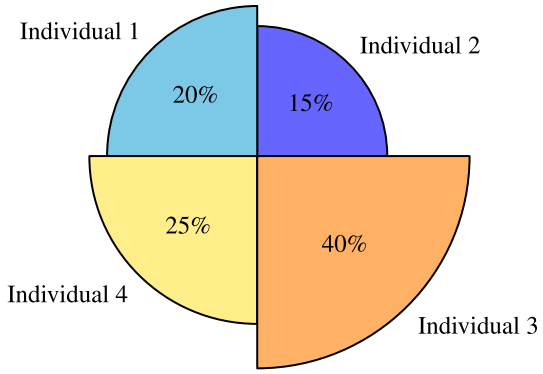


Fig. 4 Roulette selection example

$parent_1$	130	230011	1301022
$parent_2$	220	320100	2201330
offspring	130	320100	1301022

Fig. 5 Example of two-point crossover

2. **position contains y-sensor:** a *x*-sensor, *z*-sensor or *w*-sensor is allocated instead of a *y*-sensor;
3. **position contains z-sensor:** a *x*-sensor, *y*-sensor or *w*-sensor is allocated instead of a *z*-sensor;
4. **position contains w-sensor**: a *x*-sensor, *y*-sensor or *z*-sensor is allocated instead of a *w*-sensor.

The choice of which sensor will replace the previous one obeys the probability distribution mentioned in Sect. 4.1.3. Figure 6 exemplifies a possible mutation operation.

The pseudocode of the mutation operation is presented in Algorithm 3. I_o is the offspring resulting from the crossover, m_p is the probability of the mutation and c_s is the size of the chromosome. According to a certain probability, each chromosome allele can be mutated (line 4). Whenever an allele is chosen to be mutated, it is verified which sensor is allocated at this position and one of the other three modes of operation is chosen (lines 5-13). Similarly to the crossover pseudocode, Algorithm 3 also executes in $\Theta(c_s)$ time with the ChooseSensor function requiring $\mathcal{O}(1)$ time.

4 SOLUTION METHODOLOGY 15

Before mutation 13**0**3201001**1**30102**2**

After mutation 13**2**3201003**3**30102**0**

Fig. 6 Example of a mutation operation for an offspring instance

Algorithm 3 Mutation(I_o, m_r, c_s)

```
1:  I_m ← I_o
2:  for i ← 1 to c_s do
3:      R ← Random(0, 1)
4:      if R > m_p then
5:          if I_o[i] == y_sensor then
6:              I_m[i] ← ChooseSensor(z_sensor, w_sensor, x_sensor)
7:          else if I_o[i] == z_sensor then
8:              I_m[i] ← ChooseSensor(y_sensor, w_sensor, x_sensor)
9:          else if I_o[i] == w_sensor then
10:             I_m[i] ← ChooseSensor(y_sensor, z_sensor, x_sensor)
11:         else if I_o[i] == x_sensor then
12:             I_m[i] ← ChooseSensor(y_sensor, z_sensor, w_sensor)
13:         end if
14:     end if
15: end for
16: return I_m
```

4.1.7 Feasibility

The crossover and mutation operations may yield an individual that does not respect Eq. 3, *i.e.,* at least one *x*-sensor needs to be allocated to the cover area. When this happens the individual is said to be infeasible and a feasibility procedure needs to be carried out. This can be done by randomly selecting a *locus* and substituting by an *x*-sensor.

4.1.8 Local search

MAs are categorized as hybrid Evolutionary Algorithms (EA) and are very successful in practice, forming a rapidly growing research area with great potential. Commonly combining an evolutionary and a heuristic method (a hybrid EA) performs better than each one of those algorithms alone [69]. In this kind of algorithms, the evolutionary search is augmented by the addition of one or more phases of local search.

This MA phase aims to examine a set of neighbors of the current solution and replace it with a better one, if it exists. Therefore, after performing the crossover and the mutation operations, a first improvement local search (LS) procedure is applied in the offspring. The LS proposed is based on an exchange movement. For each chromosome position of the individual, the LS procedure replaces the current sensor for a different one. The choice is made according to the probability presented in the mutation operator (see Sect. 4.1.6). When the best individual does not improve over a specific number of generations, the local search is also applied to the best individual of the current population. The pseudocode for the LS can be found in Algorithm 4. The while cycle is responsible for $\mathcal{O}(c_s)$ iterations. The Fitness function that is being invoked in Line 14 grows linearly with the size of the individual, and thus requires $\Theta(c_s)$ time. In addition, the ChooseSensor function executes in $\mathcal{O}(1)$ time. Overall, Algorithm 4 executes in $\mathcal{O}(c_s^2)$ time.

Algorithm 4 localSearch(I_o, c_s)

```
1:  I_ls ← I_o
2:  improve ← false
3:  i ← 1
4:  while i ≤ c_s and Not improve do
5:    if I_o[i] == y_sensor then
6:      I_ls[i] ← ChooseSensor(z_sensor, w_sensor, x_sensor)
7:    else if I_o[i] == z_sensor then
8:      I_ls[i] ← ChooseSensor(y_sensor, w_sensor, x_sensor)
9:    else if I_o[i] == w_sensor then
10:     I_ls[i] ← ChooseSensor(y_sensor, z_sensor, x_sensor)
11:   else if I_o[i] == x_sensor then
12:     I_ls[i] ← ChooseSensor(y_sensor, z_sensor, w_sensor)
13:   end if
14:   if Fitness(I_ls) ¿ Fitness(I_o) then
15:     improve ← true
16:   else
17:     I_ls[i] ← I_o[i]
18:   end if
19:   i ← i + 1
20: end while
21: return I_ls
```

4.1.9 Pseudocode of the memetic algorithm

Algorithm 5 shows the pseudocode for the memetic algorithm proposed in this paper to solve the SAP. Variables *r*, *g* and *t* control, respectively, the number of resets, the number of GA generations and the number of generations without improvement. The variables *numMaxResets*, *numMaxGeneration* and *numMaxWithoutImprovement* are input parameters. For each reset (lines 3-44), a new population is generated. A constructive heuristic generates the initial population (line 6), after which the fitness of each individual is calculated (line 7). To facilitate the execution of the selection operator and the storage of the best individual (line 9), the population is sorted from the best individual to the worst (line 8).

For each MA generation (lines 10-42), *n* pairs of individuals are selected to participate of the crossover operation and *n* offspring individuals are generated (line 13). According to a certain probability (m_p), each offspring may suffer a mutation in its genetic code (line 16). Function Random(0, 1) is a function that provides a random number in the interval [0, 1], according to a uniform distribution. The LS procedure then follows (line 19), after which the fitness of the offspring is calculated. The parents and respective offspring are then evaluated to determine who will remain in the population of the next generation (lines 22-32). The best individual of each generation is also stored (line 36). If the fitness of the best individual fails to improve over a certain consecutive number of generations then the LS is applied to it (line 39). The best individual found is returned with the conclusion of the GA (line 45). The overall time complexity of the algorithm is $\mathcal{O}(\text{numMaxResets}(P_{size}c_s + P_{size}\log P_{size} + P_{size} + \text{numMaxGeneration}(n(c_s + c_s^2) + n(c_s)) + P_{size} + c_s + c_s^2)$.

