Applied Soft Computing 47 (2016) 395–414

Contents lists available at ScienceDirect

Applied Soft Computing

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

On the efficiency of the binary flower pollination algorithm: Application on the antenna positioning problem

Zakaria Abd El Moiz Dahi*, Chaker Mezioud, Amer Draa

MISC Laboratory, Constantine 2 University – Abdelhamid Mehri, Algeria

ARTICLE INFO

Article history:
Received 10 June 2015
Received in revised form 9 March 2016
Accepted 30 May 2016
Available online 16 June 2016

Keywords:
Flower pollination algorithm
Mapping techniques
Antenna positioning problem
Cellular networks

ABSTRACT

The Flower Pollination Algorithm (FPA) is a recently proposed continuous metaheuristic that was claimed to give promising results. However, its potential in binary problems has been vaguely investigated. The use of mapping techniques to adapt metaheuristics to handle binary optimisation problems is a widely-used approach, but these techniques are still fuzzy and misunderstood, since no work thoroughly studied them for a given problem or algorithm. This paper conducts a consistent and systematic study to assess the efficiency of the FPA and the common mapping techniques. This is done through proposing four Binary variants of the FPA (BFPA) that have been got by applying the principal mapping techniques existing in the literature. As benchmark problem; an NP-hard binary one in advanced cellular networks, the Antenna Positioning Problem (APP), is used. In order to assess the scalability, efficiency and robustness of the proposed BFPAs, the experiments have been carried out on realistic, synthetic and random data with different dimensions, and several statistical tests have been carried. Two of the top-ranked algorithms designed to solve the APP; the Population-Based Incremental Learning (PBIL) and the Differential Evolution algorithm (DE), are taken as a comparison basis. The results showed that the normalisation and angle modulation are the best mapping techniques. The experiments also showed that the BFPAs have some shortcomings but, they could outperform the PBIL in 4 out of 13 instances and the DE in 6 out of 13 instances and no statistical difference was found in the remaining instances. Besides, the BFPAs outperformed or gave competitive technical results compared to the PBIL and DE in all problem instances.

1. Introduction

Many real-world challenging problems in several industrial fields, transportation, packaging and networking are formulated as binary optimisation problems. We cite, for instance, the Traveling Salesman Problem (TSP) [33], the Bin Packing Problem (BPP) [49], the Knapsack Problem (KP) [86], the Permutation Flow-Shop Problem (PFSP), the Job-Shop Scheduling Problem (JSSP) [50], the Lot-Sizing Problem (LSP) [56] and the Feature Selection Problem (FSP) [39]. These examples make the need for creating efficient algorithms to solve binary problems as much challenging and tricky as creating algorithms to solve continuous ones.

The main motivation behind the design of metaheuristics is ultimately their use by non-experts for solving such problems. Regardless to their source of inspiration, metaheuristics differ on their architectures, operators and solution representation. On the basis of the latter, metaheuristics can be further classified into *continuous metaheuristics*, designed originally to operate only on real-coded solutions, and *binary metaheuristics*, designed to operate only on binary-coded ones. Considering this problem-dependency and knowing that an efficient resolving of a given problem often requires an optimal choice of the algorithm, limits the use of metaheuristics to specialists within abstract scientific research. Indeed, most of the scientific works propose powerful algorithms without further consideration for their industrial application while there is a growing demand for optimisation software usable by non-specialists within an industrial environment.

A promising way to cope with this issue, is to supply the industry with algorithms that can handle both binary and continuous problems. Indeed, despite the fact that most continuous metaheuristics were originally designed to deal with continuous optimisation problems, many research efforts are deployed in order to adapt them for solving binary ones. Kennedy and Eberhart [53] proposed a binary version of the Particle Swarm Optimisation algorithm (BPSO). Rashedi et al. [29] devised a binary version of the Gravitational Search Algorithm (BGSA). A binary Harmony Search

* Corresponding author.
E-mail addresses: zakaria.dahi@univ-constantine2.dz (Z.A.E.M. Dahi), chaker.mezioud@univ-constantine2.dz (C. Mezioud), draa.amer@gmail.com (A. Draa).

http://dx.doi.org/10.1016/j.asoc.2016.05.051
1568-4946/

algorithm (BHS) was also proposed in [21]. Recently, the authors in [24,34,76,81] proposed binary versions of the Bat Algorithm (BBA), the FireFlyAlgorithm (BFFA), the Cuckoo Search (BCS) and the Artificial Bee Colony algorithm (BABC), respectively.

For these reasons, one of today's hot topics of research consists of designing efficient mapping techniques that allow a given metaheuristic to act on both real and binary-coded solutions [93]. On the other hand, the drawbacks of using these techniques is that they may increase the complexity of the problem, affect the behaviour of the algorithms, their efficiency and their sensitivities. So, the whole difficulty lies in finding the adequate framework that will avoid adding more nonlinearities and mathematical difficulties either to the problem or the algorithm [93]. Some techniques have already been presented in the literature [53,81], but since no work was conducted to study their efficiency, sensitivity, controlling factors, these techniques still fuzzy and misunderstood. In fact, many questions still surround them such as: what are their drawbacks and advantages? What are their controlling factors? How do they behave? Is it worth using them? Are they really efficient? Ultimately, how to design more efficient mapping techniques?

The Flower Pollination Algorithm (FPA) is a recent swarm-inspired algorithm proposed in [98]. It was initially designed to address continuous optimisation problems and showed a promising level of efficiency, compared to other metaheuristics such as the PSO [44] and Genetic Algorithm (GA) [46], but unfortunately only one work studied its efficiency when tackling binary optimisation problems [25]. Despite that, this work was not enough encompassing and a bit simplistic. In fact, many research questions remains concerning this recent algorithm, such as: is this metaheuristic really efficient? What is its efficiency compared to today's top-ranked algorithms? Is it worth adapting continuous metaheuristics to solve binary problems? Ultimately, what kind of algorithm to use when tackling such problems?

Given the above considerations, this paper conducts a systematic study on both the FPA and the mapping techniques in order to answer some of the above-stated research questions. This is done by proposing four binary variants of the FPA on the basis of the principal mapping techniques existing in the literature. These techniques are: the Nearest-Integer (NI) method [76], the normalisation, the Angle Modulation (AM) method, and finally the Sigmoid Function (SF) method [53,81].

As a benchmark problem, an NP-hard optimisation problem in advanced cellular networks, the Antenna Positioning Problem (APP), is used. This problem occurs during the design phase of any kind of cellular phone networks: 2G, GPRS, EDGE, 3G, 3G$^+$, LTE and 4G [58]. Despite the fact that this phase regroups the most important issues of the radio mobile network life cycle [60], the antenna positioning problem remains the most influential and the most quality-determining, due to the fact that a good (bad) antenna positioning may lead to situations of inextricable overlapping, that might make frequency assignment easier (harder) in the 2G networks, induce (avoid) cell breathing in 3G networks or finally alter (facilitate) the mobility management in LTE and 4G networks and inevitably jeopardise (protect) the rest of the design phase [58,60].

For consistency and reliability, the experiments are carried out on realistic, synthetic and random instances with different sizes to assess the scalability, robustness and the efficiency of the proposed BFPAs. Two of the top-ranked state-of-the-art algorithms used to solve the APP, the Population-Based Incremental Learning (PBIL) and the Differential Evolution algorithm (DE), are used as a comparison basis. Several statistical analysis tests are conducted also to assess the quality of solutions.

The remainder of the paper is structured as follows. In Section 2, we introduce the basic concepts related to the flower pollination algorithm, the antenna positioning problem and discretisation approaches. In Section 3, we present the related work to each one of the latter. In Section 4, we present the newly-proposed variants of the flower pollination algorithm. Section 5 is dedicated to the experimental results and their interpretations and discussion. Finally, we conclude the paper in Section 6.

2. Basic concepts

Basic concepts related to the flower pollination algorithm, the antenna positioning problem and real-coded to binary-coded search space mapping process are given in this section.

2.1. The flower pollination algorithm

The flower pollination algorithm is a recent metaheuristic inspired by the biological process of flower pollination. More specifically; the algorithm simulates the notions of flower consistency and the pollinator behaviour found in nature. Before moving to presenting the flower algorithm, we start by giving a brief description of the biological phenomenon [98–100].

2.1.1. Biological pollination

Flower pollination is typically associated with the transfer of pollen which is basically achieved by insects and animals. Pollination can take two major forms: abiotic and biotic. About 90% of flowering plants belong to biotic pollination; that is, pollen is transferred by a pollinator such as insects and animals. The other 10% of pollination takes an abiotic form which does not require any pollinators [98]. Some pollinators exclusively visit one species of flower, this is called flower constancy. The latter permits the reproduction of the same flower species. This is also advantageous for the pollinators, since they will be sure of the availability of nectar supply with a limited memory and minimum cost of learning.

As far as the availability of pollinators is concerned; there are two types of pollination: self-pollination and cross-pollination. In the case of the latter, pollen is obtained from a flower of a different plant, while in self-pollination, the fertilisation of one flower is done by the pollen of the same flower. Self-pollination is very useful when no reliable pollinator is available, whereas cross-pollination, also called global pollination, would permit the pollen of a flower to travel for long distances, thanks to the Lévy flight [42] behaviour of pollinators [98].

2.1.2. The flower algorithm

In [98], Yang emulated the biological pollination process using the four following idealisation rules.

1. Biotic and cross-pollination are considered as global pollination processes with pollen-carrying pollinators performing Lévy flights.
2. Abiotic and self-pollination are considered as local pollination.
3. Flower constancy can be considered as the reproduction probability; it is proportional to the similarity of two flowers involved.
4. Local pollination and global pollination are controlled by a switch probability $P \in [0, 1]$. Due to the physical proximity and other factors, such as wind; local pollination can have a significant fraction p in the overall pollination activities.

For simplifying the task of implementing these rules, the author considers that each plant only has one flower and each flower only produces one pollen gamete. Thus, there is no need to distinguish a pollen gamete, a flower, a plant or a solution to a problem. This simplification means that for a problem with D dimensions, a solution vector $\vec{X} = \{x_1, x_2, \ldots, x_D\}$ is equivalent to a flower and/or a pollen gamete.

On the basis of this formulation, and through exploiting the notions of local and global pollination, Yang proposes to use Eq. (1) to implement Rule 1 and flower constancy (Rule 3), and to use Eq. (2) to implement local pollination and flower constancy; Rules 2 and 3, respectively.

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \vec{L} \cdot (\vec{X}_i^t - \vec{G}) \quad (1)$$

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \epsilon \cdot (\vec{X}_j^t - \vec{X}_k^t) \quad (2)$$

In Eq. (1), $\vec{X}_i^t$ is the ith pollen or solution vector $\vec{X}_i$ at iteration t, and $\vec{X}_i^{t+1}$ is a candidate solution for iteration $t+1$. $\vec{G}$ is the current best solution, where $\vec{G} = \{g_1, g_2, \ldots, g_D\}$. The variable $\vec{L}$ is the strength of pollination, basically a D-dimensional step size, where $\vec{L} = \{l_1, l_2, \ldots, l_D\}$.

As for animals, pollinators search for food in a random or quasi-random manner. In general, the foraging path of an animal is effectively a random walk because the next move is based on the current location/state and the transition probability to the next location. The direction it chooses depends implicitly on a probability which can be modelled mathematically. Since pollinators, insects and birds can move for long distances with various distance steps, many studies showed that this phenomenon has the same characteristics as a Lévy flight [97].

As a projection of its natural counterpart, the Lévy flight is a kind of random algorithm or a random movement process similar to a random walk; a particle makes a sequence of moves, where each move is in a random direction. Unlike a random walk, where the length of the moves is identical, Lévy flight moves have a random length generated from a heavy-tailed probability distribution. For instance, the Lévy distribution is an example of heavy-tailed probability distribution [32,43]. In general, Lévy distribution should be defined in terms of Fourier transform [97]. A Lévy flight is classified as a Markov process. In Markovian processes, the future state of an object depends only on the object's present state and a limited number of past states.

Lévy flights have been used as a fundamental key to improve algorithms in many fields. Several Lévy flight implementations have also been proposed in the literature such as: McCulloch's algorithm [47], the Rejection algorithm [95] and Mantegna's algorithm [80]. The latter is the one used in the flower pollination algorithm. It consists of drawing a step $l_h > 0$ from a Lévy distribution according to Eq. (3), where $l_h \in \vec{L}$ and $h = 1, \ldots, D$.

$$l_h \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \cdot \frac{1}{s^{1+\lambda}} \quad (s \gg s_0 > 0). \quad (3)$$

In Eq. (3), $\Gamma(\lambda)$ is the standard gamma function. This Lévy distribution is valid for large steps, $s > 0$. In theory, it is required that s_0 verifies the condition $|s_0| \gg 0$, but in practice s_0 can be as small as 0.1 [97,99]. s is drawn using Eq. (4).

$$s = \frac{U}{|V|^{1/\lambda}}; \quad U \sim N(0, \sigma^2), \quad V \sim N(0, 1). \quad (4)$$

Variables U and V are two random numbers obeying a Gaussian distribution with the specificities shown in the same equation, $U \sim N(0, \sigma^2)$ means that the samples are drawn from a Gaussian normal distribution with a zero mean and a variance of σ^2. The latter is calculated using Eq. (5) [99].

$$\sigma^2 = \left\{ \frac{\Gamma(1+\lambda)}{\lambda \Gamma[(1+\lambda)/2]} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}} \right\}^{1/\lambda}. \quad (5)$$

In Eq. (2) of local pollination, $\vec{X}_i^t$ is the ith pollen or solution vector $\vec{X}_i$ at iteration t, $\vec{X}_i^{t+1}$ is a candidate solution for iteration $t+1$, $\vec{X}_j^t$ and $\vec{X}_k^t$ are pollen gametes obtained from different flowers of the same plant species, which is equivalent to the choice of two solutions (j and k) chosen randomly from the population. The variable ϵ is drawn from a uniform distribution in [0, 1], it is used to implement a random walk [98,99].

As stated above, switching between local and global pollination is controlled by a probability P. Yang in his work stated that the best value has been found to be equal to 0.8 for most applications [98–100]. Pseudo-code of Algorithm 1 summarises the framework of the flower pollination algorithm.

Algorithm 1. The Flower Pollination Algorithm

```
Objective function Q(X), X = {x1, ..., xD}.
Initialise a population of N flowers in random positions.
Find the best solution G in the initial population.
Set the switch probability P ∈ [0, 1].
while t< Max Generation do
  for i = 1 : N (all N flowers in the population) do
    if rand < P then
      Draw a D-dimensional step vector L which obeys a Lévy distribution.
      Do global pollination via X_i^{t+1} = X_i^t + L·(X_i^t − G).
    else
      Draw ε from a uniform distribution in [0,1].
      Choose randomly X_j and X_k from the population.
      Do local pollination via X_i^{t+1} = X_i^t + ε·(X_j^t − X_k^t).
    end if
    Evaluate the newly-generated solution X_i^{t+1}.
    If the newly-generated solution is better, replace X_i^t by X_i^{t+1}.
  end for
  Update the current best solution G.
end while
```

2.2. Discretisation approaches for metaheuristics

A potential solution to continuous optimisation problems with D variables is encoded with D real values. Thus, the search space can be modelled as a D-dimensional hypercube ($\mathbb{R}^D$) (*phenotype space*), where solution's values can be updated smoothly within an infinite set of real numbers [93], whereas a solution to binary optimisation problems is encoded with distinct binary values. Therefore, the search space can be modelled as a D-Boolean lattice ($\{1, 0\}^D$) (*genotype space*) [93], where each solution is updated across the corner of the hypercube.

