A 0-1 bat algorithm for cellular network optimisation: a systematic study on mapping techniques

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

Modelling and Implementation of Complex Systems Laboratory,
Faculty of New Technologies of Information and Communication,
Constantine 2-Abd Elhamid Mehri University,
Constantine City, Algeria
Email: zakaria.dahi@univ-constantine2.dz
Email: chaker.mezioud@univ-constantine2.dz
Email: draa_amer@yahoo.fr
*Corresponding author

Abstract: Many research efforts are deployed today in order to design techniques that allow continuous metaheuristics to also solve binary problems. However, knowing that no work thoroughly studied these techniques, such a task is still difficult since these techniques are still ambiguous and misunderstood. The bat algorithm (BA) is a continuous algorithm that has been recently adapted using one of these techniques. However, that work suffered from several shortfalls. This paper conducts a systematic study in order to investigate the efficiency and usefulness of discretising continuous metaheuristics. This is done by proposing five binary variants of the BA (BBAs) based on the principal mapping techniques existing in the literature. As problem benchmark, two optimisation problems in cellular networks, the antenna positioning problem (APP) and the reporting cell problem (RCP) are used. The proposed BBAs are evaluated using several types, sizes and complexities of data. Two of the top-ranked algorithms designed to solve the APP and the RCP, the population-based incremental learning (PBIL) and the differential evolution (DE) algorithm are taken as comparison basis. Several statistical tests are conducted as well.

Keywords: bat algorithm; binary problems; mapping techniques; antenna positioning problem; APP; reporting cell problem; RCP.

Reference to this paper should be made as follows: Dahi, Z.A.E.M., Mezioud, C. and Draa, A. (2017) 'A 0-1 bat algorithm for cellular network optimisation: a systematic study on mapping techniques', *Int. J. Reasoning-based Intelligent Systems*, Vol. 9, No. 1, pp.22–42.

Biographical notes: Zakaria Abdelmoiz Dahi received his PhD in Computer Science from the University of Constantine 2 – Abdelhamid Mehri, Constantine City, Algeria, in 2017. He is currently a researcher at the University of Constantine 2 – Abdelhamid Mehri. His current research interests involve soft computing, the design and application of metaheuristics to real problems, especially in telecommunication networks.

Chaker Mezioud received his PhD in Computer Science from the University of Mentouri 1, Constantine city, Algeria, in 2012. He is a Lecturer at the University of Constantine 2 – Abdelhamid Mehri. His research interests include geographical information systems and communication networks optimisation.

Amer Draa is a Lecturer at the Constantine 2 University, Algeria. He received his Engineer, Magister and PhD degrees in Computer Science in 2002, 2004 and 2011 respectively, from Mentouri University, Algeria. His current research interests include complex adaptive systems, metaheuristics, computer vision and nature-inspired computing.

This paper is a revised and expanded version of a paper entitled 'Binary bat algorithm: on the efficiency of mapping functions when handling binary problems using continuous-variable-based metaheuristics' presented at the 5th IFIP TC 5 International Conference, CIIA 2015, Saida, Algeria, 20–21 May 2015.

1 Introduction

Metaheuristics can be classified on the basis of their solution representation, architecture and operators into two types: continuous and binary. Continuous metaheuristics are algorithms that are initially designed to tackle only continuous problems, while binary metaheuristics are designed to only solve binary problems. This problem-dependency makes choosing the optimal algorithm that ensures an efficient solving, a hard task even for specialists. In fact, this requires advanced knowledge on both the problem addressed and algorithm used. The latter limit the use of metaheuristics to experts within pure abstract scientific research while there is a constant demand for efficient algorithms usable by novice users within real-life environments.

A promising solution to this issue is to supply the industry with continuous algorithms that also can handle binary problems. Many discretisation approaches were proposed in the literature, but they can be all grouped in two subcategories: modifying the algorithm's operators and using mapping techniques. The latter is the one studied in this work.

Given the above considerations, one of today's hot research topics is the design of mapping techniques that allow continuous metaheuristics to work on binary problems as efficiently as they do on continuous ones. Many research efforts have been deployed in this direction and they have showed very encouraging results. Kennedy and Eberhart (1997) proposed a binary version of the particle swarm optimisation algorithm. Rashedi et al. (2010) devised a binary version of the gravitational search algorithm. A binary harmony search algorithm was also proposed in Ramos et al. (2011). Recently, Pampara and Engelbrecht (2011) and Rodrigues et al. (2013) proposed binary versions of the cuckoo search and the artificial bee colony algorithm, respectively.

However, the use of such techniques has some drawbacks like adding more nonlinearities and mathematical difficulties either to the problem or the algorithm (Bäck et al., 1997) which can alter the algorithm's efficiency as well as sensitivity. These facts make creating efficient mapping techniques a complicated task. This becomes even harder as far as these techniques are ambiguous and misunderstood. In fact, no work thoroughly studied their efficiency, behaviour and controlling factors. Thus, many questions still surround them such as what are their drawbacks and advantages? How do they behave? What are their controlling factors? Are they efficient? Does designing efficient algorithms imply using more complex mapping techniques? Ultimately, how to design better mapping techniques?

The bat algorithm (BA) is a recently proposed continuous metaheuristic that has been adapted using mapping techniques in order to solve binary problems (Mirjalili et al., 2013). However, that work was a bit simplistic and suffers from shortfalls in both foundations and experimentations, which makes the assessment of the binary BA not reliable. In fact, many questions still surround this algorithm as well. Is it really efficient? How does it behave? How is its efficiency when compared against top-ranked algorithms in the literature?

In addition, it is worth mentioning that a similar work to the one presented in this paper has been conducted for the analysis of the mapping techniques and the flower pollination algorithm (FPA) (Dahi et al., 2016b). However, even if that work was limited by the use of only four mapping techniques and one validation benchmark, it opened the door for another question: *Could the results drawn from the FPA be generalised to other metaheuristics*?

Under the light of these facts, the aim of this paper is to conduct a systematic study on the mapping techniques in order to investigate some of the above-mentioned questions and ultimately find if it is worth or not discretising continuous metaheuristics to solve binary problems. This is done by proposing five binary variants of the bat algorithm (BBAs) based on the principal mapping techniques existing in the literature. The investigated techniques are the *nearest-integer technique*, the *normalisation* technique, the *angle modulation*, the *sigmoid function* and finally, the *great-value priority* technique.

As a benchmark problem, two optimisation problems in cellular networks (2G, GPRS, EDGE, 3G, 3G$^+$ and 4G), the antenna positioning problem (APP) and the reporting cell problem (RCP), are used. In order to assess the efficiency, scalability and robustness of the proposed BBAs, different types of problem instances (e.g., random, synthetic and realistic) with different sizes are used. Two of the top-ranked state-of-the-art algorithms designed to solve the APP and RCP, the population-based incremental learning (PBIL) and the differential evolution (DE) algorithm, are used as comparison basis. Several statistical tests are carried as well.

The remainder of the paper is structured as follows. In Section 2, we present basic concepts of the BA and the discretisation approach used. In Section 3, we present the used benchmark problems: the antenna positioning and RCPs. In Section 4, we introduce the devised BBAs. Section 5 is dedicated to the experimental study. Finally, in Section 6, we conclude the paper.

2 Preliminaries

In this section, we introduce basic concepts related to the BA and the studied discretisation approaches.

2.1 The BA

In this section, we firstly present the classical continuous BA. Then, we perform a literature review and a critical analysis of previous BA-based works.

2.1.1 Biological inspiration and principles

Two main types of bats exist: megabats and microbats. They differ in several aspects such as their size, claws, ears,

underfur, diet and especially their vision. In fact, unlike megabats, microbats do not have well-developed visual acuity. So, they use a type of sonar called 'echolocation' in order to move, detect their preys, avoid obstacles and locate their roosting crevices in the dark (Yang, 2010).

Echolocation consists of emitting a sound pulse and listening to the echo that bounces back from any obstacle that it encounters. The bandwidth frequency, pulses and harmonics of the echolocation vary in properties depending on the species and can be adjusted to the hunting strategies. The properties of the bouncing wave (e.g., delay between emission and reception, distance between the bat's ears, loudness of the echo, etc.) are used to extract information that is necessary for the current task, like constructing a three dimensional scenario of their actual environment, detecting the distance and orientation, the type and the speed of the prey or the obstacle (Yang, 2010).