Algorithm 5 - Memetic Algorithm

$r \leftarrow 0$
$bestInd \leftarrow \emptyset$
while $r \leq numMaxResets$ **do**
  $g \leftarrow 0$
  $t \leftarrow 0$
  GenerateInitialPopulation($P(g)$)
  CalculateFitness($P(g)$)
  Sort($P(g)$)
  $bestInd \leftarrow$ FindBestInd($P(g)$)
  while $g \leq numMaxGeneration$ **do**
    Select n pair (Ind1, Ind2)
    for each pair (Ind1, Ind2) selected **do**
      $offspring \leftarrow crossover(Ind1, Ind2)$
      $R \leftarrow Random(0, 1)$
      if $R > m_p$ **then**
        $offspring \leftarrow mutation(offspring)$
      end if
      CalculateFitness($offspring$)
      LocalSearch($offspring$)
      CalculateFitness($offspring$)
    end for
    for each triple $(offspring, Ind_1, Ind_2) \in P(g)$ **do**
      if Fitness(Ind_1) > Fitness(Ind_2) **then**
        if Fitness($offspring$) > Fitness(Ind_2) **then**
          $Ind_2 \leftarrow offspring$
        end if
      else if Fitness(Ind_2) > Fitness(Ind_1) **then**
        if Fitness($offspring$) > Fitness(Ind_1) **then**
          $Ind_1 \leftarrow offspring$
        end if
      end if
    end for
    $bestInd_g \leftarrow$ FindBestInd($P(g)$)
    $t \leftarrow t + 1$
    if Fitness($bestInd_g$) > Fitness($bestInd$) **then**
      $bestInd \leftarrow bestInd_g$
      $t \leftarrow 0$
    else if $t > numMaxWithoutImprovement$ **then**
      $LocalSearch(bestInd)$
    end if
    $g \leftarrow g + 1$
  end while
  $r \leftarrow r + 1$
end while
return $bestInd$

5 Hyperparameter optimization

Both the GA and the MA employ a set of hyperparameters whose values need to be determined. As a result, this section presents an hyperparameter optimization process based on a statistical analysis in order to grant the results greater reliability (Sect. 6). This is then followed by a comparison study of both algorithms for synthetic instances of different sizes. Later, a comparison against previously reported results is carried out. In all simulations of this paper, the values of α_j (for $j = 1, \ldots, o$) are the same as those used in [29, 30].

Given that both algorithms proposed are non-deterministic, each hyperparameter combination was executed 30 times. During the hyperparameter optimization procedure an $L = 10$ instance was employed, resulting in a vertex grid of dimension 10×10. The individual stages of the optimization procedure can be briefly stated as follows:

1. Execute each hyperparameter combination 30 times;
2. Calculate the median (m_d) for each combination;
3. Select the 5 best value combinations;
4. Calculate the interquartile range (IR) of the 5 best combinations;
5. Choose the combination with the smallest IR value.

5.1 Genetic algorithm hyperparameter optimization

The above procedure was applied to validate the hyperparameter values for the GA. A brief description of each hyperparameter, alongside the set of respective values evaluated, can be found in Table 2.

5.1.1 Phase 1 - baseline behavior performance

In order to perform the tests, the combination of hyperparameters resulted in a search space of size $7 \times 7 \times 6 \times 5 = 1470$. Given that each combination was executed 30 times, a total of 44100 GA executions were performed in this first experiment. Subsequently, the 5 combinations that yielded the best results in terms of OF were selected according to median value and interquartile range. Table 3 lists the combinations (C) with the best hyperparameters classification, alongside the respective average ($\bar{x}$), median value (m_d), interquartile range (IR), standard deviation σ and execution time in seconds.

The solutions presented are ordered by IR. It is possible to see that the combinations listed present the same m_d value and IR values that are close to each other. The boxplot of the 5 best combinations is presented in Fig. 7, where the y-axis represents the OF values for the solutions obtained. The data demonstrates the occurrence of few outliers (namely in C_1, C_4 and C_5). In addition, there is also similarity in the dispersion value of the solutions obtained for each combination. Combinations C_1 and C_5 present the best dispersions, although, combination C_1 presents the best IR result according to Table 3.

Figure 8 presents a line plot with the median values of the OF according to the number of generations and for each combination of set C. The main goal is to verify whether a tendency for convergence is occurring or has already materialized for each combination. The plot depicts an accelerated initial decrease followed by a deceleration after 4000 generations that yields diminishing returns in terms of improving the OF. Combination C_1 presents the best IR results the first test stage.

5.1.2 Phase 2 - extending the number of maximum generations

Due to the fact that the 5 best combinations presented in Table 3 produced results very close to each other, we opted to re-execute the test. We chose to maintain the same set of hyperparameter values with the exception of the maximum number of generations gen_m, which was extended to 20000. The same test and analysis procedures were repeated, resulting in $5 \times 30 = 150$ GA executions. The results obtained are presented in Table 4 alongside the respective Δ hyperparameter variations from Table 3. A quick inspection reveals that the m_d values remained the same and the IR worsened.

The boxplot results for Table 4 are presented in Fig. 9. Combinations C_{11} and C_{41} still exhibit some outliers and overall solution dispersal remains close, mimicking the behaviour previously shown. However, if we examine the corresponding line plot with the median values of the OF, respectively presented in Fig. 10, it is possible to verify that after 7500 generations there exists a convergence tendency from all combinations. Configuration C_{11} still exhibits the best performance, albeit with a smaller difference when compared against the others.

5.1.3 Phase 3 - addition of a reset operator

Figure 10 also shows that from generation 7500 up to, approximately, generation 18500 each combination remained stagnant performance-wise. A possible technique

Table 2 GA hyperparameters alongside respective values evaluated

Hyperparameter	Description	Value set
m_r	Mutation rate	{0.05, 0.06, 0.07, 0.08, 0.09, 0.10, 0.20}
c_r	Crossover rate	{0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50}
p_s	Population size	{250, 500, 750, 1000, 1500, 2000}
gen_m	Maximum number of generations	{1000, 3000, 5000, 10000, 15000}

Table 3 Five best hyperparameter combinations concerning OF and ordered by IR (Phase 1)

C	m_r	c_r	p_s	gen_m	$\bar{x}$	m_d	**IR**	σ	Time(s)
$\mathbf{C_1}$	0.05	0.25	1500	10000	−23.89	−24.69	**2.69**	2.19	**343**
C_2	0.06	0.25	1500	10000	−23.56	−24.69	3.13	2.37	365
C_3	0.08	0.30	1500	15000	−23.44	−24.69	3.23	2.24	545
C_4	0.09	0.45	1000	15000	−23.06	−24.69	3.85	3.18	515
C_5	0.07	0.25	2000	10000	−24.05	−24.69	3.89	2.55	487

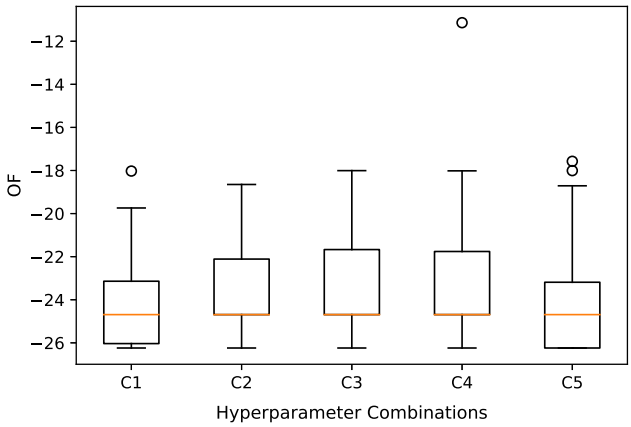


Fig. 7 Box plot for the 5 best hyperparameter combinations presented in Table 3 (Phase 1)