On the basis of the difference between the above-mentioned search spaces, some metaheuristics, such as the particle swarm optimisation algorithm, were initially designed to deal with continuous problems, while others, such as the genetic algorithm, were designed with the capacity of handling binary problems. This problem-dependency of metaheuristics is due to many factors. The principal ones are the type of search process (operators) that they use and the solution's representation they adopt. This problem-dependency restricts the use of a metaheuristic to a specific type of problem. In addition, an optimal use of a given metaheuristic requires advanced knowledge on both the metaheuristic itself and the type of problem being tackled. Thus, the use of such algorithm is limited mainly to experts within an abstract research field rather than non-experts within industrial and real-life environment. These facts made the design of algorithms that can handle both binary and continuous problem, one of today's hot topic of research.

Technically speaking, when adapting a continuous metaheuristic to deal with binary optimisation problems, the adaptation occurs mainly at two levels. The first one is the solution representation and the second is the dynamics of the algorithm (operators). The solution can be adapted by using either a binary or grey-coded representation. This fact brings the adaptation of a given metaheuristic to the adaptation of its operators. In fact, the original search mechanisms of the algorithm were designed to operate on real-valued search space. So, the whole difficulty lies in adapting this dynamics to work in the binary space as efficiently as it works in the continuous one.

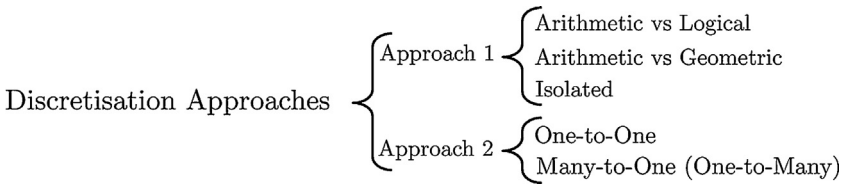

Fig. 1. Discretisation approaches for metaheuristics.

Two main approaches of adaptation exist in the literature (see Fig. 1). The first one is to directly act on the operators of the algorithm (*dynamics*) in order to make them work on binary solutions. This approach can be also further subdivided into three other subcategories. The first one consists of substituting the arithmetic operators in the algorithm by logical ones [22,41,96,102], while the second subcategory consists of finding the corresponding geometric interpretation of the algorithm's operators in Hamming space [8–10]. The final subcategory regroups some isolated strategies that have been proposed in the literature, but none of them followed one of the above-mentioned two subcategories [16,17,101]. Nevertheless, the use of this first approach is still limited to some specific metaheuristics, and thus cannot be applied to all algorithms. In addition, the resulting algorithms ends up by being simple variants of the genetic algorithm [31].

In the second approach, the original operators and structure of the algorithm are kept, but a mapping mechanism is inserted in order to map the real-valued solutions produced by the algorithm into binary ones that can be used to solve the problem. This approach can be further subdivided into two subcategories according to the scheme *A*-to-*B*: one-to-one and many-to-one (or one-to-many), where *A* represents the number of real-coded solutions in the phenotype space and *B* the number of binary solutions that they could represent in the genotype space and vice versa. However, a complex mapping function may introduce additional nonlinearities and other mathematical difficulties, which can hinder the search process substantially and affect the algorithm's efficiency [92,104]. Thus, an efficient design of a mapping technique should be done by avoiding issues such as the *Hughes effect* (i.e. adding extra dimensions to the problem search space), *computational complexity* (i.e. adding extra calculation during the mapping process), and by avoiding the loss of precision or *Hamming Cliffs effects* [75] when mapping continuous solutions to binary ones.

On the other hand, unlike the first approach, this one can be theoretically applied to all metaheuristics. In addition, it allows conserving the original framework (i.e. solution representation, operators, architecture, etc.) of the metaheuristic. Thus, it can be applied on binary problems but still can solve continuous ones. On the basis of this analysis, the second approach is the one studied in our work.

2.3. Antenna positioning problem

As a problem formulation of the antenna positioning problem, we opt for the one given by Calegari et al. [67,69] on the basis of the fact that this model is the one used in real-life cellular networks deployment. In addition, this model is a generation-independent, which allow its use in different generations of cellular networks such as 2G, GPRS, EDGE, 3G, 3G$^+$, LTE and 4G [88,90].

The antenna positioning problem aims at finding the smallest subset of base station locations among a set of potential locations in a way to ensure the largest possible coverage for a given geographical area. The APP can be modelled as a graph problem, since it recalls other problems in graph theory such as the Minimum Dominating Set (MDS) [68,88], the Maximum Independent Set (MIS) [71] and the Unicast Set Covering Problem (USCP) [26,28].

Let L be the set of all potentially covered areas and M the set of all potential locations of base stations. Suppose G defined by Formula (6), is a graph in which E is the set of edges verifying that each transmitter is linked to the area it covers. One seeks the minimum subset of transmitters that covers the maximum surface of a given area. In other words, the objective is to find a subset $M' \subseteq M$, such that $|M'|$ is minimised and $|Neighbours(M', E)|$ defined by Formula (7) is maximised [68]; where *Neighbours* represents the set of covered areas, and M' represents the set of transmitters used to cover these areas,

$$G = (M \cup L, E) \tag{6}$$

and

$$|Neighbours(M', E)| = \{u \in L \mid \exists v \in M', (u, v) \in E\}. \tag{7}$$

Fig. 2(a) gives the representation of nodes of a graph representing a network (antennas and covered areas), while Fig. 2(b) represents the projection of these nodes on the network. Fig. 2(c) illustrates what the segments of the graph are representing in real networks. Fig. 2(d) goes back to the graph representation by modelling the APP as a graph problem. Finally, Fig. 2(e) is an enhanced representation of the APP using bipartite graphs.

A Base Transceiver Station (BTS) is a radio transmitting device with a specific type of coverage (see Fig. 3). In this work, we use three types of coverage introduced in [28]: the squared,

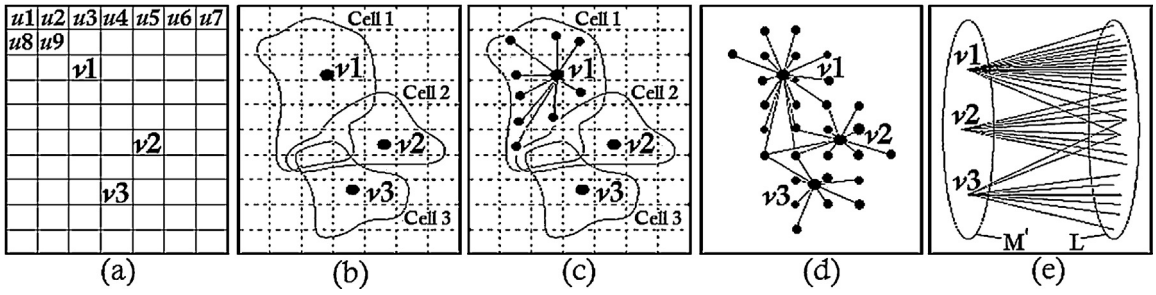

Fig. 2. A graph-based representation of the APP.

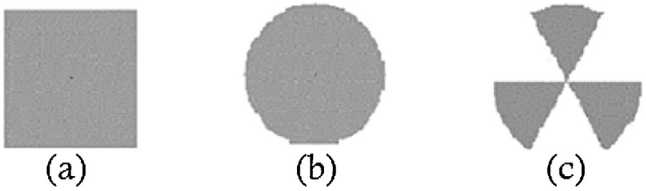

Fig. 3. Antenna coverage models: (a) squared, (b) omnidirectional and (c) directive.

omnidirectional and directive coverages. A cell is a part of a geographical area that is covered by a BTS.

In this work, we address the most practical modelling of the antenna positioning problem with two objectives: maximising the covered area while minimising the number of base stations used. The working area is discretised in a rectangular grid with H_d and V_d dimensions (see Fig. 4), in which $\{site_1, site_2, site_3, \ldots, site_D\}$ is the set of potentially preselected sites where the antennas can be placed. Each potential site location is identified by its Cartesian coordinates $\{site_1 = (x_1, y_1), site_2 = (x_2, y_2), site_3 = (x_3, y_3), \ldots, site_D = (x_D, y_D)\}$.

3. Related work

In this section, we perform a literature review of both flower pollination algorithm and antenna positioning problem.

3.1. The flower pollination algorithm

Several works have studied this recently proposed algorithm. At first, a flower pollination algorithm for mono-objective optimisation was proposed in [98]. Then, a multiobjective version of the FPA was introduced in [99]. Later, the flower pollination algorithm was used for solving academic and industrial problems.

The calculation of double retrograde dew points was solved in [38] using the FPA. Then, the flower pollination algorithm was used in [77] for solving the economic load dispatch problem. The authors in [66] used the canonical flower pollination algorithm for solving the sudoku puzzles problem. In [79], the authors proposed a novel FPA based on dimension by dimension improvement. A comparative study between the FPA and the bat algorithm was conducted in [64]. The authors in [51], used the FPA to solve the optimal reactive power dispatch problem. In [52], the FPA was hybridised with the particle swarm optimisation algorithm to solve the active power loss. A chaotic FPA was introduced in [40,65]. The FPA was also used in [59] for wireless sensor network optimisation. The authors in [57] used the FPA to solve the optimal control problem in a multi-machine system. The authors in [54] proposed a hybrid FPA for global constrained optimisation. Finally, a study on the FPA for continuous optimisation was conducted in [6,85].

Lately, a binary version of the FPA was proposed by Rodrigues et al. in [25]. However, that work was a bit simplistic. The work of that authors suffered from several shortcomings at two main levels. The first shortcoming is coming from the foundations of the paper. In fact, the paper proposes a binary version of the flower pollination algorithm based on the mapping techniques, while many other discretisation approaches exist. The authors only used the sigmoid function as a mapping technique, while several mapping techniques were proposed in the literature. In addition, the authors used one mapping scheme, while other schemes exist. No literature review or analysis was performed in order to support any of the above-mentioned choices.

The second drawback is coming from the experimental and validation section of the paper. In fact, the design of the latter does not allow to have a reliable assessment of the performances of the proposed binary variant of the flower pollination algorithm. First, the authors of that work have chosen the feature selection task as benchmark problem. The latter does not represent a wide range of instance's data types, sizes or complexities. Second, the comparison was not made against the top-ranked state-of-the-art algorithms used to solve the feature selection problem specifically, or binary problems in general. Third, the design and the parameter setting of the experiments do not allow performing a reliable performance assessment. In fact, the population size, the number of iterations and executions were set to: 30, 100 and 25 respectively without any scientific discussion. In addition, those parameters were not the ones used in the articles from which the algorithms taken as a comparison basis were proposed. The use of such values will not allow neither to properly analyse the behaviour of the proposed variant nor performing reliable statistic tests. Finally, the parameters of the proposed variant of the flower pollination algorithm were set empirically and no fine-grained tuning was performed.

In addition, the authors of that work used totally different instances and problem formulation from the ones used in the articles from which the algorithms taken as a comparison basis were proposed. These facts will not allow reproducing the same efficiency as the one these algorithms showed in their original papers.

Fourth, many evaluation aspects have been discarded such as the scalability and robustness of the proposed binary variant. In fact, only six instances of the problem were used. The choice of the size of instances does not allow the study of scalability. Only one type of data was used, which does not allow the study of robustness of the proposed variant.

Finally, the authors used few evaluation metrics to assess the efficiency of the proposed binary variant of the flower pollination algorithm. The authors of that work used only the mean of the results obtained through all executions as a performance metric. Other important metrics such as the best and the worst solutions and the standard deviation metrics were discarded. In addition, the statistical analysis was a bit simplistic, since no discussion of the choice of tests or their parameters was given.

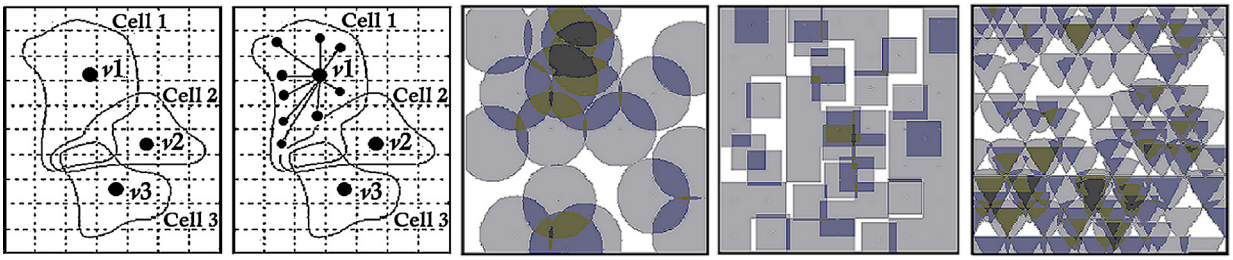

Fig. 4. Representation of the discretised area.

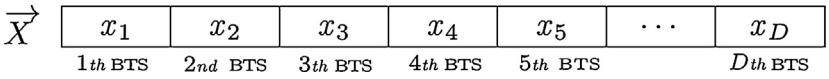

Fig. 5. A typical solution representation.

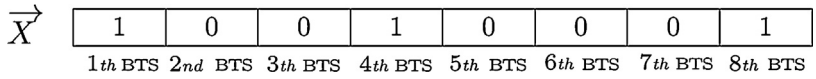

Fig. 6. Example of the solution representation.

3.2. Antenna positioning problem

The first works on the APP were mainly done within the STORMS[1] project and were mostly dedicated to the formulation and the modelling of the problem. The first general formal definition of the APP was introduced in [15], then a more detailed formulation of the problem recalling the Minimum Dominating Set Problem (MDSP) was presented later in [69]. In parallel, the first mathematical formulation of the APP was introduced in [67,69], where the authors used a parallel island-based version of the genetic algorithm to solve the APP. Another detailed formulation of the APP recalling the Unicast Set Cover Problem (USCP) was given in [68]; the authors of that work used a standard steady state genetic algorithm and a parallel island-based genetic algorithm. The two previously-mentioned formulations were grouped in the work of Kuonen et al. [72], where the authors first used a simple greedy algorithm and then a genetic algorithm to tackle the APP.

A simple greedy algorithm, a weighted set genetic algorithm and a parallel island-based genetic algorithm were used in [71] to solve the APP. Then, a sequential and a parallel steady state genetic algorithms were proposed in [26]. A parallel genetic algorithm for the base station placement was introduced in [27]. A differential evolution algorithm with problem-dependent operators was presented in [91].

A faster differential evolution algorithm with more suitable operators for the APP was proposed in [89], then Simulated Annealing (SA) and the Population-Based Incremental Learning (PBIL), the canonical Differential Evolution (DE) and an evolutionary-based algorithm (CHC) were used in [61] to solve it. In [28], the authors used a Simulated Annealing (SA), a Steady State and Generational Genetic Algorithms (SSGA, GenGA) and an evolutionary-based algorithm (CHC) to tackle the APP. Furthermore, a multiobjective evolutionary-based algorithm (MOCHC) was proposed in [13] to solve the antenna positioning problem. Several evolutionary algorithms were also presented in [62] to solve it.