This echolocation phenomenon has been mathematically translated into the recently proposed BA (Yang, 2010). The latter consists of the cyclic application of the initialisation, global search, local search and the update phases. Pseudo-code of Algorithm 1 describes the general framework of the BA.

Algorithm 1 The BA

```
Perform initialisation.
while (Termination criterion not reached) do
    while (Individuals not all processed) do
        Perform global search.
        if (Local search condition fulfilled) then
            Perform local search.
        end if
        Perform evaluation.
        if (Update condition fulfilled) then
            Perform update.
        end if
    end while
end while
```

In the following sections, we describe each one of the BA phases. For more details, one can refer to Yang's (2010) original work.

2.1.1.1 Initialisation

Considering a problem with D-dimensions, the BA starts by initialising a population of N individuals, where each individual, $\vec{X}_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$, represents a solution to the problem being solved, and where $i = 1, \ldots, N$ and $j = 1, \ldots, D$. Each element x_{ij} of each individual $\vec{X}_i$, represents a variable to be optimised. The latter is initialised according to a given strategy (e.g., random or heuristic) from $[L_{Bj}, U_{Bj}]$, where L_{Bj} and U_{Bj} are the lower and upper bounds of the interval where the j^{th} element is supposed to evolve within.

In addition, for each individual in the population, an initial frequency f_i and velocity $\vec{V}_i = \{v_{i1}, v_{i2}, \ldots, v_{iD}\}$ are generated, where $\vec{V}_i$ is initialised with a null D-dimensional vector ('Matlab code of the bat algorithm', https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm--demo-), while f_i is initialised with a value from $[f_{\min}, f_{\max}]$, where $f_{\min}$ and $f_{\max}$ are the lower and upper bounds of the interval where the frequency is allowed to evolve.

For each individual $\vec{X}_i$ in the population an initial loudness A_i and rate r_i are also generated randomly, where $A_i \in [A_{\min}, A_{\max}]$ and $r_i \in [r_{\min}, r_{\max}]$. The values of $A_{\min}$ and $A_{\max}$ are the lower and upper bounds of the interval where the loudness is supposed to evolve, while $r_{\min}$ and $r_{\max}$ are the upper and lower limits of the interval where the rate is supposed to evolve.

Once the population initialised, the quality of each individual is assessed by evaluating the latter using the objective function of the problem being addressed. Then, all individuals are ranked on the basis of their fitness values and the best individual $\vec{G}$ is extracted.

The classical bat iteration consists of applying sequentially on each individual in the population the following phases: global search, local search and the update (Yang, 2010).

2.1.1.2 Global search

At this step, for each individual $\vec{X}_i^t$ in the actual iteration t, a new individual $\vec{X}_i^{t+1}$ is created using formulas (1)–(3), where f_i^{t+1} and $\vec{V}_i^{t+1}$ define respectively the frequency and the velocity associated with the i^{th} individual at iteration $t + 1$ and β is randomly drawn from the interval [0, 1] according to a uniform distribution. The new solution $\vec{X}_i^{t+1}$ and its associated velocity $\vec{V}_i^{t+1}$ at iteration t are computed using formulas (2) and (3), respectively (Yang, 2010).

$$f_i^{t+1} = f_{\min} + \left(f_{\max} - f_{\min}\right)\beta \tag{1}$$

$$\vec{V}_i^{t+1} = \vec{V}_i^t + \left(\vec{X}_i^t - \vec{G}\right) f_i^{t+1} \tag{2}$$

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \vec{V}_i^{t+1} \tag{3}$$

However, it is to be noted that the formula (1) updating the frequency in Yang (2010) is not as the one implemented in 'Matlab code of the bat algorithm' (https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm--demo-). For consistency we use the formula published in Yang's (2010) original work.

2.1.1.3 Local search

After the global search phase, according to a given condition, the local search phase might (or not) be applied on the individual $\vec{X}_i$. In fact, a random number R is generated from a standard uniform distribution.

Then, if $R > r_i$, the local search is performed, otherwise not. When a local search is performed, a new individual $\vec{X}_i^{t+1}$ is generated using a random walk as described in formula (4), where $\epsilon \in [-1, 1]$ and A^t is the average loudness of all individuals at iteration t (Yang, 2010).

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \epsilon A^t \tag{4}$$

However, it is to be noted that in the original work of Yang (2010), the local search phase is performed by selecting a solution among the best ones in order to generate a new solution using formula (4), but no explanation was given on how this selection is performed. On the other hand, the BA's implementation given by Yang in 'Matlab code of the bat algorithm' (https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm--demo-) always performs the local search by selecting the best individual in the population. Thus, for consistency we perform the local search always on the selected individual itself. In addition, in the original work of Yang (2010), another step exists between the local search and the update phase. However, in both Yang's (2010) original work and implementation ('Matlab code of the bat algorithm', https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm--demo-), no reference or explanation was made to that phase.

2.1.1.4 Evaluation and update

In the evaluation step, the quality of the newly-generated individual $\vec{X}_i^{t+1}$ is assessed using the objective function of the problem being addressed, while within the update phase a decision is made about replacing the individual $\vec{X}_i^t$ by the newly-produced one $\vec{X}_i^{t+1}$. A random number R is generated from a standard uniform distribution. Then, if $R < A_i$ and $f(\vec{X}_i^{t+1}) < f(\vec{X}_i^t)$ (for minimisation problems) or $f(\vec{X}_i^{t+1}) > f(\vec{X}_i^t)$ (for maximisation problems), the i^{th} individual is accepted, otherwise it is discarded. If the new individual is updated, its loudness A_i and rate r_i are updated as well. This is done using formulas (5) and (6), where α and γ are constants and $r_i^{t=0}$ is the initial rate of the i^{th} individual (i.e., generated in the initialisation phase). The variables α and γ have to verify the conditions $0 < \alpha < 1$ and $0 < \gamma$, respectively (Yang, 2010).

$$A_i^{t+1} = \alpha A_i^t \tag{5}$$

$$r_i^{t+1} = r_i^{t=0}\left(1 - e^{(-\gamma t)}\right) \tag{6}$$

However, it is worth mentioning that Yang (2010) within his original work specified that the replacement condition to be fulfilled is $R < A_i$ and $f(\vec{X}_i) < f(\vec{G})$. But, in his BA's implementation ('Matlab code of the bat algorithm', https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm--demo-) the condition used is $f(\vec{X}_i^{t+1})$ < or > $f(\vec{X}_i^t)$. Thus, for consistency and coherence, we use the replacement condition implemented in 'Matlab code of the bat algorithm' (https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm--demo-).

Once the update phase achieved, the individuals are ranked on the basis of their fitness values. Then, the best individual $\vec{G}$ is updated. A classical iteration of the BA is finished once all the individuals in the population are processed. Then, a new iteration is reiterated until a given stop criterion is reached.

It is worth mentioning that Yang (2010) in his work set the values of $\alpha = \gamma = 0.9$, $f_{\min} = 0$, $f_{\max} = 100$, $A_{\min} = 1$, $A_{\max} = 2$, $r_{\min} = 0$ and $r_{\max} = 1$. However, the author specified that the values of these variables and other such as the size of the population N can be tuned in compliance with the addressed problem (Yang, 2010).

2.1.2 Literature review and critical analysis

As mentioned above, the BA is a continuous metaheuristic that was first proposed by Yang in 2010. Then, it was used to solve several academic and real-life continuous problems (Gandomi et al., 2012; Tsai et al., 2012; Yang, 2011, 2012; Yang and Gandomi, 2012; Yang and He, 2013).

Recently, the BA has been used to solve numerous continuous problems in medicine and bioinformatics (Kora and Kalva, 2015; San Chua et al., 2015), communication (Nguyen et al., 2015; Thafasal Ijyas and Sameer, 2015), signal processing (Al-Muraeb and Abdel-Aty-Zohdy, 2016; Severino et al., 2015) and in many other fields (Gao et al., 2016; Sambariya et al., 2016). For more details, one can refer to the BA's literature review conducted in Chawla and Duhan (2015) and Yang and He (2013).