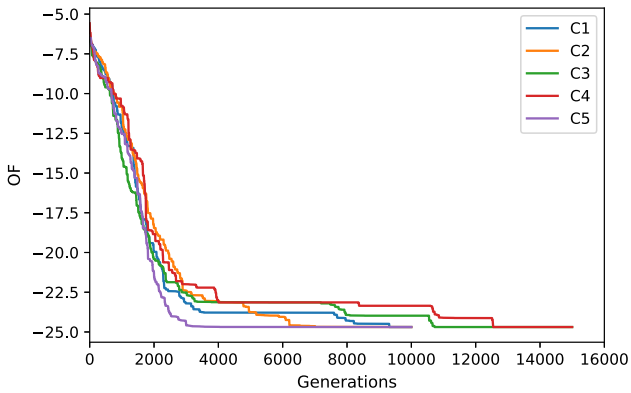


Fig. 8 Median value plot of the OF according to the number of generations (Phase 1)

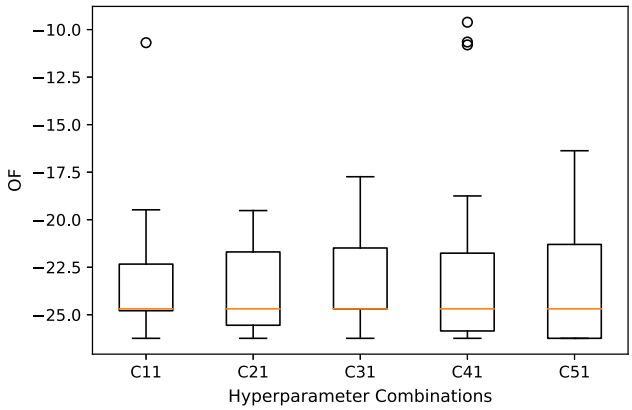


Fig. 9 Box plot for the 5 best hyperparameter combinations presented in Table 4 (Phase 2)

to deal with such a behaviour is to apply a population reset after a predetermined number of generations without improvement. Accordingly, we implemented a reset operator and examine its impact for the reset values $\{300, 500, 1000, 2000, 5000, 10000\}$. The hyperparameter combinations employed remained the same as in Phase 2 (Table 4). In this stage a total of $5 \times 6 \times 30 = 900$ GA executions were performed. The best results where found for reset value 5000 and are listed in Table 5 alongside the respective Δ hyperparameter variations from Table 4. Combination C_{22} presented the best m_d and IR values and the second best time. Figure 11 presents the boxplot for each of these combinations. Combinations C_{22}

Table 4 Hyperparameter combinations for $C_1, \ldots, C_5$ with gen_m extended to 20000 (Phase 2)

C	m_r	c_r	p_s	gen_m	$\bar{x}$	$\Delta\bar{x}$	m_d	Δm_d	**IR**	**Δ IR**	σ	$\Delta\sigma$	Time(s)
$\mathbf{C_{11}}$	0.05	0.25	1500	20000	−23.31	0.58	−24.69	0	**2.94**	0.25	3.10	0.91	**701**
C_{21}	0.06	0.25	1500	20000	−23.71	−0.15	−24.69	0	3.91	0.78	2.28	−0.09	726
C_{31}	0.08	0.30	1500	20000	−23.17	0.27	−24.69	0	3.39	0.16	2.65	0.41	841
C_{41}	0.09	0.45	1000	20000	−22.64	0.42	−24.69	0	3.39	−0.46	4.60	1.42	714
C_{51}	0.07	0.25	2000	20000	−23.42	0.63	−24.69	0	5.68	1.79	3.02	0.47	956

and C_{12} present the smallest dispersals, with the former of these having a more balanced m_d value.

Figure 12 presents the line plot with the median values of the OF for the third phase in terms of number of generations. The plot shows that, for all combinations, there does not occur an interval greater than 5000 generations without some improvement being observed. This behaviour enables the combinations evaluated to achieve better results during the same number of generations. Therefore, the addition of the reset operator allowed for an improvement in the evolution of the solutions. Overall, combination C_{22} presents the best results, according to the m_d, IR and time values.

5.2 Memetic algorithm results for hyperparameter optimization of GA

This stage employed the five best combinations that were obtained during the GA experiments. The values of each hyperparameter can be found in the left side of Table 6 whilst the right side present the results achieved with the MA execution and a reset value of 5000 alongside the respective Δ hyperparameter variations from Table 5.

Table 6 shows that for all combinations the average and median values were the same, respectively, 26.24. This result can be better explained by looking at the plot of Fig. 13 which shows the median OF value evolution for the selected combinations. Notice that, after a steep decline, all combinations start to converge to the same OF value. Accordingly, we believe this is the optimal value for an instance of this dimension. Consequently, the standard deviation and interquartile range presented value zero. This result is due to the local search employed by the MA that helps in exploring the search space. Given the similarity

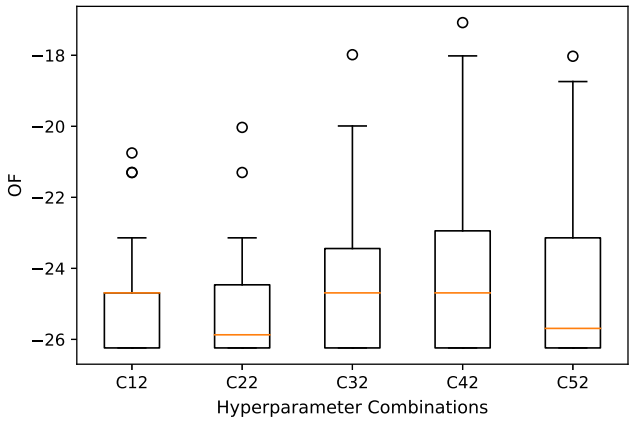


Fig. 11 Box plot for the 5 best hyperparameter combinations presented in Table 5 (Phase 3)

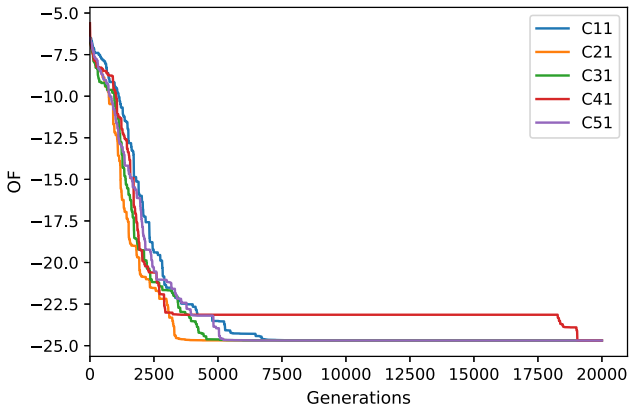


Fig. 10 Median value plot of the OF according to the number of generations and with $gen_m = 20000$ (Phase 2)