Three evolutionary algorithms which are the genetic algorithm, the Memetic Algorithm (MA), and the Chromosome Appearance Probability Matrix algorithm (CAPM) were used in [103] to solve the APP. The authors of [88,90] used several evolutionary and non-evolutionary algorithms for benchmarking several solvers of the APP. Several multiobjective algorithms such as the Non-Dominated Sorting Genetic Algorithm II (NSGA-II), the Strength Pareto Evolutionary Algorithm 2 (SPEA2) and the adaptive and non-adaptive Indicator-Based Evolutionary Algorithm (IBEA) were also used in [19] to tackle a multiobjective formulation of the APP. Then, a parallel hyper-heuristic was devised in [20] for the antenna positioning problem. A multiobjective formulation of the APP was proposed in [18]. Several parallel island-based algorithms were designed in [30] for the APP; and finally a differential evolution and a binary bat algorithm for solving the APP were recently proposed in [45] and [105], respectively.

[1] STORMS: Software Tools for the Optimisation of Resources in Mobile Systems.

4. The binary flower pollination algorithm

In this section, we introduce the proposed Binary Flower Pollination Algorithm (BFPA). Firstly, we start by explaining the solution representation and the objective function of the antenna positioning problem. Then, we present the mapping techniques used to discretise the flower pollination algorithm. Finally, the search process of the resulting binary variants of the flower pollination algorithm is explained.

4.1. Problem formulation

In this section, we present both the solution representation and objective function introduced by Calegari et al. [67,69] within their modelling of the antenna positioning problem.

4.1.1. Solution representation

A potential solution of the APP can be represented as a vector of integers described as follows. Each vector $\vec{X}=\{x_1, x_2, x_3, \ldots, x_D\}$ represents a potential configuration of the mobile network. The number of elements, D, of each vector represents the number of potential candidate sites where BTSs can be placed.

Each element of the solution vector represents a base station in the network (i.e. $j=1$ in the solution vector represents the first BTS in the network, $j=2$ for the second BTS, …, $j=D$ for the last BTS). Each element is strictly binary-valued: $x_j \in \vec{X} / x_j = 1 \vee x_j = 0$. If $x_j=1$, the jth base station is selected, otherwise it is discarded. The vector presented in Fig. 5 illustrates this representation.

The vector in Fig. 6 is an example of a potential solution for the APP. It represents a configuration for a network with eight available locations, where the 1st, 4th and the 8th BTSs are selected, while the 2nd, 3rd, 5th, 6th and the 7th BTSs are discarded.

4.1.2. Objective function

The objective function introduced by Calegari et al. [67,69] within their modelling of the APP is defined by Formula (8).

$$Maximize: Q(\vec{X}) = \frac{C_R^{\alpha}}{N_A(\vec{X})} \tag{8}$$

with:

$$C_R = \frac{C_A}{T_A} * 100 \tag{9}$$

$$C_A = \sum_{i=1}^{V_d} \sum_{j=1}^{H_d} C_{S_{ij}}, \quad C_{S_{ij}} \in \{0, 1\} \tag{10}$$

$$T_A = H_d * V_d \tag{11}$$

$$N_A(\vec{X}) = \sum_{k=1}^{D} x_k, x_k \in \{0, 1\} \tag{12}$$

The APP fitness function Q is used to assess the quality of a given individual $\vec{X}$, where $\vec{X} = \{x_1, x_2, \ldots, x_D\}$ represents a potential configuration of the network (i.e. solution). Each element x_k from that

solution, represents the state (positioned or not) of the kth BTS in the network, where $k = 1, \ldots, D$ and D is the number of available sites where BTSs can be placed. The variable x_k can take the value 1 if the kth BTS is deployed and 0 otherwise.

The variable N_A represents the number of deployed antennas, while C_R is the ratio between the surface of the covered area C_A and the total surface of the working area T_A. The working area is modelled as a discrete cell grid with ($H_d * V_d$) size, where variables H_d and V_d represent respectively the horizontal and vertical dimensions of that grid. The variable C_A corresponds to the number of cells that are covered by at least one antenna, where C_S represents the state (covered or uncovered) of a given cell. It can take the value 1 if the cell is covered and 0 otherwise. A cell is said to be covered if it is located within the coverage perimeter of a given antenna. The coverage perimeter of an antenna is defined as the whole set of cells that falls within its coverage range (i.e. radius) including the cell where the BTS itself is placed (see Table 2). It is worth to mention also that a cell can be covered by one or many antennas. In both cases the variable C_S of that cell takes the value 1.

The variable T_A corresponds to the total surface of the working area, which is defined as the number of cells (covered and uncovered) in that area. It is to be noted that the variables C_A, T_A and the radius of an antenna are expressed in terms of cells.

Taking now the parameter α, some researchers such as Segura [19] stated that a decision maker must select its value. The latter is tuned considering the importance given to the coverage in relation with the number of deployed BTSs. However, since there is usually no information about the quality of the solutions which can be achieved, such a task is very difficult. Thus, obtaining the desired solutions is hard, and probably several values of α must be tested. For this reason, other works such as [72] stated that α is a parameter greater than 1 in order to favour the coverage. The same authors stated later in [73] that α is a parameter that can be tuned to favour the service ratio with respect to the number of BTSs used.

The authors of that work showed that for α between 0.5 and 1, only one BTS is chosen (that with the largest cell), for α between 1 and 1.5 the coverage is not acceptable (<55%), for α between 1.5 and 4 the coverage grows quickly, while the number of BTSs grows slowly and finally, for α between 4 and 10 the coverage remains almost constant, while the number of BTSs keeps growing.

A good value for α should thus be chosen from the interval [1.5,4]. Other works, such as [67–70], stated that the parameter α can be tuned to favour the cover rate with respect to the number of transmitters. They also stated that so far, telecommunication companies have considered that the results obtained with $\alpha = 2$ are of best quality. With respect to these findings, all the works achieved after treating the antenna positioning problem used the value $\alpha = 2$ [13,18,26–28,30]. Recently, many other works treating the antenna positioning problem used the value $\alpha = 2$ as a default value [45,61,62,88–91,103].

Since the use of the value $\alpha = 2$ has already been discussed and demonstrated in the literature, we use in this work the same value for scientific relevance and for comparison purposes with state-of-the-art algorithms.

4.2. The investigated mapping techniques

In this section we present the mapping techniques used to discretise the FPA. For scientific relevance, we decided to investigate the most used techniques in the literature, and whose the efficiency is well established. In addition, these techniques are chosen to illustrate a wide range of mapping schemes: one-to-one, many-to-one (one-to-many). Thus, the mapping techniques studied in this work are the nearest-integer method [7], normalisation [34], the angle modulation technique [14] and the sigmoid function [74].

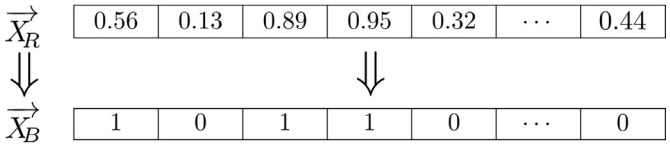

Fig. 7. The nearest-integer mapping technique.

It is to be noted also that within the next sections, $\vec{X_R}$ represents the real-coded solution (i.e. before the mapping process), where $\vec{X_R} = \{X_{R_1}, X_{R_2}, X_{R_3}, \ldots, X_{R_D}\}$ and $\vec{X_B}$ corresponds to the binary-coded solution (i.e. after the mapping process), where $\vec{X_B} = \{X_{B_1}, X_{B_2}, X_{B_3}, \ldots, X_{B_D}\}$. The variable D is the size of solution vectors $\vec{X_R}$ and $\vec{X_B}$.

4.2.1. Nearest-integer

This method has been used in many works [7,12,23,63,83]. The mapping within this technique consists of assigning to each element of $\vec{X_B}$, the nearest integer (0 or 1) obtained by rounding or truncating (up or down) its corresponding element from $\vec{X_R}$. An example of how this technique works is illustrated in Fig. 7.

4.2.2. Normalisation

Some works have been conducted on this technique [34–36]. The mapping within this method consists of two steps. Firstly, the normalisation of the solution vector $\vec{X_R}$ by linearly scaling it using Formula (13). Then secondly, the condition represented by Formula (14) is applied on the normalised vector $\vec{X_N}$ in order to produce $\vec{X_B}$.

$$X_{N_j} = \frac{(X_{R_j} + X_{R_{\min}})}{(|X_{R_{\min}}| + X_{R_{\max}})} \tag{13}$$

$$X_{B_j} = \begin{cases} 1, & \text{If } X_{N_j} \geqslant 0.5 \\ 0, & \text{Else} \end{cases} \tag{14}$$

The variables $X_{R_{\min}}$ and $X_{R_{\max}}$ represent the minimum and maximum values of the solution vector $\vec{X_R}$. X_{R_j}, X_{B_j} and X_{N_j} are the jth elements of the solution vectors $\vec{X_R}$, $\vec{X_B}$ and $\vec{X_N}$, respectively, where $j = 1, \ldots, D$. Fig. 8 is an example of how this technique works.

4.2.3. Angle modulation

Many works were based on this method [14,34–36,48,84]. The mapping within this technique is performed over two steps. Firstly, a vector $\vec{X_G} = \{X_{G_1}, X_{G_2}, X_{G_3}, \ldots, X_{G_D}\}$ is generated by using a trigonometric function from signal processing. This function is defined by Formula (15), where X_{C_1}, X_{C_2}, X_{C_3} and X_{C_4} are coefficients of the function. Roughly speaking, the generator function consists of applying $\vec{X_G}$ to a sample vector $\vec{X_R}$. The elements of latter are drawn from the interval [0,1] and have equally-spaced intervals between one and another. Secondly, the condition represented by Formula (16) is applied on the solution vector $\vec{X_G}$ in order to produce $\vec{X_B}$.

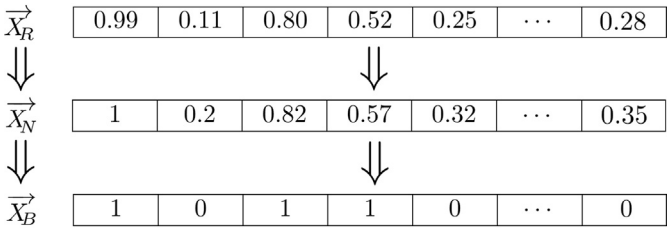

Fig. 8. The normalisation mapping technique.

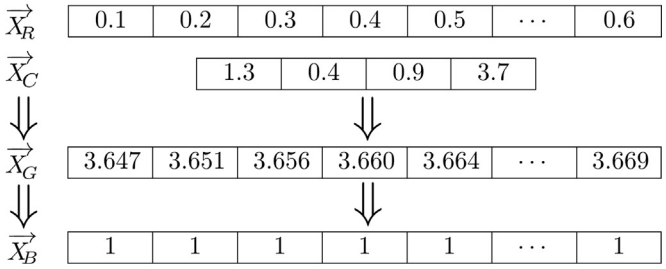

Fig. 9. The angle modulation mapping technique.

$$X_{G_j} = \sin(2\Pi(X_{R_j} - X_{C_1}) * X_{C_2} * \cos(2\Pi(X_{R_j} - X_{C_1}) * X_{C_3})) + X_{C_4} \tag{15}$$

$$X_{B_j} = \begin{cases} 1, & \text{If } X_{G_j} \geqslant 0 \\ 0, & \text{Else} \end{cases} \tag{16}$$

The particularity of this mapping technique is that instead of optimising a D-dimensional solution vector $\vec{X_R}$, we will optimise a 4-dimensional one $\vec{X_C}$, where the latter represents potential values of the coefficients X_{C_1}, X_{C_2}, X_{C_3} and X_{C_4}. Fig. 9 gives an example of how this method works.

4.2.4. Sigmoid function

Several works have used this technique [24,53,74,81,87]. The mapping within this technique is performed over two steps. Firstly, for each element X_R of the solution vector $\vec{X_R}$, a probability X_{SF} is calculated using the sigmoid function defined by Formula (17). Then, using the probability vector $\vec{X_{SF}}$, each element in the solution vector $\vec{X_B}$ will either have the value 1 or 0 by applying the condition represented by Formula (18).

$$X_{SF_j} = \frac{1}{1 + e^{X_{R_j}}} \tag{17}$$

$$X_{B_j} = \begin{cases} 1, & \text{If } R_j \leqslant X_{SF_j} \\ 0, & \text{Else} \end{cases} \tag{18}$$

Assuming that $\vec{R} = \{R_1, R_2, R_3, \ldots, R_D\}$ is a vector of randomly-generated positive numbers drawn from a standard uniform distribution. Fig. 10 is an illustration of the way this technique performs.

4.3. Search process

As its continuous counterpart, the binary flower pollination algorithm takes as an input a population of N individuals (i.e. solutions) and makes them evolve toward a fitter state by applying a series of operators. The latter consist of global or local pollination,

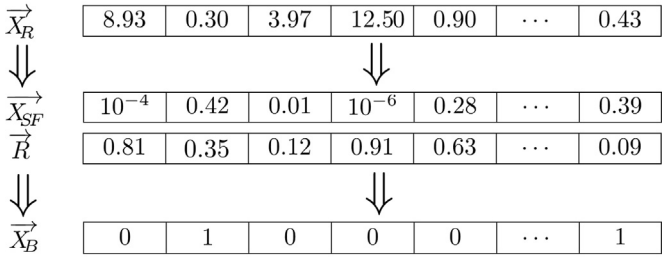

Fig. 10. The sigmoid function mapping technique.

mapping process, evaluation and finally replacement. The whole process is repeated until a given termination criterion is reached. Fig. 11 illustrates the main framework of the binary flower pollination algorithm.

In the following sections we give more insight of each one of the phases of the proposed BFPA. It is also worth to mention that a detailed pseudo-code is also available in [1].