The BA has also been adapted to address binary optimisation problems. Two main binary variants of it have been proposed in the literature. The first one was presented by Nakamura et al. in 2012, while the second was recently devised by Mirjalili et al. (2013). The only difference between both variants is that the first one uses the sigmoid function as mapping technique and the second uses a v-shaped transformation function. Both variants have been used for solving a wide range of binary optimisation problems such as feature selection (Enache and Sgarciu, 2015; Laamari and Kamel, 2014), multi-objective problems (Laamari and Kamel, 2015), graph problems (Crawford et al., 2015; Djelloul et al., 2014), security (Enache et al., 2015), antenna placement (Dahi et al., 2015) and many other binary problems in several fields (Jaddi et al., 2015; Kang et al., 2015).

However, many of these works suffered from some shortfalls. Taking the recent work of Mirjalili et al. (2013) as an example, one can notice that it has shortcomings at two main levels: foundations and experimental validation. When analysing that work with respect to the foundation, one can find that the authors proposed a binary BA using a mapping technique, while other discretisation approaches exist in the literature. They proposed a v-shaped mapping function and this by theoretically analysing the drawbacks

of the classical sigmoid function. However, the authors analysed only the sigmoid function, while other mapping techniques exist in the literature, which might not have the sigmoid function's drawbacks. This fact makes the use of the proposed v-shaped function questionable. In addition, no numerical comparison was made between the sigmoid function or other state-of-the-art functions and the newly-proposed v-shaped function in order to affirm such theoretical deduction. Besides, no literature review or critical analysis of state-of-the-art mapping techniques was done in order to justify each one of the above-mentioned choices.

Taking now the experimental validation side, one can notice that the authors assessed the performances of the proposed BBA on both a set of continuous benchmark functions and the photonic crystal design problem. However, both do not represent a wide range of complexities and properties. In fact, more challenging sets of benchmark exist such as the CEC testbed and other real-world optimisation problems. In addition, the used benchmark functions are continuous and not binary ones. Indeed, the authors used a binary coding to represent the continuous parameters to be optimised, while they could apply directly the continuous BA without any discretisation. Actually, mapping functions are used for converting continuous-coded solutions into binary ones for solving binary problems and not the opposite. This fact makes the use of mapping functions by the authors not well-motivated. Besides, the use of the BBA to solve continuous problems does not allow understanding the problem properties and the algorithm behaviour as well.

Again, for the set of benchmark functions, the proposed BBA was compared against the basic genetic algorithm (GA) and binary particle swarm optimisation (BPSO) algorithm, while other top-ranked state-of-the-art algorithms exist for solving these problems in particular and binary problems in general. In addition, the GA and BPSO taken as comparison basis were not used with their optimal configurations and no fine-grained tuning of their parameters was performed neither the used parameters were justified. All of these facts do not allow performing a reliable assessment of the efficiency of the proposed BBA. Furthermore, only one statistical test was used, while many other powerful ones exist and are even necessary. The choice of that test was not justified. Some shortfalls can also be noticed in the interpretation of the statistical test performed within that work. This makes the statistical study not fully reliable. The proposed BBA did not also show promising results when compared to the GA and BPSO. Indeed, it could only outperform the latter in 12 out of 22 functions.

Considering the photonic design problem, experiments were performed only on one instance, which does not allow representing a wide range of problem complexity. Besides, the authors did not compare the proposed BBA to any other algorithm. The experiments were performed only for 10 executions and 1,000 iterations, which does not allow performing reliable statistical and numerical assessments of the BBA. Several numerical metrics such as the best solution achieved, the mean and standard deviation of the fitness values over the whole set of runs were neglected. Moreover, no statistical tests were performed in order to assess the efficiency of the proposed BBA for solving the photonic design problem. For both the benchmark functions and the photonic design problem, the authors did not use a wide range of data types (e.g., realistic, synthetic, etc.), which does not allow performing a reliable assessment of the robustness of the proposed BBA against different types of data. Furthermore, both problems are minimisation problems. Thus, no real assessment of the BBA sensitivity against different types of problem (e.g., minimisation and maximisation) can be made. Moreover, instances of problems (of both the benchmark functions and the photonic crystal design problem) are small-sized (75 and 25 bits) and the used benchmark functions have all the same size. Thus, a reliable study on the scalability of the proposed variant cannot be conducted.

The other works in the literature have used the same variants that were proposed by Mirjalili et al. (2013). Thus, each one of the above-mentioned shortfalls can be mostly generalised to these works.

2.2 Discretising continuous metaheuristics

In this section, we introduce basic concepts of the discretisation approaches used to discretise continuous algorithms in general and more particularly those used in our work to discretise the BA.

2.2.1 Literature review and critical analysis

The adaptation of a continuous metaheuristic to deal with binary problems basically consists of adapting both the solution representation and dynamics of the algorithm (operators). The solution representation can be adapted by simply using a binary or grey-coded representation. So, the whole difficulty lies in adapting the algorithm's dynamics in order to become able to work in a binary space as efficiently as it works in a continuous one.

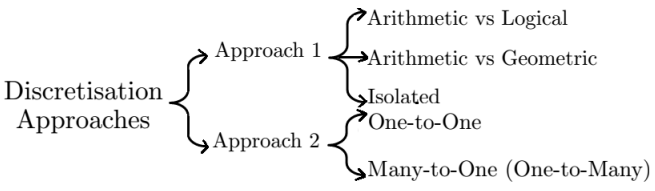
Two main approaches of operators adaptation exist in the literature (see Figure 1). The first directly acts on the operators of the algorithm in order to make them able to work on binary solutions. This approach can be further subdivided into three other subcategories. The first one consists of substituting the arithmetic operations in the algorithm by logical ones (Martinez Cruz et al., 2013; Wang and Guo, 2012), while the second subcategory consists of finding the corresponding geometric interpretation of the algorithm's operators in Hamming space (Moraglio and Silva, 2010; Moraglio et al., 2013). The third subcategory uses some isolated strategies that have been proposed in the literature, but none of them followed one of the above-mentioned subcategories (Aratsu et al., 2013; Deng et al., 2013).

Nevertheless, this first approach suffers from some shortcomings like being limited to some specific metaheuristics such the DE algorithm (Martinez Cruz et al.,

2013) and PSO (Abdelsalam and Al-shaar, 2013), whose dynamics is ruled by a specific type of formulas. To the best knowledge of the authors, this approach was never applied for discretising other complex metaheuristics such as the BA, the cuckoo search algorithm (Rodrigues et al., 2013), the harmony search algorithm (Ramos et al., 2011), gravitational search algorithm (Rashedi et al., 2010) and the artificial bee colony (Pampara and Engelbrecht, 2011)]. In addition, so far, it can translate only operations of addition, multiplication and subtraction into logical ones or find their geometrical interpretation, while other mathematical concepts cannot be discretised using this approach.

Moreover, this approach tends to totally change the original algorithm (i.e., architecture, operators, etc.), which makes its discrete version lose all links with the original algorithm. Actually, the resulting discrete algorithm is technically a simple or a complex variant of the GA. This is due to the fact that the discretisation of the algorithm's operators turns them into simple or complex versions of mutation and crossover. We can cite for instance, the binary DE algorithm (Martinez Cruz et al., 2013; Wang and Guo, 2012). When discretising the latter by substituting the arithmetic operators in its mutation formula, it becomes a simple bit-flip mutation. Thus, the binary DE uses its original crossover and the new bit-flip mutation introduced by the discretisation process. The binary version of the PSO, known as the geometrical particle swarm optimisation (GPSO) algorithm, is another example of this shortfall (Alba et al., 2008; Ghaleb and Alharkan, 2015). In fact, when substituting the operators of the continuous PSO by their corresponding versions in the geometric space, the updating formula of the original PSO becomes a variant of the crossover called three parent-mask-based crossover (3PMBCX). Besides the 3PMBCX, the produced GPSO uses a mutation as well. A third example can be given by another binary variant of the PSO (Abdelsalam and Al-shaar, 2013; Rafiei and Zadeh, 2012), where arithmetic operators of the position update formula are substituted by logical ones. Thus, the updating formula becomes a simple bit-flip mutation and makes the discrete version of the PSO a simple evolutionary algorithm using mutation as a main operator. Consequently, one can say that the first approach of discretisation does not bring the essence of adaptation sought in this work. In fact, it consists of proposing variants of an already existing algorithm, the GA, rather than discretising a continuous algorithm.