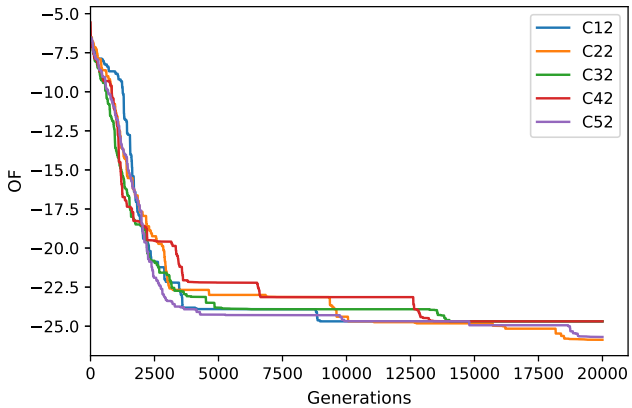


Fig. 12 Median value plot of the OF according to the number of generations and reset value of 5000 (Phase 3)

Table 5 Results for the five best performing hyperparameter combinations of phase 1 with population reset value 5000 (Phase 3).

C	m_r	c_r	p_s	gen_m	$\bar{x}$	$\Delta\bar{x}$	m_d	Δm_d	**IR**	**Δ IR**	σ	$\Delta\sigma$	Time(s)
C_{12}	0.05	0.25	1500	20000	−24.77	−0.88	−24.69	0	2.60	−0.09	1.54	−0.65	650
$\mathbf{C_{22}}$	0.06	0.25	1500	20000	−24.97	−1.41	−25.87	−1.18	**1.85**	−1.28	1.62	−0.75	696
C_{32}	0.08	0.30	1500	20000	−24.38	−1.32	−24.69	0	4.65	1.42	2.22	−0.96	818
C_{42}	0.09	0.45	1000	20000	−23.86	−0.8	−24.69	0	4.80	0.95	2.56	0.01	711
C_{52}	0.07	0.25	2000	20000	−24.43	−0.38	−25.69	0	3.10	−0.79	2.35	−0.2	1.252

Table 6 Five best hyperparameter combinations for the MA and with a reset of 5000.

C	m_r	c_r	p_s	gen_m	$\bar{x}$	$\Delta\bar{x}$	m_d	Δm_d	**IR**	Δ **IR**	σ	$\Delta\sigma$	Time(s)
C_{12}	0.05	0.25	1500	20000	−26.24	−1.47	−26.24	−1.55	0	−2.60	0	−1.54	532
C_{22}	0.06	0.25	1500	20000	−26.24	−1.27	−26.24	−0.37	0	−1.85	0	−1.62	522
C_{32}	0.08	0.30	1500	20000	−26.24	−1.86	−26.24	−1.55	0	−4.65	0	−2, 22	592
$\mathbf{C_{42}}$	0.09	0.45	1000	20000	−26.24	−2.38	−26.24	−1.55	0	−4.80	0	−2.56	**338**
C_{52}	0.07	0.25	2000	20000	−26.24	−1.81	−26.24	−0.55	0	−3.10	0	−2.35	981

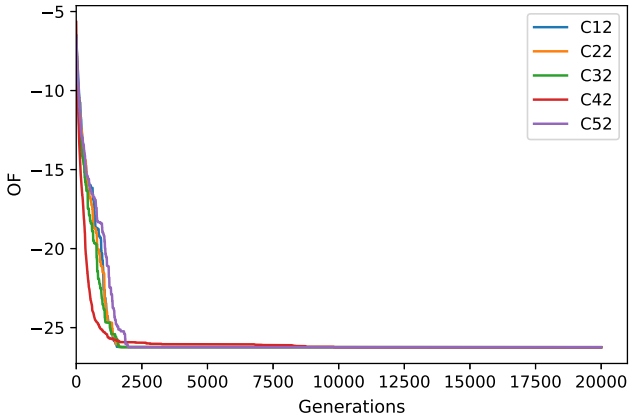


Fig. 13 Median value plot of the OF according to the number of generations for the MA

that all the results present, the combination selected was C_{42} since it required the least computational time.

6 Simulation and results

This section evaluates and compares the results obtained by the GA and the MA for three different sizes of instances. The algorithms were implemented in C++ and the experiments were performed in a computer with an Intel Core i5 7300HQ processor with 8 GB of RAM. The same procedure realized for hyperparameter optimization was performed here in order to evaluate the performance of the algorithms and avoid bias results. Thus, each algorithm was executed 30 times for every instance. The vertex grids examined consisted of 100 ($L = 10$), 225 ($L = 15$) and 400 ($L = 20$) nodes, which is a far more complex scenario than the previously addressed in the literature [29, 30]. The hyperparameter values employed in the experiments were the ones determined to be the best in Sects. 5.1.3 and 5.2. The average ($\bar{x}$), median (m_d), interquartile range (IR), standard deviation (σ) and average mean time are presented in Table 7, with the best values being presented in bold font. Figures 14, 15 and 16 present the average and median OF evolution of the GA and the MA for the selected instances.

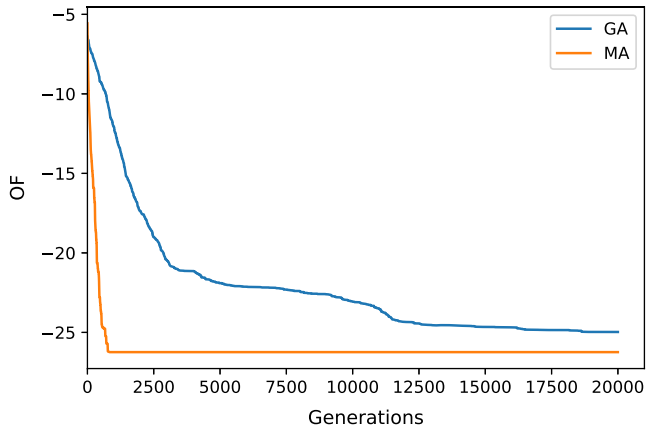
The numerical and visual data presented shows that the MA presented the best results for the $L = 10$ instance. Besides obtaining the best results the algorithm was also able to converge faster on the solution. Overall, the MA also exhibited the best behavior for the $L = 15$ instance, with the exception of the time dimension. Albeit, the MA presented a steeper decrease in OF than the GA; for $gen_m = 20000$ both algorithms failed to demonstrate a clear convergence tendency, which might indicate some room for further improvement. For $L = 20$ the GA showcased the best average, median and time values, whilst the MA exhibited the best IR and standard deviation. It is important to emphasize that the hyperparameter performance study performed in Sect. 5 employed an $L = 10$ size instance. This set of hyperparameter values may not be the most suitable for larger instances.

Furthermore, the population size that exhibited the best behavior for the MA contained 500 less individuals than the leading GA combination. The significant increase in the quantity of nodes, alongside the population growth, can be a justification for the MA not having outperformed the GA for the largest test instance. The considerable increase in execution time for the biggest instances can be attributed to the fact that the MA carries out a local search.