4.3.1. Initialisation

It consists of generating a population of N binary individuals. Each individual $\vec{X_i}$ represents a potential network configuration (i.e. solution) for the APP, where $i = 1, \ldots, N$. Each element

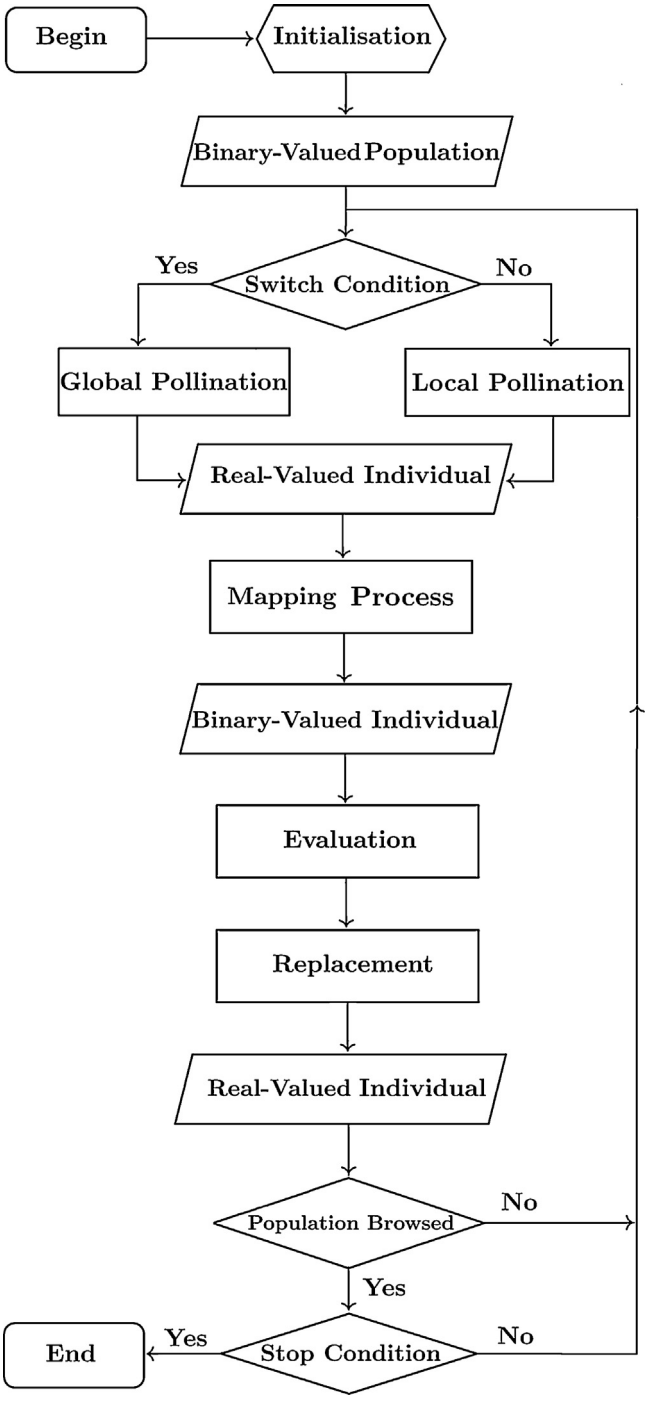

Fig. 11. Flowchart of the binary flower pollination algorithm.

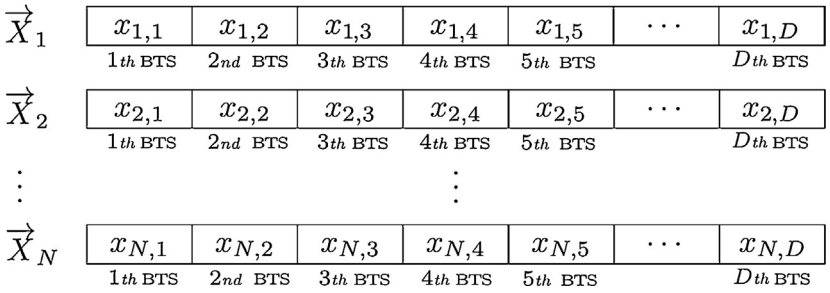

Fig. 12. Population representation for the antenna positioning problem.

$x_j \in \vec{X} / \vec{X} = \{x_1, x_2, \ldots, x_D\}$ represents the state (selected or discarded) of the *jth* BTS, where $j = 1, \ldots, D$. The size, D, of each individual represents the number of available locations where BTSs can be deployed. Fig. 12 illustrates the BFPA population representation for the case of the APP.

Each element x_{ij} in the population is initialised by generating a random number R from a standard uniform distribution. Then, if $R \geqslant 0.5$, x_{ij} will have the value 1, otherwise 0. Once the population initialised, each individual is evaluated using the fitness function defined by Formula (8). Afterwards, the individuals are ranked on the basis of their respective fitness function values and the best individual $\vec{G}$ is extracted.

Once the initialisation phase achieved, a classical iteration of the BFPA consists of applying sequentially on each individual in the population, the following phases: global or local pollination, mapping process, evaluation and replacement.

4.3.2. Global/local pollination

Once the initial population created, a perturbation mechanism, either the global or local pollination, is applied as explained in Section 2.1.2. It is worth to mention that within the first algorithm's iteration, the global or local pollination are applied on binary-valued individuals (initial population). After, starting from the second iteration, they are applied on real-valued ones.

4.3.3. Population mapping

The BFPA evolves a population of real-valued solutions, but the problem resolving requires binary-valued ones. So, a mapping process is needed, where each one of the produced solutions $\vec{X}_i$ by the global or local pollination is mapped into a binary solution $\vec{X}_{B_i}$. This process is performed using one of the mapping techniques presented in Section 4.2.

4.3.4. Evaluation

At this step, the quality of each one of the produced real-valued solution $\vec{X}_i$ is assessed by evaluating its corresponding binary-valued representation $\vec{X}_{B_i}$. This is done using the fitness function defined by Formula (8) (see Section 4.1.2).

4.3.5. Replacement

As final step, a replacement is performed in order to decide what will be the population composition for the next iteration. An elitist replacement is used, where an individual is replaced by the newly-generated one only if the latter is better. However, it is also worth to mention that the replacement is done on the real-valued population and not the binary one produced by the mapping. In fact, the proposed BFPA evolves real-valued individuals through iterations, while the binary-valued ones are temporarily created in each iteration in order to evaluate the quality of the real-valued solutions.

Once the whole population browsed, the individuals are ranked on the basis of their fitness values and the new best individual $\vec{G}$ is extracted. Then, a classical iteration of the BFPA algorithm is reiterated until a given stop criterion is reached.

The inclusion of the discretisation step results in giving birth to four binary variants of the flower pollination algorithm. The first one uses the nearest-integer method as a discretisation technique. It is called the Nearest-Integer based Binary Flower Pollination Algorithm (NI-BFPA). The second variant is the Normalisation based Flower Pollination Algorithm (N-BFPA), which is based on the normalisation method. The third variant using the sigmoid function as a mapping method is called Sigmoid Function based Binary Flower Pollination Algorithm (SF-BFPA). Finally, the fourth variant is the Angle Modulation based Flower Pollination Algorithm (AM-BFPA), which is based on the angle modulation method.

5. Experimental study and analysis

In this section, we present the experiments we conducted to practically analyse the real performances of the proposed binary FPA.

5.1. Experimental design

Our experimental design is based on the methodology proposed by Eiben, Jelasity [11] and Talbi [31]. All our tests where carried out using an Intel I3 core with 2GB RAM and a Windows 7 OS. The implementation was done using Matlab 7.12.0 (R2011a).

To investigate the robustness of the proposed FPA variants, different types of datasets are used: realistic, synthetic and randomly-generated. The realistic instance is provided by the University of Malaga representing the city of Malaga, Spain. It was used in several works [13,18,20,30,45,88,90,103] and can be found in [3]. Synthetic instances are provided by the University of Laguna, Spain. They were used in several works [13,28,30,61,62,89] and can be found in [4,5]. Randomly-generated instances are used too, they are provided by the University of Constantine II, Algeria. They were used in [105] and available in [2]. Different-sized instances are used in order to assess the scalability of the proposed approach. Instances of dimensions 149, 349, 549, 749 and 1000 are used. Each instance contains a specific number of potential positions of base stations.

Each position is identified by Cartesian coordinates (x, y). The treated area is modelled as a discrete grid where each cell represents a 15 m × 15 m surface. For the case of synthetic instances, the modelled area corresponds to a 18.5 km^2 working area, whereas the random instances correspond to a 20 km^2 working area. Finally, the realistic instance corresponds to a 27 km^2 urban area of Malaga city, Spain (see Fig. 13).

Table 1 summarises the used dataset types, their corresponding sizes and the coverage types associated with them. It is worth to mention that *circle* and *omnidirectional* represent the same type of coverage.

Unlike the majority of previous works that used only one type of coverage, we use three types of coverage (see Fig. 3):

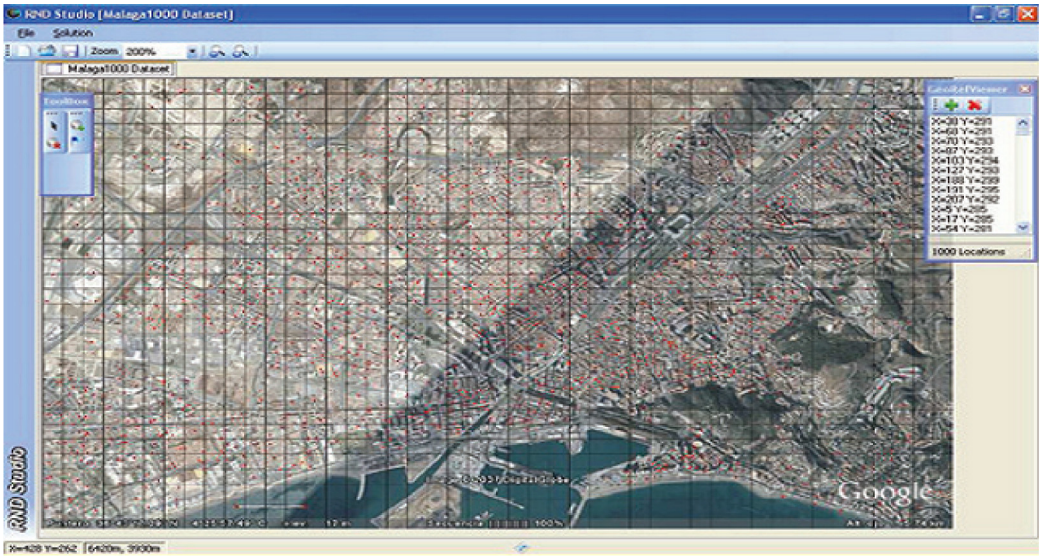

Fig. 13. Discrete representation of the realistic instance: Malaga city, Spain.

Table 1
Instances used: type, size and coverage.

Instance	Grid size	Instance size	Coverage
Synthetic	287 × 287	149	Omnidirectional Squared Directive
		349	Omnidirectional Squared Directive
Random	300 × 300	549	Omnidirectional Squared Directive
		749	Omnidirectional Squared Directive
Realistic	450 × 300	1000	Omnidirectional

squared, omnidirectional and directive, introduced in [28]. The coverage parameters, and the working area dimensions used here are those used in most literature. They are extracted from [13,27,28,30,61,62,68,89] for synthetic data. The parameters of realistic data are extracted from [13,18,20,30,45,88,90,103]; and for the randomly-generated data, the parameters are extracted from [105]. It is worth to mention that directive antennas cover one sixth of the area of omnidirectional antennas having the same radius. Table 2 summarises the used antenna coverage parameters.

The proposed variants NI-BFPA, N-BFPA, AM-BFPA and SF-BFPA, are compared to two of the top-ranked state-of-the-art metaheuristics used to solve the APP, the Population-Based Incremental Learning (PBIL) and the Differential Evolution algorithm (DE). The PBIL was proposed in [82] and used to solve the APP in [88,90], while the DE was first proposed in [78] and used to solve the APP in [61,62,88–91]. Several empirical tuning experiments allowed us to find out that for values of $\lambda \in [1, 2]$ the mean of the fitness value is altered only by (1/100) or (1/1000) of the original fitness value. So, all the variants are evaluated using the same value $\lambda = 1.5$ as advised in [37]. The switch probability parameter P of the FPA we use in this work is tuned through several fine-grained tuning experiments. Detailed results of this tuning are available in [1]. Table 3 summarises the parameters used when evaluating the four variants, whereas the PBIL and DE parameters are extracted from [88,90]. It is worth to mention that a *While* loop is used instead of a *For* loop for a fair comparison based on the maximum number of objective function evaluations (MNOFE).

It is to be noted that the symbol "S" stands for squared coverage, "C" for circle coverage (i.e. omnidirectional) and finally "D" for directive coverage.

Each algorithm is run till reaching a termination criterion of 100,000 fitness evaluations, 100 individuals and 1000 iterations are used. All the algorithms are repeated over 30 independent runs. Several results are reported such as the *best* and the *worst* fitness values, and also the *mean* and *standard deviation* of the fitness values over the 30 executions. Technical results obtained by the best individual in each experiment are reported too. The technical results are the *# antennas*, *coverage*, *overcoverage* and the *total coverage*. The metric *# antennas* represents the number of antennas used, while *coverage* corresponds to the number of cells that are covered by only one antenna. The variable *overcoverage* represents the number of cells that are covered by more than one antenna, while *total coverage* represents the number cells covered by one or many antennas. Several statistical tests such as the *one-sample Kolmogorov–Smirnov test*, the *Bartlett test*, the *Friedman two-way* and the *Kruskal–Wallis one-way* analysis of variance tests and a *post-hoc test* are performed in order to assess the quality of solutions.

5.2. Experimental results

Tables 4–8 show the numerical results obtained by the BFPAs variants, PBIL and the DE for each instance of the problem.

Table 2
Antenna coverage parameters.

	Antenna type	Squared	Omnidirectional	Directive
Synthetic data	Radius	20 cells	22 cells	22 cells
Random data	Radius	24 cells	26 cells	26 cells
Realistic data	Radius	None	30 cells	None

Table 3
Values of the switch probability P of the flower pollination variants.

Algorithms	Instances												
	149			349			549			749			1000
	S	C	D	S	C	D	S	C	D	S	C	D	C
NI-BFPA	0.1	0.2	0.1	0.6	0.6	0.1	0.7	0.8	0.5	0.6	0.6	0.6	0.2
N-BFPA	0.1	0.1	0.9	0.2	0.1	0.6	0.2	0.2	0.1	0.1	0.1	0.1	0.1
AM-BFPA	0.3	0.4	0.5	0.5	0.5	0.5	0.1	0.5	0.5	0.1	0.1	0.2	0.2
SF-BFPA	0.1	0.1	0.2	0.5	0.5	0.1	0.6	0.6	0.6	0.7	0.4	0.6	0.5

Table 4
Comparing the FPA variants to the PBIL and DE for Instance 149.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
149	Squared	NI-BFPA	146.193	135.098	140.528	3.462
		N-BFPA	148.117	125.602	137.167	7.212
		AM-BFPA	201.252	113.445	125.560	25.972
		SF-BFPA	143.596	128.812	135.874	4.315
		PBIL	200.466	189.520	194.208	3.154
		DE	201.252	201.252	**201.252**	10^{-13}
	Circle	NI-BFPA	127.538	119.834	122.061	2.228
		N-BFPA	122.698	107.424	114.614	3.787
		AM-BFPA	151.714	99.484	107.293	13.995
		SF-BFPA	121.669	108.868	116.210	3.443
		PBIL	148.345	140.208	144.992	2.300
		DE	153.565	151.415	**152.909**	0.760
	Directive	NI-BFPA	46.591	45.412	45.854	0.345
		N-BFPA	45.257	43.982	44.545	0.292
		AM-BFPA	43.574	42.868	43.170	0.250
		SF-BFPA	46.791	42.928	44.981	1.037
		PBIL	49.087	48.431	48.807	0.181
		DE	49.233	49.038	**49.159**	0.045

Table 5
Comparing the FPA variants to the PBIL and DE for Instance 349.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
349	Squared	NI-BFPA	101.204	94.987	97.573	1.517
		N-BFPA	132.088	120.717	126.612	3.479
		AM-BFPA	201.252	201.252	**201.252**	10^{-14}
		SF-BFPA	110.454	96.567	102.328	3.762
		PBIL	177.260	152.862	164.638	5.802
		DE	177.294	137.073	157.881	8.993
	Circle	NI-BFPA	93.799	87.329	89.932	1.666
		N-BFPA	116.690	102.013	108.347	3.731
		AM-BFPA	153.290	153.290	**153.290**	10^{-14}
		SF-BFPA	96.572	82.837	90.929	2.987
		PBIL	144.945	130.543	136.462	3.193
		DE	125.201	113.277	119.013	2.839
	Directive	NI-BFPA	45.088	43.683	44.285	0.364
		N-BFPA	45.249	43.813	44.414	0.386
		AM-BFPA	43.253	42.846	43.090	0.138
		SF-BFPA	44.843	42.209	43.843	0.659
		PBIL	53.717	52.094	**53.059**	0.447
		DE	52.596	50.655	51.458	0.527

Considering the "*Mean*" metric in Tables 4–8, one can note that the binary variants of the flower pollination algorithm outperform the PBIL algorithm in 2 out of 13 instances. In addition, they could outperform the DE in 6 out of 13 instances. The proposed BFPAs could also give competitive results to the one of DE in 2 out of 13 instances. However, the PBIL outperforms the proposed BFPAs in 11 out of 13 instances. Also, the DE could outperform the BFPAs in 4 out of 13 instances.