Figure 1 Discretisation approaches for metaheuristics



In the second category of approaches, the original operators and structure of the algorithm are kept, but a mapping mechanism is inserted in order to convert the real-valued solutions produced by the algorithm into binary ones. This category 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 continuous search space and *B* is the number of binary solutions that could be represented in the binary search space and vice versa.

However, a complex mapping function may introduce additional nonlinearities and other mathematical difficulties, which may substantially hinder the search process and affect the algorithm's efficiency (Bäck,, 1996). Thus, an efficient design of a mapping technique is a tricky task that 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), avoiding the loss of precision and *Hamming Cliffs effects* (Caruana and Schaffer, 1988), and many other constraints (Rashedi et al., 2010) that need to be considered when mapping continuous solutions to binary ones.

On the other hand, unlike the first category, this one can be applied to all metaheuristics. In addition, it allows conserving the original framework of the algorithm (i.e., operators, architecture, etc.). Finally, according to Krause et al. (2013), this category is statistically the most used in literature. On the basis of this analysis, the second category is the one studied in our work.

2.2.2 The studied mapping techniques

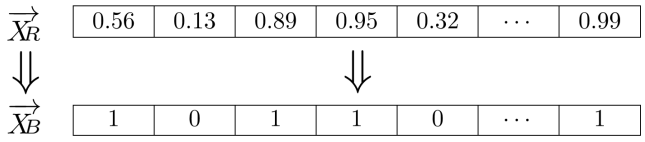
In this section we present the mapping techniques used to discretise the BA. For scientific relevance, we decided to investigate the most used techniques in the literature, and whose efficiency is well-established. In addition, these techniques are chosen to illustrate a wide range of mapping schemes: one-to-one and many-to-one (one-to-many). The mapping techniques studied in this work are the nearest-integer method (Kimiyaghalam et al., 2013), normalisation (Pampara and Engelbrecht, 2011), the angle modulation technique (Keles, 20047), the sigmoid function (Liu et al., 2014) and the great-value priority technique (Congying et al., 2011).

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

2.2.2.1 Nearest-integer

This method has been used in many works (Jiang et al., 2013; Kimiyaghalam et al., 2013). 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 Figure 2.

Figure 2 The nearest-integer mapping technique



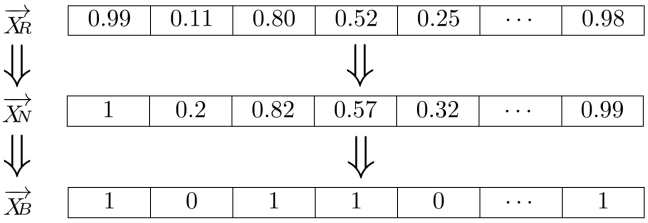
2.2.2.2 Normalisation

Some works have been conducted on this technique (Pampara and Engelbrecht, 2011; Pampara et al., 2006). 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 (7). Then secondly, the condition represented by formula (8) is applied on the normalised vector $\vec{X}_N$ in order to produce $\vec{X}_B$.

$$X_{N_j} = \frac{\left(X_{R_j} + X_{R_{\min}}\right)}{\left(\left|X_{R_{\min}}\right| + X_{R_{\max}}\right)} \tag{7}$$

$$X_{B_j} = \begin{cases} 1, & \text{If } X_{N_j} \geqslant 0.5 \\ 0, & \text{Otherwise} \end{cases} \tag{8}$$

Figure 3 The normalisation mapping technique



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 j^{th} elements of the solution vectors $\vec{X}_R$, $\vec{X}_B$ and $\vec{X}_N$, respectively, where j = 1, …, D. Figure 3 is an example of how this technique works.

2.2.2.3 Angle modulation

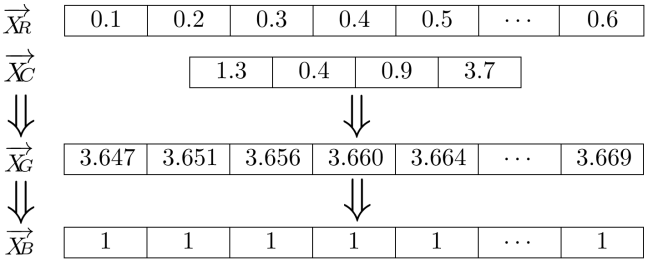
Many works were based on this method (Keles, 2007; Pampara and Engelbrecht, 2011; Pampara et al., 2006). The mapping within this technique is performed over two steps. Firstly, a vector $\vec{X} = \{X_{G_1}, X_{G_2}, X_{G_3}, ..., X_{G_D}\}$ is generated by using a trigonometric function. This function is defined by formula (9), 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 $\vec{X}_C$ 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 (10) is applied on the solution vector $\vec{X}_G$ in order to produce $\vec{X}_B$.

$$X_{G_j} = \sin\left(\begin{array}{c} 2\Pi\left(X_{R_j} - X_{C_1}\right) * X_{C_2} \\ * \cos\left(2\Pi\left(X_{R_j} - X_{C_1}\right) * X_{C_3}\right) \end{array}\right) + X_{C_4} \tag{9}$$

$$X_{B_j} = \begin{cases} 1, & \text{If } X_{G_j} \geqslant 0 \\ 0, & \text{Otherwise} \end{cases} \tag{10}$$

The particularity of this mapping technique is that instead of optimising a D-dimensional solution vector $\vec{X}_R$, the algorithm will optimise a four-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}. Figure 4 gives an example of how this method works.

Figure 4 The angle modulation mapping technique



2.2.2.4 Sigmoid function

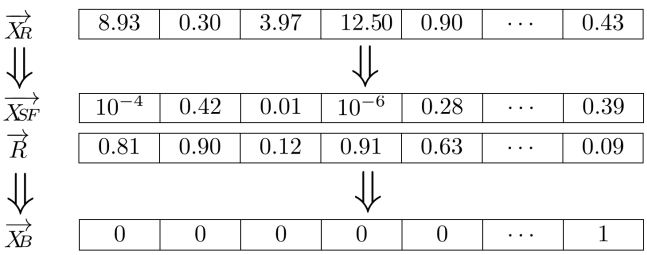
Several works have used this technique (Kennedy and Eberhart, 1997; Liu et al., 2014; Nakamura et al., 2012). The mapping within this technique is performed over two steps. Firstly, for each element X_R of the solution vector $\vec{X}_R$, a probability X_{SF} is calculated using the sigmoid function defined by formula (11). Then, using the probability vector $\overrightarrow{X_{SF}}$, each element in the solution vector $\vec{X}_B$ will either have the value 1 or 0 by applying the condition represented by formula (12).

$$X_{SF_j} = \frac{1}{1 + e^{XR_j}} \tag{11}$$

$$X_{B_j} = \begin{cases} 1, & \text{If } R_j < X_{SF_j} \\ 0, & \text{Otherwise} \end{cases} \tag{12}$$

Assuming that $\vec{R}$ = {R_1, R_2, R_3, …, R_D} is a vector of randomly-generated positive numbers drawn from a standard uniform distribution. Figure 5 is an illustration of the way this technique performs.