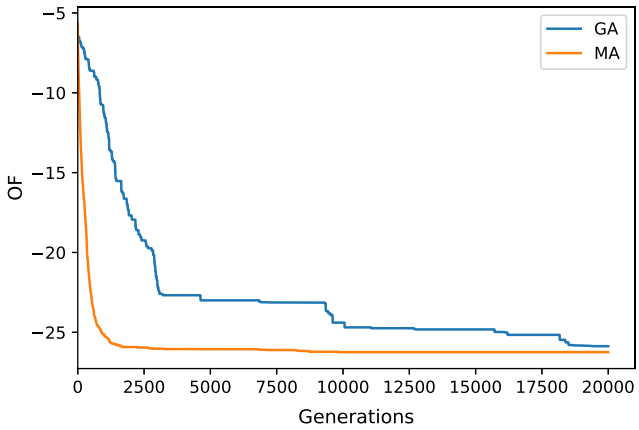
From the WSN perspective there are some aspects that should be highlighted. The average and median results demonstrate that the MA outperformed the GA for the $L = 10$ and $L = 15$ instances. This implies that the proposed heuristic was able to generate an optimal topology with respect to cover area and energy consumption, by allocating the best amounts of the different kinds of sensor nodes. However, the results presented in Table 7 demonstrate that the MA requires a longer time to obtain the optimal solution for the $L = 15$ and $L = 20$ instances. This behavior may have an impact on intended applications. Namely, topology control is a strategy applied beforehand to plan optimal sensor allocation. Accordingly, it may not be interesting (or even feasible) to have to wait a long time to

Table 7 GA versus MA for different instances size

Algorithm	m_r	c_r	p_s	gen_m	$\bar{x}$	m_d	**IR**	σ	Time(s)
$L = 10$ - 100 nodes									
GA	0.06	0.25	1500	20000	−24.97	−25.87	1.85	1.62	696
MA	0.09	0.45	1000	20000	**−26.24**	**−26.24**	**0**	**0**	**338**
$L = 15$ - 225 nodes									
GA	0.06	0.25	1500	20000	−19.08	−19.33	3.15	1.98	**1192**
MA	0.09	0.45	1000	20000	**−23.23**	**−23.38**	**2.10**	**1.88**	10652
$L = 20$ - 400 nodes									
GA	0.06	0.25	1500	20000	**−13.52**	**−13.52**	1.22	1.16	**3318**
MA	0.09	0.45	1000	20000	−12.11	−11.97	**0.85**	**0.72**	18294

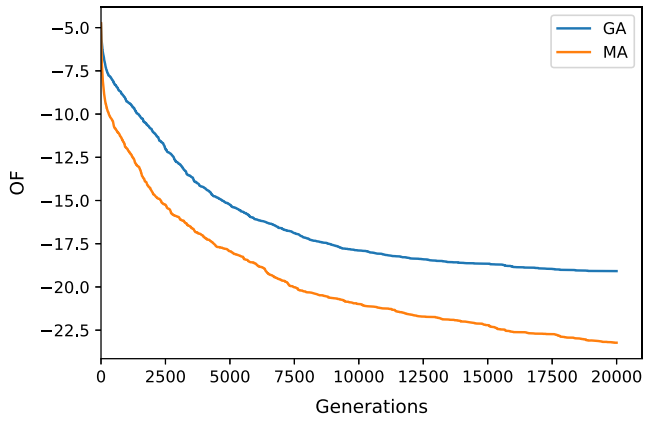


(a) Average OF values by generation - $L = 10$.

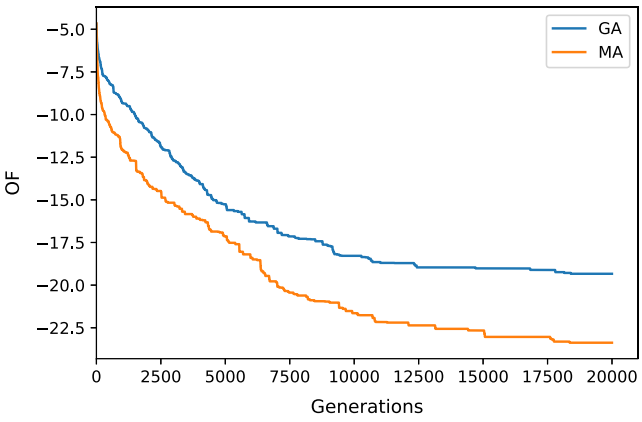


(b) Median OF values by generation - $L = 10$.

Fig. 14 GA vs MA comparison - $L = 10$



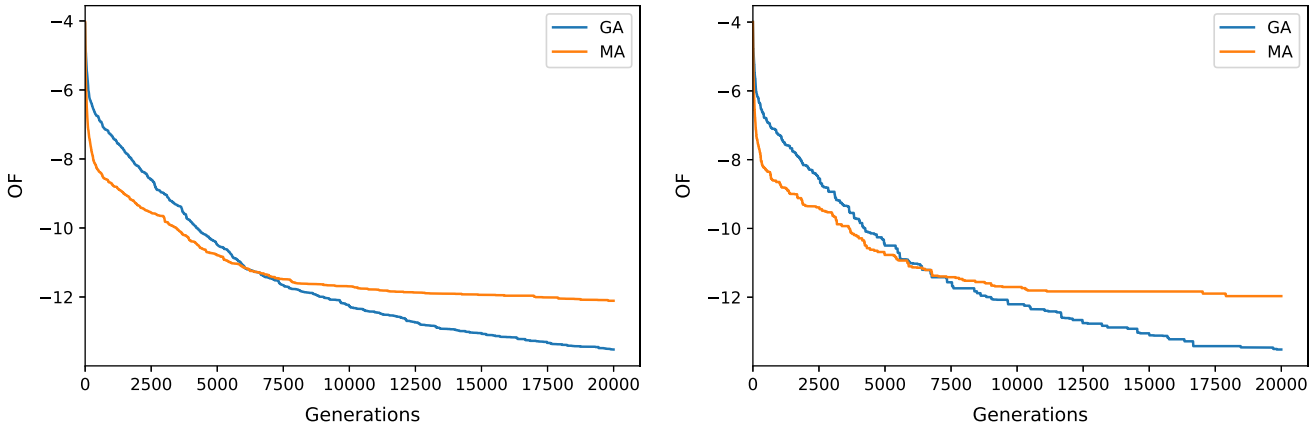
(a) Average OF values by generation - $L = 15$.



(b) Median OF values by generation - $L = 15$.

Fig. 15 GA vs MA comparison - $L = 15$

get the network up and running. Therefore, there may be a trade-off between the optimal MA solution and the time available to deploy a WSN. Nevertheless, the MA performed better than the GA in all metrics, including simulation time, for an $L = 10$ instance. Thus, this may indicate that the MA can be directly used when small instances are considered. On the other hand, for larger instances (*e.g.*: $L = 15$) the time required for the network deployment must be taken into account when choosing which heuristic to be apply.



(a) Average OF values by generation - $L = 20$.

(b) Median OF values by generation - $L = 20$.