Also, on the basis of the metric "*Best*" in Tables 4–8, one can note the proposed variants outperform the PBIL in 4 out of 13 instances and the DE in 7 instances. On the other hand, on the basis of the same metric, the PBIL outperforms the BFPAs in 9 out of 13 instances. Besides, the DE could outperform also the BFPAs in 4 problem instances.

Tables 9 and 10 report the technical results obtained by the best individuals in each experiment. Technical metrics specific to the antenna positioning problem are reported such as # *antennas*, *coverage*, *overcoverage* and the *total coverage*. It is worth to mention that the three latter metrics are expressed in terms of *grid cells* and percentage, respectively.

On the basis of Tables 9 and 10, one can see that when considering one of the technical metrics such as the coverage, overcoverage and number of used antennas, the proposed variants of the FPA outperform or achieve competitive results to the ones obtained by the PBIL and DE algorithms in all 13 instances.

Figs. 14–26 show the objective function evolution through generations for the NI-BFPA, N-BFPA, AM-BFPA, SF-BFPA, PBIL and DE for each size of problem instance and for each type of coverage.

Table 6
Comparing the FPA variants to the PBIL and DE for Instance 549.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
549	Squared	NI-BFPA	72.393	67.920	70.000	1.328
		N-BFPA	166.349	148.404	158.100	5.178
		AM-BFPA	142.564	132.945	136.751	2.255
		SF-BFPA	134.950	118.086	127.310	5.311
		PBIL	195.225	181.509	**187.938**	3.659
		DE	162.829	151.013	155.815	2.918
	Circle	NI-BFPA	72.450	67.726	69.876	1.215
		N-BFPA	145.254	130.400	139.384	3.823
		AM-BFPA	124.345	117.732	120.409	2.004
		SF-BFPA	118.736	103.224	112.946	3.739
		PBIL	168.995	159.980	**164.694**	2.561
		DE	139.537	131.276	135.310	1.968
	Directive	NI-BFPA	48.168	46.098	47.073	0.623
		N-BFPA	59.261	55.336	57.643	1.045
		AM-BFPA	54.348	52.408	53.195	0.496
		SF-BFPA	55.040	49.365	51.807	1.323
		PBIL	70.329	67.729	**68.992**	0.621
		DE	62.899	60.065	61.249	0.636

Table 7
Comparing the FPA variants to the PBIL and DE for Instance 749.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
749	Squared	NI-BFPA	46.442	43.785	44.968	0.775
		N-BFPA	162.215	144.393	153.984	4.028
		AM-BFPA	139.279	131.965	135.514	2.240
		SF-BFPA	136.968	120.172	127.093	4.825
		PBIL	186.051	174.676	**181.270**	3.299
		DE	153.701	144.077	147.405	2.328
	Circle	NI-BFPA	46.263	43.552	44.729	0.751
		N-BFPA	147.752	130.811	136.895	4.430
		AM-BFPA	122.442	115.955	119.652	1.966
		SF-BFPA	115.401	102.420	110.525	3.637
		PBIL	165.888	157.926	**161.049**	1.777
		DE	132.280	126.504	129.186	1.545
	Directive	NI-BFPA	38.931	36.977	37.710	0.467
		N-BFPA	61.258	55.730	57.651	1.307
		AM-BFPA	53.243	52.251	52.745	0.334
		SF-BFPA	54.454	49.188	51.027	1.355
		PBIL	70.110	67.785	**68.815**	0.620
		DE	61.887	58.583	59.429	0.591

Table 8
Comparing the FPA variants to the PBIL and DE for Instance 1000.

Instance	Coverage	Algorithm	Best	Worst	Mean	STD
1000	Circle	NI-BFPA	22.423	20.963	21.717	0.434
		N-BFPA	112.492	99.155	106.046	3.446
		AM-BFPA	99.056	86.863	90.212	2.898
		SF-BFPA	85.013	68.620	76.916	4.244
		PBIL	138.820	112.869	**125.718**	5.630
		DE	114.547	106.039	110.016	2.254

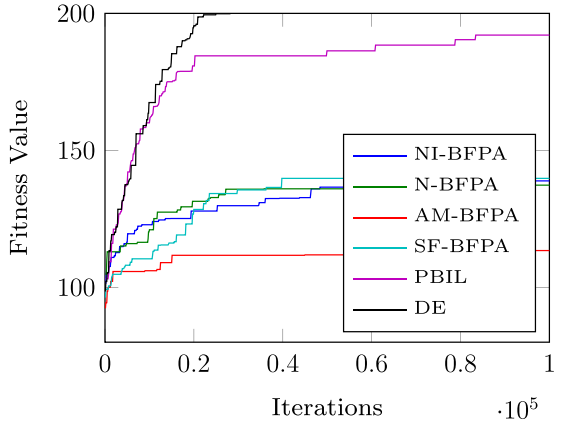

Fig. 14. Instance: 149, coverage: squared.

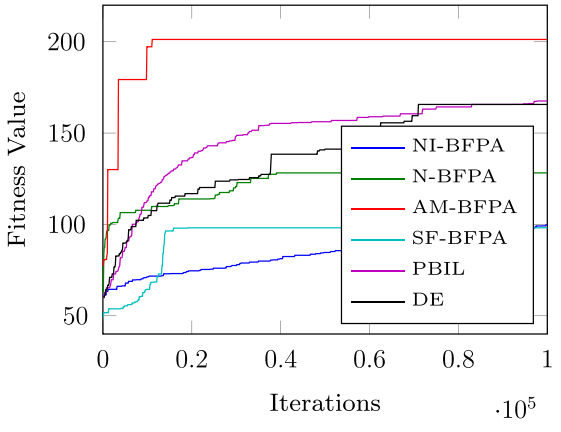

Fig. 15. Instance: 349, coverage: squared.

Table 9
Technical results obtained by the FPA variants, PBIL and the DE in Instances 149 and 349.

Instance	Coverage	Algorithm	# Antennas	Coverage (cell)	Coverage (%)	Overcoverage (cell)	Overcoverage (%)	Total coverage (cell)	Total coverage (%)
149	Squared	NI-BFPA	60	59,024	71.658	18,120	21.998	77,144	93.656
		N-BFPA	45	61,225	74.331	6022	7.311	67,247	81.641
		AM-BFPA	49	81,796	99.304	0	0.000	81,796	99.304
		SF-BFPA	57	58,497	71.018	16,023	19.452	74,520	90.471
		PBIL	49	81,599	99.065	37	0.044	81,636	99.111
		DE	49	81,796	99.304	0	0.000	81,796	99.304
	Circle	NI-BFPA	57	57,376	69.657	13,344	16.201	70,720	85.262
		N-BFPA	51	54,031	65.596	11,127	13.508	65,158	79.105
		AM-BFPA	50	68,137	82.722	3603	4.374	71,740	87.095
		SF-BFPA	58	54,042	65.609	15,152	18.395	69,194	84.004
		PBIL	50	67,256	81.652	3683	4.471	70,939	86.123
		DE	49	68,116	82.696	2833	3.439	70,949	86.136
	Directive	NI-BFPA	86	40,495	49.162	11,644	14.136	52,139	63.299
		N-BFPA	93	39,869	48.403	13,569	16.473	53,438	64.876
		AM-BFPA	92	38,392	46.610	13,760	16.705	52,152	63.315
		SF-BFPA	92	40,521	49.194	13,522	16.416	54,043	65.611
		PBIL	90	42,167	51.193	12,581	15.274	54,748	66.467
		DE	89	42,218	51.255	12,198	14.809	54,416	66.064
349	Squared	NI-BFPA	89	34,036	41.321	44,137	53.584	78,173	94.906
		N-BFPA	45	54,546	66.222	8958	10.875	63,504	77.097
		AM-BFPA	49	81,796	99.304	0	0.000	81,796	99.304
		SF-BFPA	34	45,270	54.960	5207	6.322	50,477	61.282
		PBIL	50	73,054	88.691	4491	5.452	77,545	94.143
		DE	59	55,191	67.005	18,883	22.925	74,074	89.930
	Circle	NI-BFPA	92	35,604	43.225	40,913	49.671	76,517	92.895
		N-BFPA	54	52,663	63.935	12,722	15.445	65,385	79.381
		AM-BFPA	61	73,332	89.029	6318	7.671	79,650	96.699
		SF-BFPA	41	45,387	55.102	6443	7.822	51,830	62.924
		PBIL	55	66,173	80.337	7371	8.949	73,544	89.286
		DE	61	49,513	60.111	19,434	23.594	68,947	83.126
	Directive	NI-BFPA	140	37,384	45.386	28,058	34.064	65,442	79.451
		N-BFPA	104	38,474	46.709	18,031	21.891	56,505	68.601
		AM-BFPA	99	38,210	46.389	15,690	19.048	53,900	65.437
		SF-BFPA	151	35,686	43.325	32,094	38.964	67,780	82.288
		PBIL	108	45,712	55.497	17,026	20.671	62,738	76.167
		DE	110	41,790	50.735	19,695	23.911	61,485	74.646

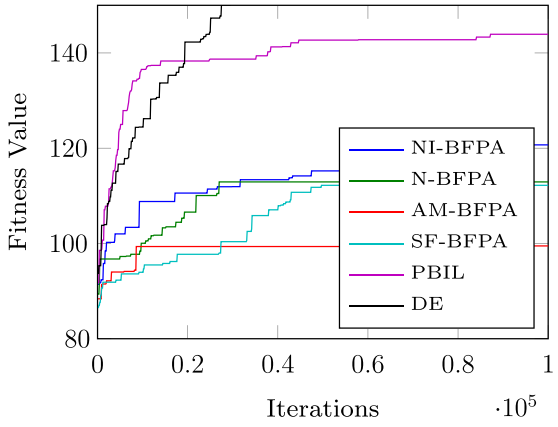

Fig. 16. Instance: 149, coverage: circle.

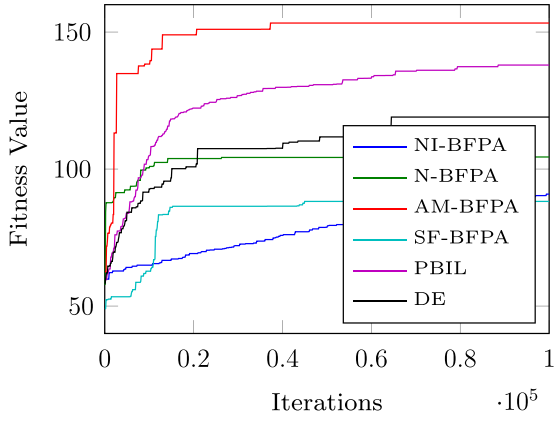

Fig. 17. Instance: 349, coverage: circle.

On the basis of Figs. 14–26, we can see that the resulting behaviour of the proposed BFPAs differs from one variant to another from the convergence rate point of view. Indeed, they show that as the size of the problem increases the difference in the convergence rate between one variant and the other increases as well.

5.3. Statistical analysis

In this section we perform statistical analysis to investigate the performances of the proposed variants for each instance. All the tests performed are hypothesis-based tests. They are defined as follows:

- H_0: defines the null-hypothesis that the assumptions are correct.
- H_1: defines the hypothesis that the assumptions are incorrect.

The assumptions are specific to each test (distribution normality, variance homogeneity, distribution equality, etc.). They are tests based on the acceptance or the rejection of the null-hypothesis H_0. They accept the null-hypothesis if: $S_T < C_V$ or $P_V > \alpha$, where S_T is the value returned by the statistical test, C_V is the critical value used, P_V is the probability value returned by the test, and α is the level of significance.

Firstly, we use the one-sample Kolmogorov–Smirnov test as a normality distribution test. Secondly, as homogeneity of variance test, we use the Bartlett test. Detailed results of these above are

Table 10
Technical results obtained by the FPA variants, PBIL and the DE in Instances 549, 749 and 1000.

Instance	Coverage	Algorithm	# Antennas	Coverage (cell)	Coverage (%)	Overcoverage (cell)	Overcoverage (%)	Total coverage (cell)	Total coverage (%)
549	Squared	NI-BFPA	136	10,344	11.493	78,958	87.731	89,302	99.224
		N-BFPA	35	56,368	62.631	12,305	13.672	68,673	76.303
		AM-BFPA	44	47,746	53.051	23,535	26.151	71,281	79.201
		SF-BFPA	30	48,188	53.542	9077	10.086	57,265	63.628
		PBIL	38	68,209	75.788	9309	10.343	77,518	86.131
		DE	48	51,115	56.794	25,510	28.344	76,625	85.139
	Circle	NI-BFPA	135	15,857	17.619	73,151	81.279	89,008	98.898
		N-BFPA	39	57,836	64.262	9903	11.003	67,739	75.266
		AM-BFPA	52	45,466	50.518	26,904	29.893	72,370	80.411
		SF-BFPA	46	45,622	50.691	20,892	23.213	66,514	73.904
		PBIL	46	67,721	75.246	11,631	12.923	79,352	88.169
		DE	54	50,168	55.742	25,608	28.453	75,776	84.196
	Directive	NI-BFPA	172	27,966	31.073	53,953	59.948	81,919	91.021
		N-BFPA	85	42,892	47.658	20,984	23.316	63,876	70.973
		AM-BFPA	87	39,446	43.829	22,440	24.933	61,886	68.762
		SF-BFPA	74	41,297	45.886	16,141	17.934	57,438	63.821
		PBIL	95	50,794	56.438	22,771	25.301	73,565	81.739
		DE	108	40,804	45.338	31,684	35.204	72,488	80.542
749	Squared	NI-BFPA	214	2772	3.081	86,951	96.612	89,723	99.692
		N-BFPA	37	55,191	61.323	14,534	16.149	69,725	77.472
		AM-BFPA	42	45,101	50.112	23,734	26.371	68,835	76.483
		SF-BFPA	25	48,342	53.713	4323	4.803	52,665	58.517
		PBIL	41	64,268	71.409	14,337	15.931	78,605	87.339
		DE	50	46,990	52.211	29,398	32.664	76,388	84.876
	Circle	NI-BFPA	215	3079	3.421	86,680	96.311	89,759	99.732
		N-BFPA	42	60,315	67.017	10,583	11.759	70,898	78.776
		AM-BFPA	46	47,022	52.247	20,522	22.802	67,544	75.049
		SF-BFPA	38	45,878	50.976	13,721	15.246	59,599	66.221
		PBIL	45	65,644	72.938	12,116	13.462	77,760	86.401
		DE	58	43,870	48.744	33,222	36.913	77,092	85.658
	Directive	NI-BFPA	227	18,031	20.034	66,575	73.972	84,606	94.007
		N-BFPA	80	44,310	49.233	18,694	20.771	63,004	70.004
		AM-BFPA	96	39,527	43.919	24,817	27.574	64,344	71.493
		SF-BFPA	77	41,503	46.114	17,163	19.071	58,666	64.753
		PBIL	96	50,603	56.226	23,233	25.814	73,836	82.041
		DE	110	40,383	44.870	31,865	35.406	72,248	80.276
1000	Circle	NI-BFPA	364	8371	6.201	118,879	88.059	127,250	90.344
		N-BFPA	40	74,656	55.301	17,411	12.897	92,067	67.081
		AM-BFPA	33	64,654	47.892	12,788	9.473	77,442	57.174
		SF-BFPA	31	54,039	40.029	15,265	11.307	69,304	51.336
		PBIL	51	87,913	65.121	26,057	19.301	113,970	84.142
		DE	60	69,706	51.634	41,924	31.055	111,630	79.764

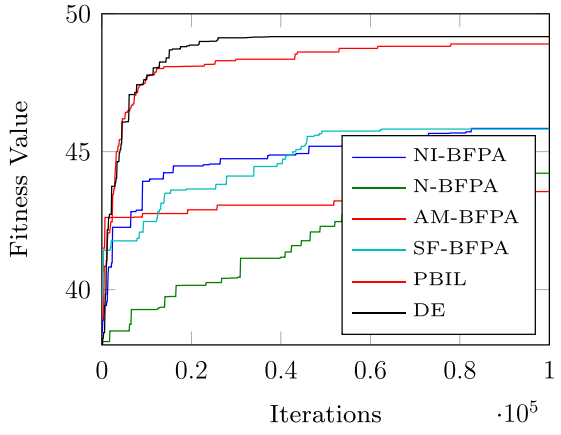

Fig. 18. Instance: 149, coverage: directive.