Figure 5 The sigmoid function mapping technique



2.2.2.5 Great-value priority

Recently, the authors in Congying et al. (2011) introduced this technique. The mapping principles of this method consists of creating a permutation vector $\vec{P} = \{p_1, p_2, \ldots, p_D\}$ from a real-valued one $\vec{X}_R$. The first element of the permutation vector p_1 will have the position of the largest element in $\vec{X}_R$, p_2 will have the position of the second largest element of $\vec{X}_R$, and so on. The procedure is reiterated until all the elements of $\vec{X}_R$ are browsed. Finally, having the permutation vector $\vec{P}$ and using formula (13), the binary-coded solution $\vec{X}_B$ is created. Figure 6 gives an example of how this technique performs.

$$X_{R_j} = \begin{cases} 1, & \text{If } p_j > p_{j+1} \\ 0, & \text{Otherwise} \end{cases} \quad (13)$$

Figure 6 The great-value priority mapping technique

$\vec{X}_R$	1.99	0.2	0.03	3.52	2.25	0.98
$\vec{P}$	4	5	1	6	2	3
$\vec{X}_B$	0	1	0	1	0	0

3 Benchmark problems

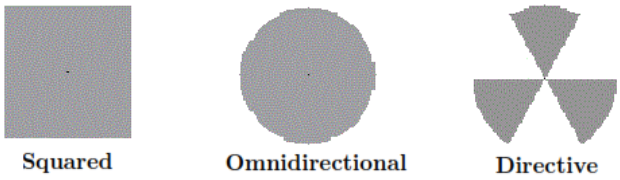
Cellular networks 2G, 3G and 4G are probably one of today's most popular means of communication. This is due to the big popularity and accessibility of the services they propose. These networks are subject to big technological and financial challenges that made the service quality a critical factor. The latter is strongly bounded to the design quality of the network itself. Thus, any dysfunction of this process might decrease the quality of the service provided. Therefore, these issues are ones of today's most important challenges that mobile operators have to confront. We can cite, for instance, the APP and the RCP. The first one consists in the network coverage optimisation, while the second is about the mobile user tracking process.

In this section, we introduce the problem modelling, solution representation and objective function of both problems.

3.1 Antenna positioning problem

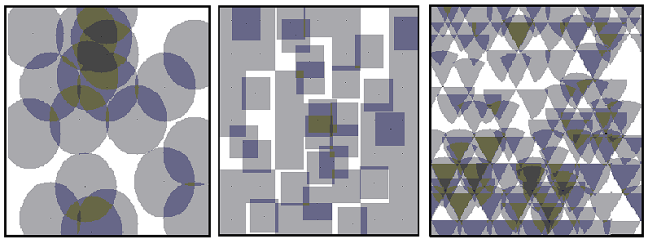
The APP formulation studied in this work was given by Calegari et al. (1996). In the latter, the APP was formulated as a binary optimisation problem and proven to be NP-hard to solve (Calegari, 1999; Priem-Mendes et al., 2009a, 2009b). This formulation considers two practical objectives: maximising the covered area, while minimising the number of used base transceiver stations (BTSs). The latter are radio transmitting devices with a specific type of coverage (see Figure 7). In this work, three types of coverage are used: squared, omnidirectional and directive (Alba et al., 2006). A cell is a part of a geographical area that is covered by a BTS.

Figure 7 Antenna coverage models



The working area to be covered is modelled as a rectangular discrete grid with H_d and V_d dimensions (see Figure 8), in which $\{site_1, site_2, \ldots, site_D\}$ is the set of potentially preselected sites where the antennas can be placed. Each potential site location is identified by its Cartesian coordinates $\{site_1 = (x_1, y_1), site_2 = (x_2, y_2), \ldots, site_D = (x_D, y_D)\}$. Figures 8(a)–8(c) illustrate a working area using the omnidirectional, squared and directive antennas, respectively.

Figure 8 Representation of the discretised area (see online version for colours)



(a) (b) (c)

3.1.1 Solution encoding

A potential solution of the APP can be represented as a vector of integers described as follows. Each vector $\vec{X} = \{x_1, x_2, \ldots, x_D\}$, represents a potential configuration of the mobile network. The size, D, of each vector represents the number of available candidate sites where antennas can be placed. The search space complexity of the APP when using this solution encoding, was assessed to be 2^D.

Each element x_j from $\vec{X}$ represents the state (i.e., placed or not) of the j^{th} antenna in the network, where $j = 1, \ldots, D$. If $x_j = 1$, the j^{th} base station is selected, otherwise it is discarded. Figure 9 is an illustration of this representation.

Figure 9 A typical solution representation

$\vec{X}$	x_1	x_2	x_3	x_4	x_5	$\cdots$	x_D
	1*st* BTS	2*nd* BTS	3*rd* BTS	4*th* BTS	5*th* BTS		*D*th BTS

The vector in Figure 10 is an example of a potential instance of the APP solution. It represents a configuration for a network with eight available locations, where the 1st, 4th and the 8th BTSs are deployed, while the remaining BTSs are discarded.

Figure 10 Example of the APP solution representation

$\vec{X}$	1	0	0	1	0	0	0	1
	1st BTS	2nd BTS	3rd BTS	4th BTS	5th BTS	6th BTS	7th BTS	8th BTS

3.1.2 Objective function

The APP objective function to be optimised is defined by formula (14).

$$\text{Maximise}: f_1(\vec{X}) = \frac{C_R^{\alpha}}{N_A(\vec{X})} \tag{14}$$

with:

$$C_R = \frac{C_A}{T_A} * 100 \tag{15}$$

$$C_A = \sum_{i=1}^{V_d} \sum_{j=1}^{H_d} C_{S_{ij}},\ C_{S_{ij}} \in \{0, 1\} \tag{16}$$

$$T_A = H_d * V_d \tag{17}$$

$$N_A(\vec{X}) = \sum_{k=1}^{D} x_k,\ x_k \in \{0, 1\} \tag{18}$$

The APP objective function, f_1, is applied to assess the quality of a potential solution vector $\vec{X} = \{x_1, x_2, \ldots, x_D\}$, where C_R represents the ratio between the total surface of the working area T_A and the actually-covered one C_A. The variables H_d and V_d are the horizontal and vertical dimensions of the discretised grid that represents the working area. The variable C_A represents the covered area, which is the amount of cells that have been covered by at least one antenna. The variable $C_{S_{ij}}$ represents the state (covered and uncovered) of the $(i, j)^{\text{th}}$ cell from the discretised grid, where $i = 1, \ldots, V_d$ and $j = 1, \ldots, H_d$. T_A is the total surface of the working area, which corresponds to the total number of cells (either covered or uncovered) in the discretised grid. The variable NA represents the number of deployed antennas. The variable α is a value that reflects the importance given to coverage. It is worth mentioning that the variables H_d, V_d, C_A, C_S and T_A are expressed in terms of cells, while the variable N_A is expressed in terms of antennas.

The variable C_S can take one of two values; 1 if the cell is covered by at least one antenna and 0 otherwise. To determine if a cell is covered or not, the coverage range of the antenna (i.e., radius) (see Table 2) is taken into account. It is to be noted that the radius of an antenna is expressed in terms of cells. When an antenna is selected to be deployed, every cell in the perimeter of its radius as well as the cell where the antenna is placed, are considered as covered. It is worth mentioning also that a cell can be covered by one or many antennas. In this case, the C_S value of that cell is equal to 1, no matter the number of antennas that covers it.

For consistency and comparison purposes with state-of-the-art algorithms, we use the same value of α ($\alpha = 2$) as used in all works treating the APP (Priem-Mendes et al., 2009a, 2009b).

3.2 Reporting cell problem

The mobility management of users in communication networks constitutes a hot research topic in today's network optimisation process. It is the process of tracking the mobile user's activity across the network. It exists in different multi-technological wired and wireless networks (e.g., vehicular, sensor, etc.). However, in this work, we are tackling this issue within advanced cellular networks (2G, GPRS, EDGE, 3G, 3G$^+$, LTE and 4G) (Razavi, 2011), since the latter are more sensitive to this issue.

The mobility management relies on two elementary tasks which are the location update and the paging. The first task is caused by the fact that each time a user moves inside the network a cost is involved to its mobility acknowledgement. Actually, an update action of its location is performed each time it moves, and a registration of its new location has to be done as well. A second cost is generated this time by the network itself when it tries to locate the mobile user present within the network's cells. This action involves a series of send/receive transactions or the so-called paging messages. The latter are performed between the network and the mobile. It is worth mentioning that generally, the number of paging transactions is directly related to the number of incoming calls. Considering these facts, the practical aim of the mobility optimisation task is to minimise the costs generated by both location updates and paging messages.

A mobility management scheme defines the way the location update or the paging are performed. Many schemes of location update exist, one can cite, for instance, the never-update, always-update. Although, both are almost never used within real networks. Other mobility management schemes exist and are practically implemented within real networks (Razavi, 2011). We can cite, for instance, the location-area scheme and the reporting-cell scheme. The latter is the one studied in our work.