Fig. 16 GA vs MA comparison - $L = 20$

Table 8 Results comparison between [29] and our GA and MA proposals for $L = 10$. Detailed result information for $L = 15$ and $L = 20$ is also presented

Results	**Fitness**	x	y	z	w	**Out-of-range**	Overlapping	f_1	f_2	f_3	f_4	f_5
$L = 10 - 100$ nodes												
[29]	−23.89	4	75	16	5	0	0	0.90	0	0	22.75	2.46
GA	−26.24	4	64	32	0	0	0	1.00	0	0	24.00	1.76
MA	−26.24	4	64	32	0	0	0	1.00	0	0	24.00	1.76
$L = 15 - 225$ nodes												
GA	−24.22	9	151	51	14	0	0	0.88	0	0	22.45	1.73
MA	−25.55	9	139	72	5	0	0	0.96	0	0	23.45	1.72
$L = 20 - 400$ nodes												
GA	−15.75	22	244	92	28	14	4	0.79	0.18	0.04	14.64	1.67
MA	−13.89	23	189	135	30	23	4	0.74	0.18	0.06	13.09	1.52

Table 8 presents a comparison between our GA and MA proposals alongside results from [29], with the addition of results from larger instances. For the smallest instance ($L = 10$), our methods achieved a larger fitness value, with the same number of x-sensors, when compared to [29]. Our solution also obtained larger f_1 and f_4 functions values, and reduced ones for f_5, which translates into a smaller energy footprint. Results for $L = 15$ demonstrate that the MA outperforms the GA, generating an optimal topology with less sensor nodes and decreased energy consumption. However, the larger instances require more computational time, specially for the MA, which uses a local search procedure. The GA performed better than the MA for the $L = 20$ instance with the latter generating a network with more x-sensors and increased out-of-range-sensors.

7 Conclusions

This paper investigates the utilization of an evolutionary heuristic to solve the SAP for topology control in WSNs. Network topology can be optimized with respect to cover area and energy expenditure, providing the sensor nodes with increased monitoring autonomy. The WSN considered was composed by nodes with distinct operational modes that presented different ranges based on transmission power. Such an optimization problem can be succinctly described as finding optimal locations and number of each kind of sensor, in order to balance energy expenditure whilst covering an $L \times L$ area.

The SAP was formulated as an integer linear programming problem and approaches based on a GA as well as a MA were implemented. The instances considered covered areas of dimension $L = 10$ (10×10 sensor nodes), $L = 15$ (15×15 sensor nodes) and $L = 20$ (20×20 sensor nodes), extending the work presented in [29]. The results obtained demonstrate that the MA outperforms the GA with respect to the covered area for both the $L = 10$ and $L = 15$ instances. The algorithm also presents the smallest execution time for the instance ($L = 10$). For $L = 15$, the MA found a better solution, but required a longer time. For the larger instance ($L = 20$), the GA outperformed the MA. Depending on the WSN application to be considered, this suggests a trade-off between optimal sensor placement

(leading to a better network energy balance when compared to a sub-optimal solution) and computational time, when one compares both evolutionary algorithms.

As future work, we intend to improve the proposed methodology designing one structured algorithm that is able to maintain the exploratory abilities of the proposed one, along with a smaller execution time. Further, an evaluation of the relationship between the evolutionary process and the solutions quality will be carried out. The authors intend to conduct case studies using real-world networks with areas that are not square. Thus, we intend to use the information of the coordinates of each node and consequently to calculate the distance between them and use this information, along with the sensor range, to solve the SAP.

Acknowledgements This work was partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) Finance Code 001 and by Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) - Reference: 210.286/2019.

References

1. Kim, H., & Han, S. (2015). An efficient sensor deployment scheme for large-scale wireless sensor networks. *IEEE Communications Letters, 19*(1), 98–101.
2. Radhika, S., & Rangarajan, P. (2019). On improving the lifespan of wireless sensor networks with fuzzy based clustering and machine learning based data reduction. *Applied Soft Computing, 83*, 105610.
3. Yan, Z., Mukherjee, A., Yang, L., Routray, S., & Palai, G. (2019). Energy-efficient node positioning in optical wireless sensor networks. *Optik, 178*, 461–466.
4. Kumar, D. P., Amgoth, T., & Annavarapu, C. S. R. (2019). Machine learning algorithms for wireless sensor networks: A survey. *Information Fusion, 49*, 1–25.
5. Stankovic, J. A. (2008). Wireless sensor networks. *Computer, 41*(10), 92–95.
6. Abdollahzadeh, S., & Navimipour, N. J. (2016). Deployment strategies in the wireless sensor network: A comprehensive review. *Computer Communications, 91–92*, 1–16.
7. Da Rocha Henriques, F., Lovisolo, L., & Barros da Silva, E. A. (2019). Rate-distortion performance and incremental transmission scheme of compressive sensed measurements in wireless sensor networks. *Sensors*. https://doi.org/10.3390/s19020266
8. do Prado, R. A., Guedes, R. M., da R. Henriques, F., da Costa, F. M., Tarrataca, L. D. T. J., & Haddad, D. B. (2020). On the analysis of the incremental $$ \ell $_0$$-LMS algorithm for distributed systems. *Circuits Systems and Signal Processing, 40*(2), 845–871. https://doi.org/10.1007/s00034-020-01500-z
9. Carmo, R. M., Tarrataca, L., Colares, J., Henriques, F. R., Haddad, D. B., & Guedes, R. M. (2020). Distributed adaptive filtering on wireless sensor networks with shared medium competition. *Learning and Nonlinear Models, 18*(1), 15–34. https://doi.org/10.21528/lnlm-vol18-no1-art2
10. d. Prado, R.A., d. R. Henriques, F., & Haddad, D.B. (2018) Sparsity-aware distributed adaptive filtering algorithms for nonlinear system identification. In: 2018 International joint conference on neural networks (IJCNN), pp. 1–8 . https://doi.org/10.1109/IJCNN.2018.8489173
11. Boukerche, A., & Sun, P. (2018). Connectivity and coverage based protocols for wireless sensor networks. *Ad Hoc Networks, 80*, 54–69.
12. Jiang, Ruixiang, & Chen, Biao. (2005). Fusion of censored decisions in wireless sensor networks. *IEEE Transactions on Wireless Communications, 4*(6), 2668–2673.
13. Yang, L., Zhu, H., Wang, H., Kang, K., & Qian, H. (2019). Data censoring with network lifetime constraint in wireless sensor networks. *Digital Signal Processing, 92*, 73–81.
14. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks, 38*, 393–422.
15. Stankovic, J. A., Abdelzaher, T., Lu, C., Sha, L., & Hou, J. C. (2003). Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE, 91*(7), 1002–1022.
16. Hussain, S., & Islam, O. (2007) An energy efficient spanning tree based multi-hop routing in wireless sensor networks. In: Wireless communications and networking conference, 2007, WCNC 2007, IEEE, pp. 4383–4388.
17. Henriques, F. R., Lovisolo, L., & Rubinstein, M. G. (2016). DECA: distributed energy conservation algorithm for process reconstruction with bounded relative error in wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking, 2016*(163), 1–18.
18. Chen, H., Li, X., & Zhao, F. (2016). A reinforcement learning-based sleep scheduling algorithm for desired area coverage in solar-powered wireless sensor networks. *IEEE Sensors Journal, 16*(8), 2763–2774.
19. Yi, C. (2009). A unified analytic framework based on minimum scan statistics for wireless ad hoc and sensor networks. *IEEE Transactions on Parallel and Distributed Systems, 20*(9), 1233–1245.
20. Singh, S. P., & Sharma, S. C. (2015). A survey on cluster based routing protocols in wireless sensor networks. *Procedia Computer Science, 45*, 687–695. https://doi.org/10.1016/j.procs.2015.03.133
21. Üster, H., & Lin, H. (2011). Integrated topology control and routing in wireless sensor networks for prolonged network lifetime. *Ad Hoc Networks, 9*(5), 835–851. https://doi.org/10.1016/j.adhoc.2010.09.010
22. Shutimarrungson, N., & Wuttidittachotti, P. (2019). Realistic propagation effects on wireless sensor networks for landslide management. *EURASIP Journal on Wireless Communications and Networking, 2019*(1), 1–14.
23. Ali, N. S., Alkaream Alyasseri, Z. A., & Abdulmohson, A. (2018). Real-time heart pulse monitoring technique using wireless sensor network and mobile application. *International Journal of Electrical Computer Engineering, 8*(6), 2088–8708.
24. Avci, O., Abdeljaber, O., Kiranyaz, S., Hussein, M., & Inman, D. J. (2018). Wireless and real-time structural damage detection: A novel decentralized method for wireless sensor networks. *Journal of Sound and Vibration, 424*, 158–172.
25. Younus, M. U., Islam, Su., & Kim, S. W. (2019). Proposition and real-time implementation of an energy-aware routing protocol for a software defined wireless sensor network. *Sensors, 19*(12), 2739.
26. Gendreau, M., & Potvin, J.-Y. (2010). *Handbook of Metaheuristics* (2nd ed.). New York: Springer.
27. França, P. M., Mendes, A., & Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research, 132*, 224–242.
28. Corne, D., Dorigo, M., & Glover, F. (1999). *New Ideas in Optimization*. United Kingdom: McGraw-Hill.