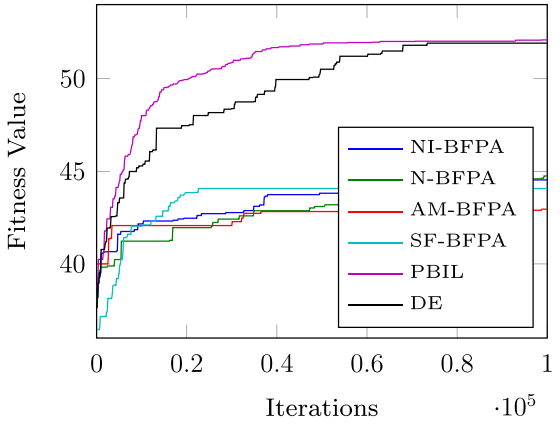

Fig. 19. Instance: 349, coverage: directive.

available in [1]. Finally, as a distribution equality test, we conduct in the next section the Friedman two-way analysis of variance and Kruskal–Wallis one-way analysis of variance tests. A detailed description of our statistical methodology is also available in [1].

5.3.1. Distribution equality tests

In this section, we conduct the tests to investigate which algorithm performs better for a specific instance. For this purpose, we conduct first a test to find if there is a difference between the *Median* of the algorithms results on a specific instance. We use non-parametric tests to find if the distributions of the results obtained

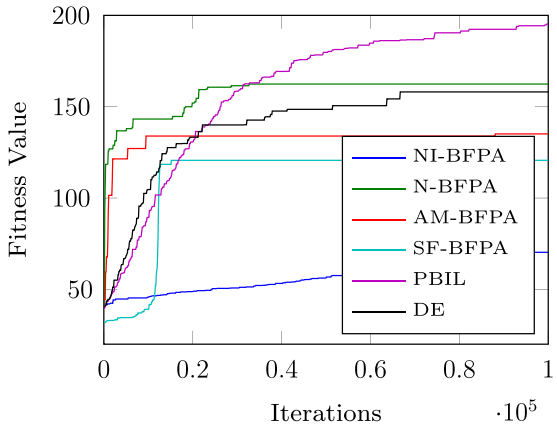

Fig. 20. Instance: 549, coverage: squared.

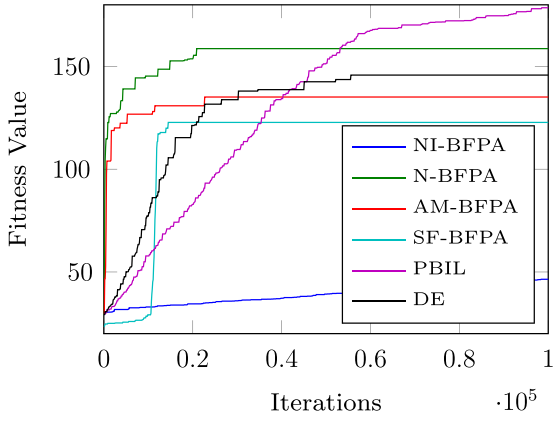

Fig. 21. Instance: 749, coverage: squared.

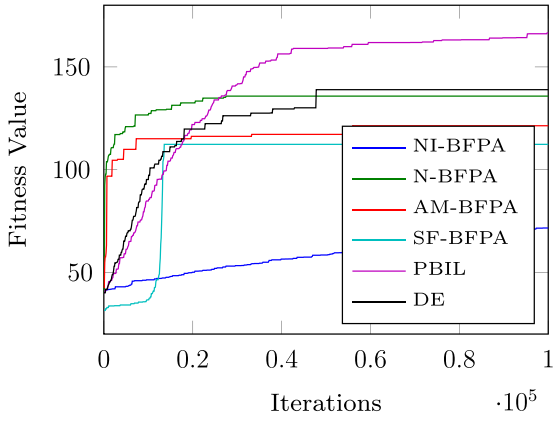

Fig. 22. Instance: 549, coverage: circle.

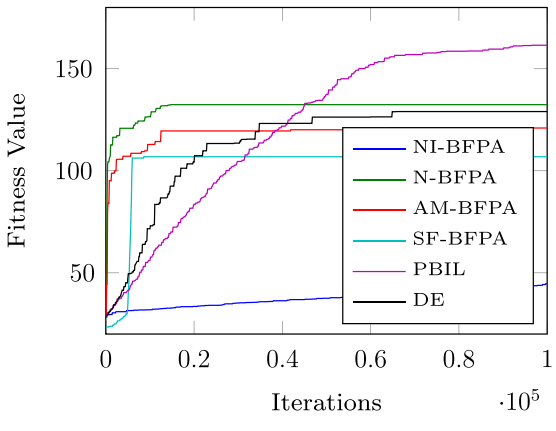

Fig. 23. Instance: 749, coverage: circle.

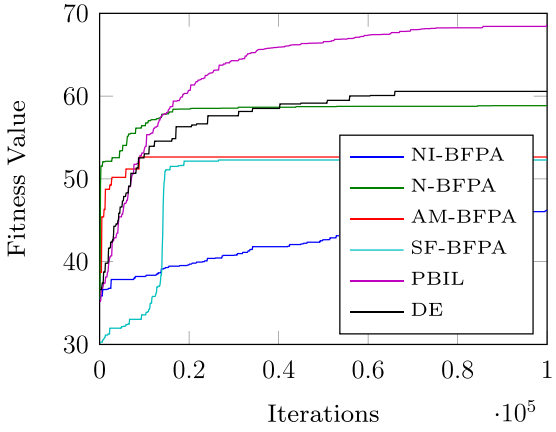

Fig. 24. Instance: 549, coverage: directive.

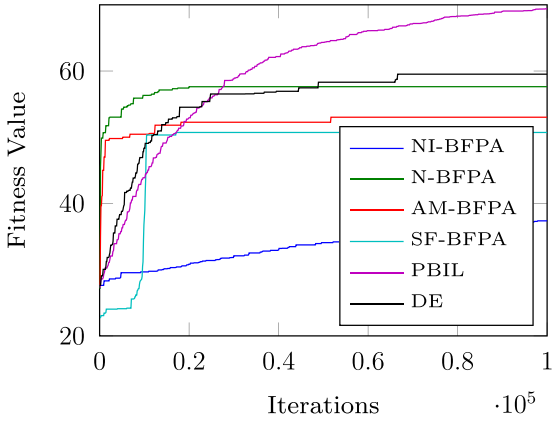

Fig. 25. Instance: 749, coverage: directive.

by the algorithms are the same or not. If the test fails, this means that at least one algorithm produces results that differ from those obtained by the others. In this case, a post-hoc test has to be done to find what algorithm is different and ultimately deduce which is the best one.

The variance analysis tests are performed on each instance. The four first instances: 149, 349, 549 and 749 include three types of coverage. For this reason, we use the Friedman two-way analysis of variance test, since it can simultaneously consider two factors of variability, where the algorithms are *factor A*, and the types of coverage are *factor B*.

For Instance 1000, there is only one coverage used which is the circle one. The Friedman two-way analysis of variance test cannot be used in this case, because there is only one factor to consider which is the algorithm's type (*factor A*). On the basis of this observation, we perform the *Kruskal–Wallis one-way analysis of variance test*. For more details, one can refer to [55,94] for the Friedman test.

For both Friedman two-way and the Kruskal–Wallis one-way analysis of variance tests, the level of significance used is 5%. In addition, having six algorithms to compare, the degree of freedom used is 5. It is calculated using the Formula $(Z-1)$, where Z is the number of algorithms. On the basis of the used level of significance and the degree of freedom, the tests are conducted using a critical value

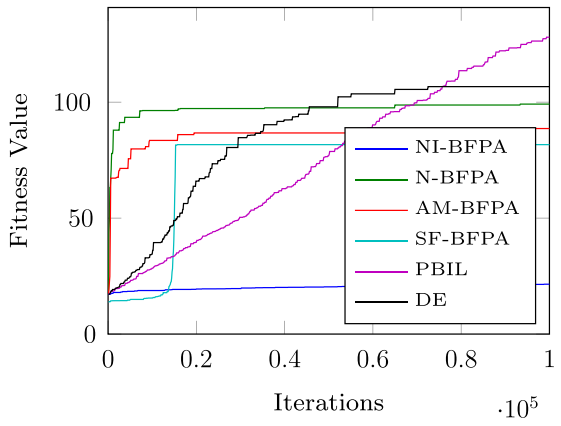

Fig. 26. Instance: 1000, coverage: circle.

equal to 11.070 according to the *Chi-square probabilities distribution table*. For both tests, the hypotheses are defined as follows:

- H_0: defines the null-hypothesis that the distribution is the same for all algorithms.
- H_1: defines the hypothesis that the distribution is not the same for all algorithms.

(A) *The Friedman two-way analysis of variance test*

The test is conducted on the results of the 90 executions obtained by each algorithm in each instance (i.e. 30 executions for each type of coverage).

As shown in Table 11, for each of Instances 149, 349, 549 and 749, the Friedman test succeeds to reject the null-hypothesis H_0. One can conclude that the distribution of the results obtained by the six algorithms is not the same. In this case, a post-hoc test is necessary to find which algorithm is different and ultimately deduce which one is better.

(B) *The Kruskal–Wallis one-way analysis of variance test*

As shown in Table 12, the test succeeds to reject the null-hypothesis H_0. Thus, one can conclude that the distribution of the results obtained by the six algorithms is not the same when using Instance 1000. In this case, a post-hoc test is also necessary to find which algorithm is different and ultimately deduce which one is better.

Table 11
Friedman two-way variance analysis test for Instances 149, 349, 549 and 749.

Instance	P_V	S_T	Null-hypothesis
149	1.133×10^{-61}	295.096	Rejected
349	3.973×10^{-40}	194.643	Rejected
549	2.834×10^{-70}	335.091	Rejected
749	1.787×10^{-70}	336.022	Rejected

Table 12
Kruskal–Wallis one-way analysis of variance test for Instance 1000.

Instance	P_V	S_T	Null-hypothesis
1000	6.430×10^{-23}	113.804	Rejected

5.3.2. Post-hoc test

A post-hoc test is performed to find where the differences occur and which one of the algorithms is better. The post-hoc performs a series of pairwise tests between two algorithms to find if there is a difference between their distributions or not. It is based on a statistical ranking of the algorithms according to the results obtained by them.

Considering that the experiments are performed for 30 executions for each type of coverage, except Instance 1000. We have 3

Table 13
Results of the post-hoc test (P_V).

Instance	Algorithms	NI-BFPA	N-BFPA	AM-BFPA	SF-BFPA	PBIL	DE	Null-hypothesis
149	NI-BFPA	/	1.000	1.000	/	/	1.000	Accepted
	N-BFPA	/	/	1.000	1.000	1.000	0.198	Accepted
	AM-BFPA	/	/	/	/	0.160	/	Accepted
	SF-BFPA	1.000	/	1.000	/	/	0.269	Accepted
	PBIL	1.000	/	/	1.000	/	1.000	Accepted
	DE	/	/	0.015	/	/	/	Rejected
349	NI-BFPA	/	1.000	/	/	/	/	Accepted
	N-BFPA	/	/	1.000	1.000	1.000	1.000	Accepted
	AM-BFPA	1.000	/	/	/	1.000	1.000	Accepted
	SF-BFPA	1.000	/	1.000	/	/	1.000	Accepted
	PBIL	0.257	/	/	0.317	/	/	Rejected
	DE	1.000	/	/	/	1.000	/	Accepted
549	NI-BFPA	/	0.215	/	/	/	/	Accepted
	N-BFPA	/	/	/	1.000	1.000	1.000	Accepted
	AM-BFPA	1.000	1.000	/	/	0.470	1.000	Accepted
	SF-BFPA	1.000	/	1.000	/	0.082	1.000	Accepted
	PBIL	0.007	/	/	/	/	/	Rejected
	DE	0.195	/	/	/	1.000	/	Accepted
749	NI-BFPA	/	0.148	/	/	/		Accepted
	N-BFPA	/	/	1.000	/	1.000	1.000	Accepted
	AM-BFPA	1.000	/	/	/	0.468	1.000	Accepted
	SF-BFPA	1.000	1.000	1.000	/	0.083	1.000	Accepted
	PBIL	0.007	/	/	/	/	/	Rejected
	DE	0.282	/	/	/	1.000	/	Accepted
1000	NI-BFPA	/	0.000	0.041	/	0.000	0.000	Rejected
	N-BFPA	/	/	/	0.012	0.013	1.000	Rejected
	AM-BFPA	/	0.512	/	/	/	1.000	Accepted
	SF-BFPA	1.000	0.293	1.000	/	/	/	Accepted
	PBIL	/	/	0.000	0.000	/	/	Rejected
	DE	/	/	0.012	0.000	0.536	/	Rejected

Table 14
Results of the statistical ranking.