The reporting-cell scheme consists of labelling each cell of the network as a *reporting* or *non-reporting* cell. A mobile performs a location-update task each time it enters a reporting cell, while it can freely move within non-reporting ones without any acknowledgement of its new location. So, using this scheme a mobile will be only paged by its last reporting cell and its neighbouring non-reporting cells. The reporting-cell scheme was first proposed by BarNoy and Kessler (1993) and was proved to be NP-complete to solve. Then, it has been formulated later as an optimisation problem by Hac and Zhou (1997).

3.2.1 Solution encoding

A potential solution of the RCP can be represented as a vector of integers described as follows. Each vector $\vec{X} = \{x_1, x_2, \ldots, x_D\}$, represents a potential configuration of the mobile network. The size, D, of each vector represents the number of cells in the network. The search space complexity of the RCP when using this solution encoding was assessed to be 2^D.

Each element x_j from $\vec{X}$ represents the state (i.e., reporting or non-reporting) of the j^{th} cell in the network, where j = 1, …, D. If x_j = 1, the j^{th} cell is labelled as a reporting one, otherwise it is considered as non-reporting. Figure 11 is an illustration of this representation.

Figure 11 A typical solution representation

$\vec{X}$	x_1	x_2	x_3	x_4	x_5	$\cdots$	x_D
	1*st* Cell	2*nd* Cell	3*rd* Cell	4*th* Cell	5*th* Cell		*Dth* Cell

The vector in Figure 12 is an example of a potential instance of the RCP solution. It represents a configuration for a network with eight cells, where the 1st, 4th and the 8th cells are reporting ones, while the remaining cells are non-reporting ones.

Figure 12 Example of the RCP solution representation

$\vec{X}$	1	0	0	1	0	0	0	1
	1*st* Cell	2*nd* Cell	3*rd* Cell	4*th* Cell	5*th* Cell	6*th* Cell	7*th* Cell	8*th* Cell

3.2.2 Objective function

The RCP objective function to be optimised is defined using formula (19).

$$\text{Minimise}: f_2(\vec{X}) = \beta * \sum_{i \in w} Lu_i + \sum_{i=0}^{D} P_i * V_i \tag{19}$$

The RCP objective function, f_2, is used to assess the goodness of a potential solution vector $\vec{X}$, where the variable Lu_i represents the number of location updates performed in the i^{th} reporting cell. The variable w is the set of reporting cells in the network and P_i is the number of paging transactions performed within the i^{th} cell. The variable D is the number of cells in the network. Finally, V_i is the vicinity factor of the i^{th} cell. Since the location update is considered to be 10 times more important than the paging, the variable β is a weight used to balance the objective function. For reliability and comparison purposes with state-of-the-art algorithms, we use the same value of β (β = 10) as the one used in previous works treating the RCP.

The vicinity of a cell is considered as the maximum number of cells that have to be browsed in case an incoming call occurs in this cell. More specifically, the vicinity value of the i^{th} reporting cell is the maximum number of non-reporting cells reachable from this cell without entering another reporting cell, including the reporting cell itself. A non-reporting cell can be reachable from different reporting cells. So, we can also define the vicinity of the i^{th} non-reporting cell as the maximum of the vicinity values of the reporting cells from which we can reach it.

Figure 13 represents a potential configuration of a network. The vicinity of the 6th non-reporting cell is the maximum vicinity value of its neighbouring reporting cells (2nd, 7th and the 11th) which are the values: 12, 15 and 6, respectively. Thus, the vicinity value of the 6th non-reporting cell is 10.

Figure 13 Vicinity calculation

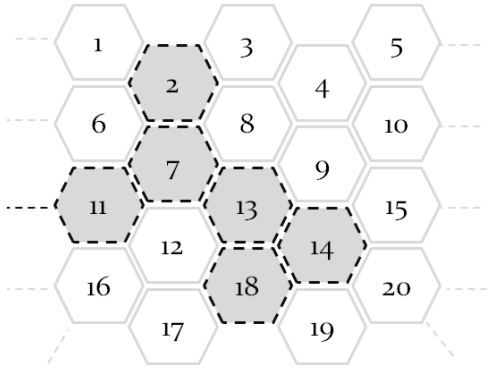
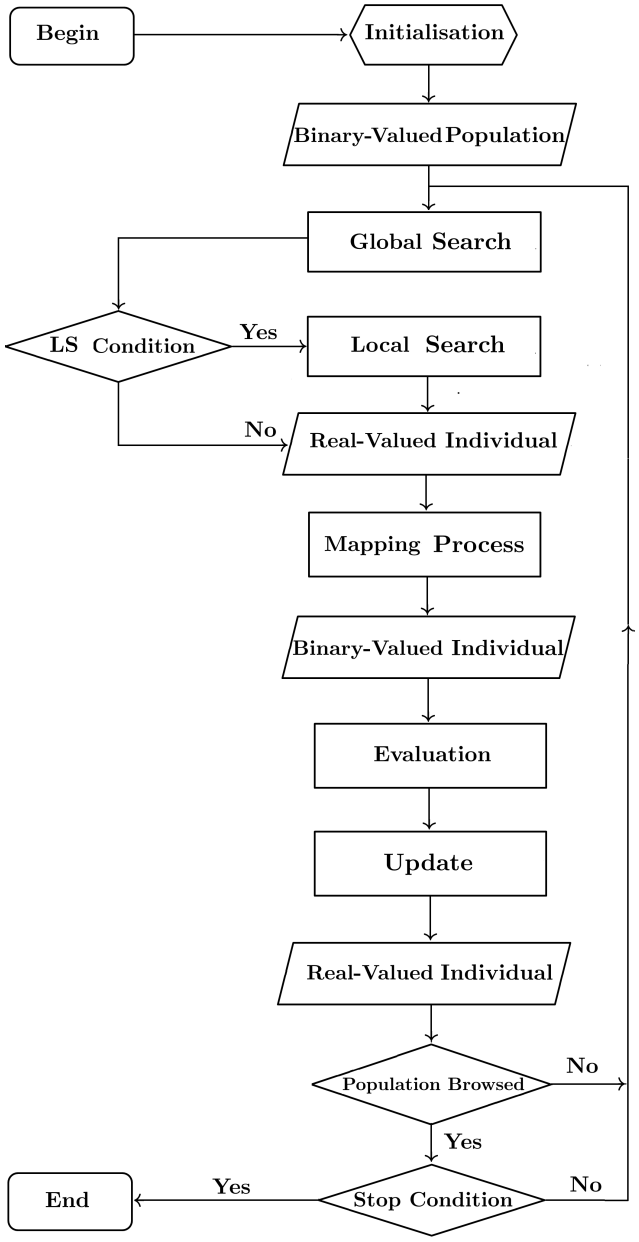


Figure 14 Flowchart of the binary BA



4 The binary BA

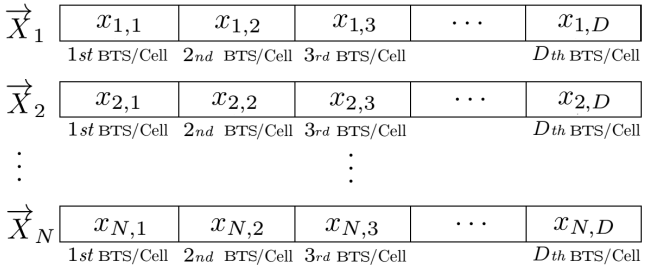
Like its continuous counterpart, the BBA makes a population of individuals evolve through iterations towards a fitter state using a series of perturbation phases. These phases are the initialisation, global search, local search, mapping process, evaluation and finally the update phase. Figure 14 illustrates the general framework of the BBA. In the following sections, we give more insight of each one of the BBA phases.

4.1 Initialisation

The BBA starts by initialising a population of N binary individuals, where each individual $\vec{X} = \{x_1, x_2, \ldots, x_D\}$, represents a solution to the problem being addressed (APP or RCP). When tackling the APP, each element x_j in a given individual, where $j = 1, \ldots, D$, represents the j^{th} antenna in the network (see Section 3.1.1), while when tackling the RCP, the element x_j represents the j^{th} cell in the network (see Section 3.2.1). Figure 15 illustrates an example of the BBA population.