29. Bhondekar, A. P., Vig, R., Singla, M. L., Ghanshyam, C., & Kapur, P. (2009). Genetic algorithm based node placement methodology for wireless sensor networks. *Proceedings of the International Multiconference of Engineers and Computer Scientists, 1*, 18–20.
30. Srivastava, J. R., & Sudarshan, T. S. B. (2015). Energy-efficient cache node placement using genetic algorithm in wireless sensor networks. *Soft Computing, 19*(11), 3145–3158. https://doi.org/10.1007/s00500-014-1473-8
31. de Brito, J.A.G., de Junior, J.R., Henriques, F.d.R., & de Assis, L.S. (2019) Topology control optimization of wireless sensor networks for iot applications. In: Proceedings of the 25th brazillian symposium on multimedia and the web. WebMedia '19, Association for Computing Machinery, New York, NY, USA pp. 477–480. https://doi.org/10.1145/3323503.3361718
32. Sun, Z., Akyildiz, I. F., & Hancke, G. P. (2011). Dynamic connectivity in wireless underground sensor networks. *IEEE Transactions on Wireless Communications, 10*(12), 4334–4344.
33. Nguyen, P. L., Hanh, N. T., Khuong, N. T., Binh, H. T. T., & Ji, Y. (2019). Node placement for connected target coverage in wireless sensor networks with dynamic sinks. *Pervasive and Mobile Computing, 59*, 101070.
34. Ma, C., Liang, W., Zheng, M., & Sharif, H. (2016). A connectivity-aware approximation algorithm for relay node placement in wireless sensor networks. *IEEE Sensors Journal, 16*(2), 515–528.
35. Fang, W., Song, X., Wu, X., Sun, J., & Hu, M. (2018). Novel efficient deployment schemes for sensor coverage in mobile wireless sensor networks. *Information Fusion, 41*, 25–36.
36. Huang, G., Chen, D., & Liu, X. (2015). A node deployment strategy for blindness avoiding in wireless sensor networks. *IEEE Communications Letters, 19*(6), 1005–1008.
37. Khalily-Dermany, M., Nadjafi-Arani, M. J., & Doostali, S. (2019). Combining topology control and network coding to optimize lifetime in wireless-sensor networks. *Computer Networks, 162*, 106859.
38. Papadimitriou, C. H. (1981). On the complexity of integer programming. *J. ACM, 28*(4), 765–768. https://doi.org/10.1145/322276.322287
39. Senouci, M. R., & Lehtihet, H. E. (2018). Sampling-based selection-decimation deployment approach for large-scale wireless sensor networks. *Ad Hoc Networks, 75–76*, 135–146.
40. Fu, X., Yao, H., & Yang, Y. (2019). Exploring the invulnerability of wireless sensor networks against cascading failures. *Information Sciences, 491*, 289–305.
41. Hasan, M. M., & Mouftah, H. T. (2017). Optimization of watchdog selection in wireless sensor networks. *IEEE Wireless Communications Letters, 6*(1), 94–97.
42. Seo, J. (2015). On minimizing energy consumption of duty-cycled wireless sensors. *IEEE Communications Letters, 19*(10), 1698–1701.
43. Bahi, J., Elghazel, W., Guyeux, C., Hakem, M., Medjaher, K., & Zerhouni, N. (2019). Reliable diagnostics using wireless sensor networks. *Computers in Industry, 104*, 103–115.
44. Li, F., Luo, J., Xin, S., & He, Y. (2016). Autonomous deployment of wireless sensor networks for optimal coverage with directional sensing model. *Computer Networks, 108*, 120–132.
45. Yun, Y., Xia, Y., Behdani, B., & Smith, J. C. (2013). Distributed algorithm for lifetime maximization in a delay-tolerant wireless sensor network with a mobile sink. *IEEE Transactions on Mobile Computing, 12*(10), 1920–1930.
46. Rakavi, A., Manikandan, M.S.K., & Hariharan, K. (2015) Grid based mobile sensor node deployment for improving area coverage in wireless sensor networks. In: 2015 3rd International conference on signal processing, communication and networking (ICSCN), pp. 1–5
47. Chou, C. T., Ignjatovic, A., & Hu, W. (2013). Efficient computation of robust average of compressive sensing data in wireless sensor networks in the presence of sensor faults. *IEEE Transactions on Parallel and Distributed Systems, 24*(8), 1525–1534.
48. Cheffena, M., & Mohamed, M. (2017). Empirical path loss models for wireless sensor network deployment in snowy environments. *IEEE Antennas and Wireless Propagation Letters, 16*, 2877–2880.
49. Tsiropoulou, E.E., Paruchuri, S.T., & Baras, J.S. (2017) Interest, energy and physical-aware coalition formation and resource allocation in smart iot applications. In: 2017 51st Annual conference on information sciences and systems (CISS), pp. 1–6 . https://doi.org/10.1109/CISS.2017.7926111
50. Jiang, C., Chen, Y., Gao, Y., & Liu, K. J. R. (2013). Joint spectrum sensing and access evolutionary game in cognitive radio networks. *IEEE Transactions on Wireless Communications, 12*(5), 2470–2483. https://doi.org/10.1109/TWC.2013.031813.121135
51. Primeau, N., Falcon, R., Abielmona, R., & Petriu, E. M. (2018). A review of computational intelligence techniques in wireless sensor and actuator networks. *IEEE Communications Surveys Tutorials, 20*(4), 2822–2854. https://doi.org/10.1109/COMST.2018.2850220
52. Ma, L., Cheng, S., & Shi, Y. (2021). Enhancing learning efficiency of brain storm optimization via orthogonal learning design. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51*(11), 6723–6742. https://doi.org/10.1109/TSMC.2020.2963943
53. Ma, L., Huang, M., Yang, S., Wang, R., & Wang, X. (2021). An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization. *IEEE Transactions on Cybernetics*. https://doi.org/10.1109/TCYB.2020.3041212
54. Ma, L., Wang, X., Wang, X., Wang, L., Shi, Y., & Huang, M. (2021). Tcda: Truthful combinatorial double auctions for mobile edge computing in industrial internet of things. *IEEE Transactions on Mobile Computing*. https://doi.org/10.1109/TMC.2021.3064314
55. Ting, C.-K., & Liao, C.-C. (2010). A memetic algorithm for extending wireless sensor network lifetime. *Information Sciences, 180*(24), 4818–4833. https://doi.org/10.1016/j.ins.2010.08.021
56. Fu, X., Pace, P., Aloi, G., Yang, L., & Fortino, G. (2020). Topology optimization against cascading failures on wireless sensor networks using a memetic algorithm. *Computer Networks, 177*, 107327. https://doi.org/10.1016/j.comnet.2020.107327
57. Manap, S., Dimyati, K., Hindia, M. N., Abu Talip, M. S., & Tafazolli, R. (2020). Survey of radio resource management in 5g heterogeneous networks. *IEEE Access, 8*, 131202–131223. https://doi.org/10.1109/ACCESS.2020.3002252
58. Bouchemal, N., Kallel, S., & Bouchemal, N. (2018) A survey: Wsn heterogeneous architecture platform for iot. In: International conference on machine learning for networking, Springer, pp. 321–332.
59. Al-Turjman, F. M., Hassanein, H. S., & Ibnkahla, M. (2013). Quantifying connectivity in wireless sensor networks with grid-based deployments. *Journal of Network and Computer Applications, 36*(1), 368–377.
60. Bondy, A., & Ramachandra, M. U. S. (2008). *Graph theory*. United Kingdom: Springer.
61. Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2001). *Introduction to algorithms* (2nd ed.). Cambridge: MIT press.
62. Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics, 16*(1), 122–128. https://doi.org/10.1109/TSMC.1986.289288