Instance	Algorithms	S_R
149	NI-BFPA	62.717
	N-BFPA	40.083
	AM-BFPA	17.050
	SF-BFPA	43.217
	PBIL	89.533
	DE	**110.400**
349	NI-BFPA	27.767
	N-BFPA	53.133
	AM-BFPA	77.933
	SF-BFPA	29.967
	PBIL	**95.350**
	DE	78.850
549	NI-BFPA	10.500
	N-BFPA	80.033
	AM-BFPA	49.367
	SF-BFPA	31.633
	PBIL	**110.500**
	DE	80.967
749	NI-BFPA	10.500
	N-BFPA	83.783
	AM-BFPA	49.300
	SF-BFPA	31.700
	PBIL	**110.500**
	DE	77.217
1000	NI-BFPA	10.500
	N-BFPA	73.800
	AM-BFPA	50.500
	SF-BFPA	30.500
	PBIL	**110.400**
	DE	87.300

types of coverage (i.e. 90 executions), except Instance 1000 we performed 30 executions. When performing the post-hoc procedure, we rank the algorithms according to their performances in each one of the 90 executions. Then, the mean of these ranks is taken. The variable S_R in Table 14 shows the results of this ranking. More detailed results of the post-hoc test are available in [1].

The level of significance used in this test is 5%. Except for the Instance 349 the use of a significance level of 5% will result in a *type II error* (i.e. null-hypothesis is false, but the multiple comparison test fails to reject it). So, for the case of Instance 349, 90% is used as a significance level. The value of the variable P_V obtained by the post-hoc test is adjusted using the *Bonferroni-correction procedure*. The hypotheses of the test are defined as follows:

- H_0: defines the null-hypothesis that the distribution is the same for the two algorithms.
- H_1: defines the hypothesis that the distribution is not the same for the two algorithms.

5.4. Interpretation and discussion

On the basis of the results of Tables 4–8, for all types of coverage of Instance 149, the best variant found is the NI-BFPA. When tackling Instance 349, the best variant was found to be the AM-BFPA (squared and circle coverages), whereas for directive coverage, the best variant is assessed to be the N-BFPA. The N-BFPA is proven to be the best variant when addressing Instances 549, 749 and 1000 and this for all types of coverage.

Also, on the basis of the metric "*Mean*" of Tables 4–8, one can note that the worst variant when tackling Instance 149 is the AM-BFPA, while for the rest of the instances: 349, 549, 749 and 1000, the worst variant is the NI-BFPA. Taking now the metric "*Best*", the worst variants in Instance 149 are the AM-BFPA (squared and circle coverages) and SF-BFPA (directive coverage), whereas for the remaining instances: 149, 349, 549 and 1000, the worst variant is still the NI-BFPA.

On the basis of the metric "*Mean*" shown in Tables 4–8, one can note that the PBIL algorithm outperforms the proposed variants of the FPA. But, still for Instance 349 (squared and circle coverages), the AM-BFPA variant outperforms both the PBIL and DE algorithms. In addition, one can see that the N-BFPA could outperform the DE in 4 out of 13 instances (Instance 549 and 749 with square and circle coverages). Also, on the basis of the metric "*Best*" shown in Tables 4–8, the AM-BFPA outperforms the PBIL in Instances 149 and 349 (squared and circle coverages). In addition, the N-BFPA variant shows close and promising results to the ones obtained by the PBIL when tackling Instances 549, 749 and 1000. It is worth to mention also that the variant AM-BFPA could outperform the DE in 3 out of 13 instances (Instance 149 with squared coverage and Instance 349 with both squared and circle coverages). Also, the variant N-BFPA could outperform the DE in 5 out of 13 instances (Instance 549 with squared and circle coverages and Instance 749 with all types of coverage).

On the basis of the metrics "*Mean*", "*Best*" and "*STD*" of Tables 4–8, one can observe that the difference between the proposed BFPAs is not very big (the order of 1/100). The magnitude of the difference increases as the size of the problem increases. It is worth to mention that whatever the size of the problem tackled is, the difference between the variants is the smallest when tackling the directive coverage.

On the basis of the metric "*STD*" in Tables 4–8, there is no real observable change as the problem size increases. So, even if one can observe that the difference in the efficiency between the proposed variants increases, the stability remains unaffected.

When observing the objective function evolution through generations in Figs. 14–26, the proposed binary variants of the FPA show almost the same convergence rate when tackling small and average instances (149 and 349). As the size of the instances increases, the differences in the convergence from one variant to another are more observable. For instance, as the size of the problem increases the NI-BFPA convergence rate decreases, whereas for the case of SF-BFPA, N-BFPA and AM-BFPA, the convergence rate increases as the size of the problem increases. Finally, the convergence rate of the PBIL algorithm is slow but regular.

It is also worth to mention that the most severe change of behaviour is the one of the NI-BFPA, then the one of the SF-BFPA. The behaviour of the AM-BFPA and N-BFPA is more stable. This explains why results of the NI-BFPA are declining as the size of the problem increases, while results obtained by N-BFPA are still stable (see Tables 4–8).

On the basis of the results of Tables 4–8, and those of Figs. 14–26, one can see that the efficiency of the mapping techniques is mainly driven by the size and complexity of the instances to be solved (i.e. type of the coverage) and not the type of data (i.e. realistic, synthetic and random).

The efficiency of all mapping schemes is almost similar when tackling small instances. But, as the size increases, the mapping techniques based on the scheme one-to-one (i.e. the rounding technique) are those whose the efficiency decreases substantially, whereas mapping techniques based on the schemes one-to-many (many-to-one) (i.e. the remaining mapping techniques) have more stable efficiency, since they are more robust than the first ones when tackling bigger instances.

Although, at a given size of problem, even the mapping techniques based on the many-to-one (one-to-many) schemes start to be different. For instance, the AM-BFPA variant was assessed to be the best when tackling Instance 349, whereas for the remaining instances the N-BFPA is assessed to be the best one. Despite that the efficiency of the mapping techniques differs as the size of the

problem increases, but their stability is still unaffected whatever the size of the problem tackled is.

As the size of the problem increases, a change of the behaviour can be noticed in the convergence rate. Despite that, the influence is not identical for all mapping techniques. Indeed, mapping techniques based on the one-to-one scheme (i.e. the rounding technique) see their convergence rate substantially affected, while mapping techniques based on many-to-one (one-to-many) scheme seem to be more stable and robust as the size of the problem increases.

On the basis of the technical results illustrated in Table 9, one can notice that for Instance 149 (squared and circle coverages) and Instance 349 (squared coverage), the AM-BFPA gave better technical results than the PBIL (i.e. fewer antennas, more coverage and less overcoverage). Besides, the NI-BFPA outperforms the PBIL for Instance 149 (directive coverage). The AM-BFPA gave similar results to the ones obtained by the PBIL for Instance 349 (circle and directive coverages).

For the rest of instances: 549, 749 and 1000, one can also note that the PBIL and the DE never outperform the variants of the FPA, because in each type of coverage and for each size of instance, one can find that one of the variants of the FPA outperform the PBIL and the DE for a given criterion (i.e. *# antennas, coverage or overcoverage*). These findings show that the binary variants of the FPA outperform or give very competitive technical results compared to the ones of the PBIL and DE.

As shown in statistical Table 13, the two algorithms that differ in Instance 149 are the AM-BFPA and the DE. For Instance 349, two couples of algorithm are different: NI-BFPA and the PBIL, the SF-BFPA and the PBIL. When tackling Instances 549 and 749, the two algorithms that differ are the NI-BFPA and the PBIL, whereas for Instance 1000, six couples of algorithms differ: NI-BFPA and N-BFPA, NI-BFPA and AM-BFPA, NI-BFPA and PBIL, NI-BFPA and the PBIL, NI-BFPA and the DE, N-BFPA and SF-BFPA, N-BFPA and the PBIL, AM-BFPA and the PBIL, SF-BFPA and the PBIL, the DE and AM-BFPA, DE and PBIL and finally the DE and SF-BFPA.

On the basis of the mean of ranks of each algorithm in Table 14, one can note that the N-BFPA outperforms the DE in Instance 749 and give competitive results to the ones obtained by the DE in Instances 349 and 549. However, the DE could outperform the BFPAs variants in Instances 149 and 1000. In addition, one can observe that the PBIL algorithm outperforms the four BFPA variants and this for all instances. But, one can also note that the best variant N-BFPA is similar to the PBIL and DE, and gave very promising results for all instances on the basis of the mean of statistical ranking (see Table 14). This is illustrated by the fact that no statistical differences are found between the N-BFPA and the PBIL or the N-BFPA and DE, and this for all instances. This, highlight the fact that when considering all the results obtained through executions, N-BFPA gave results statistically as good as those obtained by the PBIL and the DE.

6. Conclusion

In this paper, a study is conducted in order to assess the claimed efficiency of the recently proposed flower pollination algorithm and the mapping techniques. Also, many research questions that still surround them have been investigated. All of this was done by proposing four binary variants of the flower pollination algorithm based on the principal mapping techniques existing in the literature. The antenna positioning problem was used as a benchmark problem for experimental validation.

The results of the experiments showed that the flower pollination algorithm gave competitive results and even outperformed the state-of-the-art algorithms for some instances. Still, the FPA has some shortcomings when compared to the PBIL algorithm. Experiments showed also that the efficiency of the mapping techniques is driven by the size and the complexity of the problem. The behaviour of the mapping techniques is mainly affected from a convergence rate point of view. The most efficient and stable mapping techniques are the ones based on the normalisation and angle modulation techniques.

As perspective of research, we intend to explore strategies of cooperation between several mapping techniques. Also, using the fuzzy logic and the quantum-inspired mechanisms to design more efficient mapping techniques. Finally, designing adaptive mapping techniques with evolving mapping schemes and ultimately propose more efficient binary algorithms.

References

[1] Appendix. http://tinyurl.com/Appendix-Detailed.
[2] The Random Benchmark Instance of the STORMS Project Model: Instance of 549 and 749 Positions. http://www.fichier-rar.fr/2014/10/03/random-instance/.
[3] The Realistic Benchmark Instance of the STORMS Project Model: Instance of 1000 Positions. http://oplink.lcc.uma.es/problems/rnd.html.
[4] The Synthetic Benchmark Instance of the STORMS Project Model: Instance of 149 Positions. http://neo.lcc.uma.es/software/more/index.php.
[5] The Synthetic Benchmark Instance of the STORMS Project Model: Instance of 349 Positions. http://www.httpoplink.lcc.uma.esproblemsrnd.html.
[6] A. Draa, On the performances of the flower pollination algorithm: qualitative and quantitative analyses, Appl. Soft Comput. 34 (C) (2015) 349–371.
[7] A. Kimiyaghalam, M. Mahdavi, A. Ashouri, M. Bagherivand, Optimal placement of PMUs for reliable observability of network under probabilistic events using BABC algorithm, in: Proceedings of the 22nd International Conference and Exhibition on Electricity Distribution (CIRED), IEEE, 2013, pp. 1–4.
[8] A. Moraglio, C. Di Chio, J. Togelius, R. Poli, Geometric particle swarm optimization, J. Artif. Evol. Appl. (2008), 11:1–11:14.
[9] A. Moraglio, J. Togelius, S. Silva, Geometric differential evolution for combinatorial and programs spaces, Evol. Comput. 21 (4) (2013) 591–624.
[10] A. Moraglio, S. Silva, Geometric differential evolution on the space of genetic programs, in: Proceedings of the 13th European Conference on Genetic Programming (EuroGP '10), vol. 6021, Springer, 2010, pp. 171–183.
[11] A.E. Eiben, M. Jelasity, A critical note on experimental research methodology in EC, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02), IEEE, 2002, pp. 582–587.
[12] A.E. Kanlikilicer, A. Keles, A.S. Uyar, Experimental analysis of binary differential evolution in dynamic environments, in: Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '07), ACM, 2007, pp. 2509–2514.
[13] A.J. Nebro, E. Alba, G. Molina, F. Chicano, F. Luna, J.J.f. Durillo, Optimal antenna placement using a new multi-objective CHC algorithm, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO), ACM, 2007, pp. 876–883.
[14] Keles Ali, Binary differential evolution for the unit commitment problem, in: Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO '07), ACM, 2007, pp. 2765–2768.
[15] B. Chamaret, S. Josselin, P. Kuonen, M. Pizarroso, Radio network optimization with maximum independent set search, in: Proceedings of the 47th Vehicular Technology Conference, vol. 2, IEEE, 1997.
[16] C. Deng, B. Zhao, Y. Yang, H. Zhang, Binary encoding differential evolution with application to combinatorial optimization problem, in: Proceedings of the Chinese Intelligent Automation Conference, vol. 255, Springer, 2013, pp. 77–84.
[17] C. Ma, L. Qian, L. Wang, M.I. Menhas, Determination of the PID controller parameters by modified binary particle swarm optimization algorithm, in: Proceedings of the Chinese Conference on Control and Decision Conference (CCDC), IEEE, 2010, pp. 2689–2694.
[18] C. Segura, E. Segredo, Y. Gonzalez, C. Leon, Multiobjectivisation of the antenna positioning problem, in: Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence (DCAI), Springer, 2011, pp. 319–327.
[19] C. Segura, Y. Gonzalez, G. Miranda, C. Leon, A multi-objective evolutionary approach for the antenna positioning problem, in: Proceedings of the 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES), Springer, 2010, pp. 51–60.
[20] C. Segura, Y. Gonzalez, G. Miranda, C. Leon, Parallel hyperheuristics for the antenna positioning problem, in: Proceedings of the 7th International Symposium on Distributed Computing and Artificial Intelligence (DCAI), Springer, 2010, pp. 471–479.
[21] C.C.O. Ramos, A.N. de Souza, G. Chiachia, A.X. Falc ao, J.P. Papa, A novel algorithm for feature selection using harmony search and its application for non-technical losses detection, Comput. Electr. Eng. 37 (6) (2011) 886–894.