Each element x_j is initialised by generating a random number R drawn from a standard uniform distribution. Then, if $R \geqslant 0.5$, x_j takes the value 1, otherwise 0. Once all individuals of the population initialised, their fitness value is evaluated using the objective function of the problem being addressed. In our work, when solving the APP, the evaluation is done using formula (14), while for the RCP, the evaluation is done using formula (19). After, once the population evaluated, the best individual $\vec{G}$ is extracted.

Figure 15 BBA population representation



After the initialisation, an iteration of the BBA consists of applying sequentially on each individual in the population the global/local search, the mapping, the evaluation and update phases. It is worth mentioning that within the first iteration of the BBA, the above-mentioned phases are performed on a binary population (the initial one), while starting from the second iteration, those phases are performed on a real-valued one.

4.2 Global and local search

Both global and local search are performed in the BBA as they are achieved in the original continuous BA (see Section 2.1.1).

4.3 Mapping

The individuals evolved and produced by the BBA are continuous ones. On the other hand, solutions to binary problems such as the APP and the RCP are encoded as binary vectors. Thus, a phase is needed to convert real-valued solutions into binary ones. This conversion process is done using one of the mapping techniques introduced in Section 2.2.2.

4.4 Evaluation and update

Once the mapping process achieved, the produced individuals are evaluated using the objective function of the problem being addressed. In our case, for the APP it is formula (14), while for the RCP, the evaluation is done using formula (19). Then, the update phase is achieved as described within the original BA (see Section 2.1.1).

It is to be noted that the proposed binary BA evolves a population of real-coded individuals all along its phases. In order to evaluate the quality of these individuals, they are temporarily converted into binary ones during the mapping phase. Thus, the update phase achieved after the evaluation phase is done on the real-coded solutions and not the binary ones.

Once all individuals of the population processed, they are ranked on the basis of their fitness values and the best individual $\vec{G}$ is updated. The end of the update phase for the last individual in the population marks the end of a single iteration of the BBA. Then, the whole process is reiterated until a given stop criterion is reached.

The inclusion of the discretisation step using one of the mapping functions introduced in Section 2.2.2 gives birth to five BBAs. The first variant, using the nearest integer method as a discretisation technique is called the nearest-integer-based binary bat algorithm (NI-BBA). The second variant is called the normalisation-based binary bat algorithm (N-BBA), which is based on the normalisation method. The third variant uses the sigmoid function and is called the sigmoid function-based binary bat algorithm (SF-BBA). The fourth variant is the angle modulation-based binary bat algorithm (AM-BBA), which is based on the angle modulation method. The last variant, the great-value priority-based binary bat algorithm (GVP-BBA), uses the great value priority technique as a discretisation method.

5 Experimental study and analysis

In this section, we present the design, results and discussion of the experiments performed on the proposed BBAs.

5.1 Experimental design

All our tests are run on a cluster of 16 machines with heterogeneous processors, under linux or windows OS. The implementation is done using MATLAB 7.12.0 (R2011a). Two optimisation problems are used, the APP and RCP, in order to illustrate several interesting

features in optimisation, such as epistatsis, multimodality, deceptiveness.

5.1.1 The APP

To investigate the robustness of the proposed BBAs, different types of datasets of the APP 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 (Priem-Mendes et al., 2009a, 2009) and can be found in 'The realistic benchmark instance of the antenna positioning problem: instance of 1000 positions' (http://oplink.unex.es/rnd/). The Synthetic instances used here are provided by the University of Laguna, Spain. They were used in several works (Alba and Chicano, 2005; Alba et al., 2006) and can be found in 'The synthetic benchmark instance of the antenna positioning problem: instance of 149 positions' (http://neo.lcc.uma.es/software/more/index.php). Randomly-generated instances are used too, they are provided by the University of Constantine, Algeria. They were used in Dahi et al. (2016a, 2016b, 2015) and available in 'The random benchmark instances of the antenna positioning problem: instances of 549 and 749 positions' (http://www.fichier-rar.fr/2014/10/03/random-instance/). Different-sized instances are used in order to assess the scalability of the proposed BBAs. Instances of dimensions 149, 349, 549, 749 and 1,000 are used. Each instance contains a specific number of potential positions of BTSs. Each position is identified by Cartesian coordinates (x, y).

The treated area is modelled as a discrete cell grid. Each cell represents a 15*15 m^2 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. Table 1 summarises the dataset types, their corresponding sizes and the coverage types associated to them. It is worth mentioning that within this section, the terms 'circle' and 'omnidirectional' describe 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 Figure 7): squared, omnidirectional and directive, introduced in Alba et al. (2006). The coverage parameters and the working area dimensions used here are those used in most of literature. They were extracted from Alba and Chicano (2005) and Alba et al. (2006) for synthetic data. Parameters of realistic data were extracted from Priem-Mendes et al. (2009a, 2009)]. Finally, for the randomly-generated instances, the parameters are extracted from Dahi et al. (2016a, 2016b, 2015). It is worth mentioning that directive antennas cover one sixth of the area of omnidirectional antennas with the same radius. Table 2 summarises the antenna coverage parameters.

Table 1 Instances: type, size and coverage

Instance	*Grid size*	*Instances size*	*Coverage type*
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	1,000	Omnidirectional

Table 2 Antenna coverage parameters

Data	*Antenna*	*Radius (cells)*
Synthetic	Omnidirectional	22
	Squared	20
	Directive	22
Random	Omnidirectional	26
	Squared	24
	Directive	26
Realistic	Omnidirectional	30
	Squared	None
	Directive	None

Table 3 BA parameters

Parameter	*Value*
f_{max}	10
f_{min}	–10
A_{max}	2
A_{min}	1
r_{max}	1
r_{min}	0
α	0.9
γ	0.9

The proposed BBA variants: NI-BBA, N-BBA, AM-BBA, SF-BBA and GVP-BBA have been compared to one of the top-ranked state-of-the-art metaheuristics used to solve the APP, which is the PBIL (Priem-Mendes et al., 2009a, 2009b). The PBIL parameters are extracted from the previously-mentioned works. Parameters of the BBAs used in this experiment are shown in Table 3. They were chosen on the basis of several tuning experiments.

The experiments are performed until reaching the termination criterion of 20,000 objective function evaluations. We use a population of 100 individuals,

200 iterations and all the experiments are repeated over 30 independent runs. Several results are reported such as the *best* and the *worst* fitness values, the *mean* and *standard deviation* of the fitness values over 30 executions. It is to be noted that a *While* loop is used instead of a *For* loop for a fair comparison on the basis of the maximum number of objective function evaluations.

5.1.2 The RCP

The scalability of the proposed BBAs is assessed using different-sized instances of the RCP. Twelve instances of sizes 4 × 4, 6 × 6, 8 × 8 and 10 × 10 cells are used. The instances were generated on the basis of realistic patterns and are provided by the University of Malaga, Spain. They were used in Almeida-Luz et al. (2011) and Berrocal-Plaza et al. (2015) and are available in 'The realistic benchmark instances of the reporting cell problem' (http://oplink.lcc.uma.es/problems/mmp.html). Each problem instance of the RCP contains a set of attributes (*Lu*, *P*), where *Lu* represents the location updates cost of a given cell in the network, while the variable *P* corresponds to the paging cost associated to this cell.

The five BBAs are compared against one of the top-ranked state-of-the-art algorithms used to solve the RCP, which is the DE algorithm (Almeida-Luz et al., 2011). The DE parameters used in this work are extracted from the previously-cited work, whereas the BBAs parameters used in this experiment are shown in Table 3.

For reliability and fair comparison with the DE, experiment parameters used within our work are the same as the ones used within the DE's original work (Almeida-Luz et al., 2011). Thus, experiments are performed until reaching the stop criterion of 175,000 fitness function evaluations. We use a population of 175 individuals, 1,000 iterations and all the algorithms are repeated over 30 independent runs. Several results are reported such as the *best* and the *worst* fitness values, the *mean* and *standard deviation* of the fitness values over 30 executions.

5.2 Numerical results and discussion

Tables 4–8 show the results obtained when evaluating the BBAs and the PBIL on Instances 149, 349, 549, 749 and 1,000. On the basis of the metric 'mean', the best results are highlighted in italics.