63. Bacao, F., Lobo, V., & Painho, M. (2005). Applying genetic algorithms to zone design. *Soft Computing, 9*, 28–35.
64. de Assis, L. S., González, J. F. V., Usberti, F. L., Lyra, C., Cavellucci, C., & Zuben, F. J. V. (2015). Switch allocation problems in power distribution systems. *IEEE Transactions on Power Systems, 30*(1), 246–253. https://doi.org/10.1109/TPWRS.2014.2322811
65. de Assis, L. S., de P. Junior, J. R., Tarrataca, L., & Haddad, D. B. (2019). Efficient volterra systems identification using hierarchical genetic algorithms. *Applied Soft Computing, 85*, 105745. https://doi.org/10.1016/j.asoc.2019.105745
66. Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing, 4*(2), 65–85. https://doi.org/10.1007/BF00175354
67. Gong, Y.-J., Chen, W.-N., Zhan, Z.-H., Zhang, J., Li, Y., Zhang, Q., & Li, J.-J. (2015). Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing, 34*, 286–300. https://doi.org/10.1016/j.asoc.2015.04.061
68. Moscato, P. , & Norman, M.G. (1992) A "memetic" approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In: In Proceedings of the international conference on parallel computing and transputer applications, pp. 177–186, IOS Press.
69. Eiben, A., & Smith, J. (2015). *Introduction to evolutionary computing (natural computing series)* (p. 98). Germany: Springer.

Jorge A. G. de Brito Jorge Augusto Gomes de Brito was born in Rio de Janeiro, RJ, Brazil in 1990. He received the B.sC. degree in Computer Science from Centro Universitário Augusto Motta (Unisuam), in 2018. His research interests are Optimization in Wireless Sensor Networks and Smart Cities.

Diego R. M. Totte Diego Rodrigues Moreira Totte holds a technical course in Computer (2003), a degree in Information Systems (2009) and is M.Sc. student in Computer Science at Federal Center for Technological Education (CEFET/RJ), Brazil. He worked on infrastructure, systems and data projects. He is currently managing a data team on accounting projects in the insurance business sector. His research interests are Optimization in Wireless Sensor Networks and Smart Cities.

Fábio O. Silva Fábio Oliveira Silva holds a degree in Computer Engineering (2009), a master's degree in Applied Computing (2016) and is a PhD student in Instrumentation and Applied Optics at the Federal Technological Center for Education of Rio de Janeiro (CEFET/RJ). He is currently a professor of Computing at the Federal Institute of Education, Science and Technology of Bahia (IFBA). His research interests are Wireless Sensor Networks, Internet of Things, Sensing and Optimization of Energy Resources.

Jurair R. de P. Junior Jurair Rosa B.Sc. degree in Computer Science from Faculdades Integradas de Caratinga (FIC) in 2007. M.Sc. degree in Systems and Computer Engineering from the Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (Coppe) of the Federal University of Rio de Janeiro (UFRJ) in 2010. He has extensive knowledge in the development of scientific systems (R &D) and commercial systems, having worked in companies in the Port and Oil and Gas sectors between 2008 and 2015. Currently, he is a professor of Computer Engineering at the Federal Education Center Technology (CEFET/RJ), Brazil. Among the areas of interest and performance are Operational Research and Packaging Problems, with an emphasis on optimization, and the development of solutions involving Smart Cities.

Felipe da Rocha Henriques Felipe da Rocha Henriques received his B.Sc. degree in Electrical Engineering (2006) and his M.Sc. degree in Electronic Engineering (2010) from the State University of Rio de Janeiro, Brazil, and received his D.Sc. degree in Electrical Engineering (2015) from the Federal University of Rio de Janeiro, Brazil. Currently, Felipe is the Director of CEFET/RJ (campus Petrópolis), where he works as a Professor at the department of telecommunications, at the master's program in computer science and at the doctoral program in Instrumentation and applied optics. His research interests include Wireless Sensor Networks, Internet of Things, sparse representation and Compressive Sensing.

Luís Tarrataca Luís Tarrataca is a Professor at the department of computer engineering of CEFET-RJ and a researcher at LNCC's Quantum Computing Group group in Rio de Janeiro. He received his B.Sc. (2007), M.Sc (2008) and Ph.D (2013) degrees from Instituto Técnico / Techincal University of Lisbon. His research interests include quantum computation, quantum information, machine learning, computer vision and artificial intelligence algorithms.

Diego Barreto Haddad Diego Barreto Haddad received his B.Sc. degree in Electrical Engineering in 2005, and the M.Sc. and D.Sc. degrees in Electrical Engineering from the Federal University of Rio de Janeiro, Brazil, in 2008 and 2013, respectively. He is with the Federal Centee for Technological Education (CEFET/RJ). His research interests include signal processing, machine learning, computer vision and adaptive filtering algorithms.

Laura S. de Assis Laura Assis holds a bachelor's degree in Computer Science (2007), Master (2009) and PhD (2014) in Electrical Engineering from the State University of Campinas. She is currently a professor of Computer Engineering at the Federal Education Center Technology (CEFET/RJ), Brazil. Your interests of research are Operations Research, Combinatorial and Multiobjective Optimization, Bioinspired Computing, Reliability in Power Networks and Complex Networks.