[22] C.Y. Wu, K.S. Nu, Art-enhanced modified binary differential evolution algorithm for optimization, in: Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC), IEEE, 2012, pp. 1838–1843.
[23] D. Jiang, C. Peng, Z. Fan, Y. Chen, Modified binary differential evolution for solving wind farm layout optimization problems, in: Proceedings of the IEEE Symposium on Computational Intelligence for Engineering Solutions (CIES), IEEE, 2013, pp. 23–28.
[24] D. Rodrigues, L.A.M. Pereira, T.N.S. Almeida, J.P. Papa, A.N. de Souza, C.C.O. Ramos, X.S. Yang, BCS: a binary cuckoo search algorithm for feature selection, in: Proceedings of the International Symposium on Circuits and Systems (ISCAS), IEEE, 2013, pp. 465–468.
[25] D. Rodrigues, X.S. Yang, S. de, N. André, J.P. Papa, Binary flower pollination algorithm and its application to feature selection Recent Advances in Swarm Intelligence and Evolutionary Computation, vol. 585, Springer, 2015, pp. 85–100.
[26] E. Alba, Evolutionary algorithms for optimal placement of antennae in radio network design, in: Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS), IEEE Computer Society, 2004.
[27] E. Alba, F. Chicano, On the behaviour of parallel genetic algorithms for optimal placement of antennae in telecommunications, Int. J. Found. Comput. Sci. 16 (2) (2005) 343–359.
[28] E. Alba, G. Molina, J.F. Chicano, Optimal placement of antennae using metaheuristics, in: Proceedings of the 6th International Conference on Numerical Methods and Applications (NMA), vol. 4310, Springer, 2006, pp. 214–222.
[29] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, BGSA: binary gravitational search algorithm, Nat. Comput. 9 (3) (2010) 727–745.
[30] E. Segredo, C. Segura, C. Leon, On the comparison of parallel island-based models for the multiobjectivised antenna positioning problem, in: Proceedings of the 15th International Conference On Knowledge-Based and Intelligent Information and Engineering Systems (KES), Springer, 2011, pp. 32–41.
[31] E. Talbi, Metaheuristics: From Design to Implementation, Wiley, 2009.
[32] F.B.L. Neto, R. Menezes, M. Ávares, Hybrid and evolutionary agent-based social simulations using Lévy flight as randomization method, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), IEEE, 2010, pp. 1–8.
[33] G. Gutin, A. Punnen, A. Barvinok, Kh.G. Edward, I.S. Anatoliy, The Traveling Salesman Problem and Its Variations. Combinatorial Optimization, Kluwer Academic, 2002.
[34] G. Pampara, A.P. Engelbrecht, Binary artificial bee colony optimization, in: Proceedings of the IEEE Symposium on Swarm Intelligence (SIS), IEEE, 2011, pp. 1–8.
[35] G. Pampara, A.P. Engelbrecht, N. Franken, Binary differential evolution, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC'06), IEEE, 2006, pp. 1873–1879.
[36] G. Pampara, N. Franken, A.P. Engelbrecht, Combining particle swarm optimisation with angle modulation to solve binary problems, in: Proceedings of the Congress on Evolutionary Computation (CEC'05), IEEE, 2005, pp. 89–96.
[37] G.M. Platt, Computational experiments with flower pollination algorithm in the calculation of double retrograde dew points, Int. Rev. Chem. Eng. 6 (2) (2014).
[38] G.M. Platt, I.N. Bastos, R.P. Domingos, Calculation of double retrograde vaporization: Newton's methods and hyperheuristic approach, J. Nonlinear Syst. Appl. 4 (2) (2012) 107–120.
[39] H. Liu, H. Motoda, Computational Methods of Feature Selection. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC, Boca Raton, 2008.
[40] I. El henawy, M. Ismail, An improved chaotic flower pollination algorithm for solving large integer programming problems, Int. J. Digit. Content Technol. Appl. 8 (3) (2014).
[41] H.M. Abdelsalam, A. Al-Shaar, An enhanced binary particle swarm optimization algorithm for channel assignment in cognitive radio networks, in: Proceedings of The International Conference on Modelling, Identification and Control (ICMIC), IEEE, 2013, pp. 221–226.
[42] I. Pavlyukevich, Lévy flights, non-local search and simulated annealing, J. Comput. Phys. 226 (2) (2007) 1830–1844.
[43] J. Beal, Superdiffusive dispersion and mixing of swarms with reactive lévy walks, in: Proceedings of the 7th International Conference on Self-Adaptive and Self-Organizing Systems (SASO '13), IEEE Computer Society, 2013, pp. 141–148.
[44] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, IEEE, 1995, pp. 1942–1948.
[45] J.A.G. Pulido, S. Priem-Mendes, M.A. Vega-Rodriguez, P.J. Cordeiro, J.M. Sanchez-Perez, Processor for measuring radio network design quality, Wirel. Eng. Technol. 2 (3) (2011) 204–211.
[46] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, University of Michigan Press, 1975.
[47] J.H. McCulloch, Numerical approximation of the symmetric stable distribution and density, in: A Practical Guide to Heavy Tails, Birkhauser Boston Inc., 1998, pp. 489–499.
[48] J.N. Ramos, M. Bugnosen, E. Reyes, M.A. Pedrasa, Implementation and comparison of modified binary particle swarm optimization approaches to an interruptible load dispatch problem, in: TENCON – IEEE Region 10 Conference, IEEE, 2012, pp. 1–6.
[49] E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Approximation Algorithms for bin Packing: A Survey, PWS Publishing & Co., 1997, pp. 46–93.
[50] K. Chakraborty Uday, Computational Intelligence in Flow Shop and Job Shop Scheduling, 1st ed., Springer, 2009.
[51] K. Lenin, R.B. Ravindhranath, Flower pollination algorithm for solving optimal reactive power dispatch problem, Int. J. Rec. Res. Interdiscip. Sci. 1 (2014) 7–16.
[52] K. Lenin, R.B. Ravindhranath, K.M. Surya, Shrinkage of active power loss by hybridization of flower pollination algorithm with chaotic harmony search algorithm, Control Theory Inform. (8) (2014) 7–16.
[53] J. kennedy, R.C. Eberhart, A discrete binary version of the particle swarm optimization algorithm, in: Proceedings of the International Conference on Systems, Man and Cybernetics, IEEE, 1997, pp. 41404–44108.
[54] M. Abdel-Basset, M. Attia, Hybrid flower pollination algorithm for solving constrained optimisation problem, Adv. Eng. Technol. Appl. 4 (2) (2014).
[55] M. Hollander, Douglas A. Wolfe, E. Chicken, Nonparametric Statistical Methods, Wiley, 1999.
[56] M. Mohammadi, F. Nassiri, A. Malek, Optimization Models for Solving Lot-Sizing Problems: A Study in Mixed Integer Programming with Feasible Solutions, Lap Lambert Academic Publishing, 2012.
[57] M. Pambudy, M. Musofa, P.H. Sasongko, R.A. Husni, Flower pollination algorithm for optimal control in multi-machine system with GUPFC, in: Proceedings of the 6th International Conference on Information Technology and Electrical Engineering (ICITEE), IEEE, 2014, pp. 1–6.
[58] M. Rahoual, P. Siarry, Reséaux informatiques: conception et optimization, Editions Technip, 2006.
[59] M. Sharawi, E. Emary, I.A. Saroit, H. El-Mahdy, Flower pollination optimization algorithm for wireless sensor network lifetime global optimization, Int. J. Soft Comput. Eng. 4 (2014).
[60] M. Vasquez, Jin-Kao Hao, A heuristic approach for antenna positioning in cellular networks, J. Heuristics 7 (5) (2001) 443–472.
[61] M.A. Vega-Rodriguez, J.A.G. Pulido, E. Alba, D. Vega-Perez, S. Priem-Mendes, G. Molina, Evaluation of different metaheuristics solving the RND problem, in: Proceedings of the EvoWorkshops on Applications of Evolutionary Computing: EvoCoMnet, EvoFIN, EvoIASP, EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog, Springer, 2007, pp. 101–110.
[62] M.A. Vega-Rodriguez, J.A.G. Pulido, E. Alba, D. Vega-Perez, S. Priem-Mendes, G. Molina, Using omnidirectional BTS and different evolutionary approaches to solve the RND problem, in: Proceedings of the 11th International Conference on Computer Aided Systems Theory (Eurocast), Springer, 2007, pp. 853–860.
[63] M.F.P. Costa, Ana Maria A.C. Rocha, Rogèrio B. Francisco, Edite M.G.P. Fernandes, Heuristic-based firefly algorithm for bound constrained nonlinear binary optimization, Adv. Oper. Res. 1 (215182) (2014) 12.
[64] N. Sakib, W. Ul Kabir, S. Rahman, S. Alam, A comparative study of flower pollination algorithm and bat algorithm on continuous optimization problems, Int. J. Appl. Inf. Syst. 7 (9) (2014) 19–20.
[65] O. Abdel-Raouf, M. Abdel-Baset, I. El henawy, An improved flower pollination algorithm with chaos, Int. J. Educ. Manag. Eng. 4 (2) (2014).
[66] O. Abdel-Raouf, M. Abdel-Baset, I. El henawy, A novel hybrid flower pollination algorithm with chaotic harmony search for solving sudoku puzzles, Int. J. Eng. Technol. 7 (3) (2014).
[67] P. Calegari, F. Guidec, P. Kuonen, A parallel genetic approach to transceiver placement optimisation, in: Proceedings of the International Conference on Parallel and Distributed Systems Workshop (SIPAR '96), 1996, pp. 21–24.
[68] P. Calegari, F. Guidec, P. Kuonen, D. Kobler, Parallel island-based genetic algorithm for radio network design, J. Parallel Distrib. Comput. 47 (1) (1997) 86–90.
[69] P. Calegari, F. Guidec, P. Kuonen, D. Wagner, Genetic approach to radio network optimmization for mobile systems, in: Proceedings of the 47th Vehicular Technology Conference, IEEE, 1997, pp. 755–759.
[70] P. Calegari, F. Guidec, P. Kuonen, D. Wagner, Parallel Island-based Genetic Algorithm for Radio Network Design. Technical Report, 1997.
[71] P. Calegari, F. Guidec, P. Kuonen, F. Nielsen, Combinatorial optimization algorithms for radio network planning, Theor. Comput. Sci. 263 (12) (2001) 235–245.
[72] P. Calegari, F. Guidec, P. Kuonen, B. Chamaret, Radio network planning with combinatorial optimisation algorithms, in: Proceedings of the ACTS Mobile Communications Conference, 1996, pp. 707–713.
[73] P.R. Calegari, Parallelization of Population-based Evolutionary Algorithms for Combinatorial Optimization Problems (Ph.D. thesis), Claude Bernard University, 1999.
[74] Q. Liu, W. Lu, W. Xu, Spectrum allocation optimization for cognitive radio networks using binary firefly algorithm, in: Proceedings of the International Conference on Innovative Design and Manufacturing (ICIDM), IEEE, 2014, pp. 257–262.
[75] R. Caruana, J.D. Schaffer, Representation and hidden bias: gray vs. binary coding for genetic algorithms, in: Proceedings of the 5th International Conference on Machine Learning (ICML), 1988, pp. 153–161.
[76] R. Falcon, M. Almeida, A. Nayak, Fault identification with binary adaptive fireflies in parallel and distributed systems, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC'11), IEEE, 2011, pp. 1359–1366.

[77] R. Prathiba, M.M. Balasingh, S. Sakthivel, Flower pollination algorithm applied for different economic load dispatch problems, Int. J. Eng. Technol. 6 (2) (2014).
[78] R. Storn, K. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[79] R. Wang, Y. Zhou, Flower pollination algorithm with dimension by dimension improvement, Math. Probl. Eng. 58 (2014) 6–599.
[80] R.N. Mantegna, Fast, accurate algorithm for numerical simulation of lévy stable stochastic processes, Phys. Rev. E 49 (1994) 4677–4683.
[81] R.Y.M. Nakamura, L.A.M. Pereira, K.A.P. Costa, D. Rodrigues, J.P. Papa, X.S. Yang, BBA: a binary bat algorithm for feature selection, in: Proceedings of the Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), IEEE, 2012, pp. 291–297.
[82] S. Baluja, Population-based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Technical Report, 1994.
[83] S. Burnwal, S. Deb, Scheduling optimization of flexible manufacturing system using cuckoo search-based approach, Int. J. Adv. Manuf. Technol. 64 (5–8) (2013) 951–959.
[84] S. Das, R. Mukherjee, R. Kundu, T. Vasilakos, Multi-user detection in multi-carrier CDMA wireless broadband system using a binary adaptive differential evolution algorithm, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13), ACM, 2013, pp. 1245–1252.
[85] S. Łukasik, P.A. Kowalski, Study of flower pollination algorithm for continuous optimization, in: Proceedings of the 7th International Conference on Intelligent Systems: Mathematical Foundations, Theory, Analyses (IS '14), vol. 322, Springer, 2015, pp. 451–459.
[86] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Inc., 1990.
[87] S. Palit, S.N. Sinha, M.A. Molla, A. Khanra, A cryptanalytic attack on the knapsack cryptosystem using binary firefly algorithm, in: Proceedings of the 2nd International Conference on Computer and Communication Technology (ICCCT), IEEE, 2011, pp. 428–432.
[88] S. Priem-Mendes, G. Molina, M.A. Vega-Rodriguez, J.A.G. Pulido, Y. Saez, G. Miranda, C. Segura, E. Alba, P. Isasi, C. Leon, J.M. Sanchez-Perez, Benchmarking a wide spectrum of metaheuristic techniques for the radio network design problem, IEEE Trans. Evol. Comput. 13 (5) (2009) 1133–1150.
[89] S. Priem-Mendes, J.A. Gomez-Pulido, M.A. Vega-Rodriguez, A.M. Pereira, J.M.S. Perez, Fast wide area network design optimisation using differential evolution, in: Proceedings of the International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP), IEEE, 2007, pp. 3–10.
[90] S. Priem-Mendes, J.A. Gomez-Pulido, M.A. Vega-Rodriguez, J.M. Sanchez-Perez, Y. Saez, P. Isasi, The Radio Network Design Optimization Problem Benchmarking and State-of-the-Art Solvers Studies in Computational Intelligence, vol. 210, Springer, 2009.
[91] S. Priem-Mendes, J.A.G. Pulido, M.A. Vega-Rodriguez, M.D. Jaraiz Simon, J.M. Sanchez-Perez, A differential evolution based algorithm to optimize the radio network design problem, in: Proceedings of the 2nd International Conference on e-Science and Grid Computing (e-Science '06), IEEE, 2006, p. 119.
[92] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, 1996.
[93] T. Bäck, U. Hammel, H.P. Schwefel, Evolutionary computation: comments on the history and current state, IEEE Trans. Evol. Comput. 1 (1) (1997) 3–17.
[94] V.H. Robert, J. Ledolter, Engineering Statistics, Macmillan & Co. Ltd., 1987.
[95] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, 1992.
[96] X. Wang, P. Guo, A novel binary adaptive differential evolution algorithm for Bayesian network learning, in: Proceedings of the 8th International Conference on Natural Computation (ICNC), IEEE, 2012, pp. 608–612.
[97] X.S. Yang, Nature-Inspired Metaheuristic Algorithm, 2nd ed., Luniver Press, 2010.
[98] X.S. Yang, Flower pollination algorithm for global optimization, in: Proceedings of the 11th International Conference on Unconventional Computation and Natural Computation (UCNC '12), Springer, 2012, pp. 240–249.
[99] X.S. Yang, M. Karamanoglu, H. Xingshi, Flower pollination algorithm: a novel approach for multiobjective optimization, in: Engineering Optimization, 2013, pp. 1–16.
[100] X.S. Yang, Z. Cui, R. Xiao, A.H. Gandomi, M. Karamanoglu, Swarm Intelligence and Bio-Inspired Computation, Theory and Application, Elsevier Science Technology Books, 2013.
[101] Y. Aratsu, K. Mizuno, H. Sasaki, S. Nishihara, Experimental evaluation of artificial bee colony with greedy scouts for constraint satisfaction problems, in: Proceedings of the International Conference on Technologies and Applications of Artificial Intelligence (TAAI), IEEE, 2013, pp. 134–139.
[102] Y. Pirhayati, K. Mazlumi, An intelligent binary particle swarm optimization for unit commitment problem, in: Proceedings of the International Conference on Power and Energy (PECon), IEEE, 2010, pp. 248–252.
[103] Y. Saez, F. Zazo, P. Isasi, A study of the effects of clustering and local search on radio network design: evolutionary computation approaches, in: Proceedings of the 8th International Conference on Hybrid Intelligent Systems (HIS '08), IEEE, 2008, pp. 951–954.
[104] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed., Springer, 1996.
[105] Z.E.M. Dahi, C. Mezioud, A. Draa, Binary bat algorithm: on the efficiency of mapping functions when handling binary problems using continuous-variable-based metaheuristics, in: Proceedings of the 5th IIP TC International Conference on Computer Science and Its Applications (CIIA), Springer, 2015, pp. 3–14.