On the basis of the metric 'mean' of Tables 4–8, one can see that the NI-BBA is the best variant when tackling Instances 149 (squared, circle and directive coverages), 549 (squared and circle coverages), 749 (squared, circle and directive coverages) and 1000. The variant AM-BBA is assessed to be the best variant when tackling Instance 349 (squared, circle and directive coverages) and Instance 549 (directive coverage). It is to be noted that for Instance 149 (squared and circle coverages) the SF-BBA is assessed to be the worst variant, while for the remaining instances, it is the GVP-BBA that is found to be the worst variant.

Table 4 Results of the BBAs and PBIL for Instance 149

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Squared	AM-BBA	193.46	105.92	113.01	15.49
	GVP-BBA	123.64	104.38	113.67	4.60
	N-BBA	120.37	103.90	111.72	4.18
	NI-BBA	124.53	103.02	114.09	4.97
	SF-BBA	118.32	101.51	108.85	4.46
	PBIL	190.12	168.30	*177.77*	6.23
Circle	AM-BBA	102.78	92.64	96.39	2.52
	GVP-BBA	108.34	93.06	100.94	3.25
	N-BBA	103.44	93.17	98.21	2.76
	NI-BBA	114.42	94.14	100.98	4.40
	SF-BBA	103.51	91.41	96.05	3.59
	PBIL	146.06	134.24	*138.88*	3.29
Directive	AM-BBA	42.72	41.65	42.08	0.28
	GVP-BBA	42.54	39.62	40.66	0.72
	N-BBA	44.10	40.48	42.04	0.79
	NI-BBA	43.93	40.79	42.28	0.81
	SF-BBA	44.38	40.96	42.26	0.70
	PBIL	48.75	47.92	*48.35*	0.24

Table 5 Results of the BBAs and PBIL for Instance 349

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Squared	AM-BBA	201.25	97.09	*147.50*	43.45
	GVP-BBA	65.12	59.84	61.70	1.29
	N-BBA	107.11	82.52	95.93	5.39
	NI-BBA	114.35	87.44	99.49	5.84
	SF-BBA	113.14	90.88	99.12	4.80
	PBIL	146.98	131.84	139.14	3.69
Circle	AM-BBA	142.84	86.20	120.01	21.53
	GVP-BBA	63.03	58.44	60.62	1.28
	N-BBA	93.53	77.79	86.08	3.68
	NI-BBA	96.21	83.02	88.98	3.57
	SF-BBA	98.01	73.36	87.32	5.06
	PBIL	126.45	116.30	*121.14*	2.76
Directive	AM-BBA	42.76	39.88	41.57	1.10
	GVP-BBA	40.84	39.07	39.98	0.46
	N-BBA	43.07	39.41	41.06	0.91
	NI-BBA	42.28	39.43	41.02	0.78
	SF-BBA	42.14	39.04	40.52	0.91
	PBIL	51.52	49.49	*50.47*	0.46

It is also worth mentioning that when tackling Instance 149, all the BBAs achieve almost similar results when using the three types of coverage. But, as the size of the instance increases, the difference in the efficiency is more noticeable. In fact, the efficiency of AM-BBA, NI-BBA, N-BBA and SF-BBA is not affected, while the one of GVP-BBA is greatly reduced. It is to be noted also that for all sizes of the problem, when using the directive coverage,

all the variants achieve very close results. Another fact that needs to be noticed is that for all sizes of the problem the best variants AM-BBA, NI-BBA, N-BBA and SF-BBA are the ones having the highest standard deviation, while the worst one, GVP-BBA, is the one that have the smallest standard deviation.

Table 6 Results of the BBAs and PBIL for Instance 549

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Squared	AM-BBA	130.50	119.68	124.01	2.91
	GVP-BBA	42.86	39.35	40.54	0.82
	N-BBA	137.69	111.09	122.75	7.51
	NI-BBA	146.94	111.83	*128.42*	7.09
	SF-BBA	142.72	114.34	128.08	6.56
	PBIL	135.28	113.29	125.52	5.28
Circle	AM-BBA	123.24	106.23	110.91	3.32
	GVP-BBA	42.44	39.43	40.44	0.71
	N-BBA	119.95	92.59	108.62	6.34
	NI-BBA	127.42	102.97	112.57	6.71
	SF-BBA	122.15	78.31	110.26	8.07
	PBIL	128.97	108.63	*117.32*	5.22
Directive	AM-BBA	52.60	50.04	50.94	0.68
	GVP-BBA	37.15	35.36	36.10	0.43
	N-BBA	54.64	47.71	50.47	1.80
	NI-BBA	53.50	48.22	50.57	1.09
	SF-BBA	54.18	46.71	50.79	1.69
	PBIL	63.99	58.52	*60.76*	1.19

Again, considering the metric 'mean' of Tables 4–8, one can observe that the proposed BBAs outperform the state-of-the-art algorithm PBIL in 5 out of 13 instances. In fact, the variant AM-BBA outperforms the PBIL in Instance 349 (squared coverage). Taking now the variant NI-BBA, the latter outperforms the PBIL in Instance 549 (squared coverage), Instance 749 (squared and circle coverages) and finally, Instance 1,000. However, it is worth mentioning that the PBIL could outperform all the BBAs in the remaining instances.

Figures 16–20 represent the fitness evolution of the proposed BBAs and the PBIL when tackling Instances 149, 349, 549, 749 and 1,000 with circle coverage.

Considering the fitness value evolution shown in Figures 16–20, one can see that for Instance 149, all the BBAs have a similar convergence rate, while as the size of the instances increases, the difference in the convergence is more noticeable. Actually, for Instance 349, the GVP-BBA has the slowest convergence rate, whereas the N-BBA has a moderate rate of convergence, the SF-BBA, AM-BBA and NI-BBA have the quickest convergence. These behaviours are also noticed for Instances 549, 749 and 1,000.

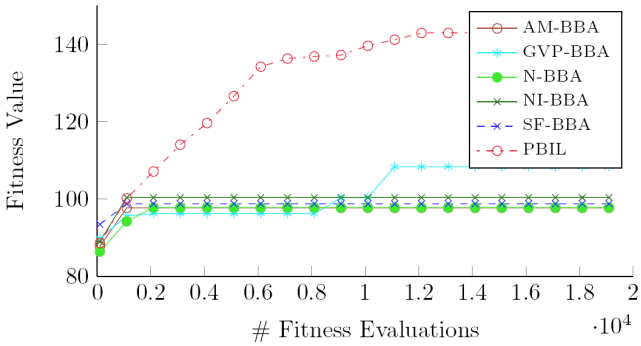
Table 7 Results of the BBAs and PBIL for Instance 749

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Squared	AM-BBA	128.51	115.77	122.30	2.94
	GVP-BBA	30.52	28.80	29.41	0.45
	N-BBA	137.40	111.90	124.15	7.41
	NI-BBA	139.20	113.98	*125.28*	5.67
	SF-BBA	136.77	110.51	123.10	5.64
	PBIL	93.66	78.69	85.07	3.75
Circle	AM-BBA	117.10	102.69	108.85	3.37
	GVP-BBA	30.41	28.73	29.28	0.47
	N-BBA	124.16	88.13	108.89	8.28
	NI-BBA	121.06	102.41	*111.56*	5.10
	SF-BBA	127.43	79.17	109.91	9.42
	PBIL	91.37	72.70	81.90	4.97
Directive	AM-BBA	52.45	49.52	50.50	0.60
	GVP-BBA	28.43	27.37	27.91	0.30
	N-BBA	53.57	46.64	50.57	1.68
	NI-BBA	53.67	47.18	51.19	1.48
	SF-BBA	54.67	47.52	50.51	1.48
	PBIL	56.39	50.60	*53.81*	1.60

Table 8 Results of the BBAs and PBIL for Instance 1,000

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Circle	AM-BBA	82.49	71.37	77.00	2.60
	GVP-BBA	18.01	17.29	17.64	0.23
	N-BBA	93.78	69.46	80.05	6.64
	NI-BBA	97.58	72.65	*82.38*	6.01
	SF-BBA	90.93	62.37	80.74	6.13
	PBIL	45.24	35.74	40.12	2.26

Figure 16 Fitness value evolution for Instance 149 with circle coverage (see online version for colours)



With regards to Figures 16–20, it is clear that for Instances 149 and 349, the PBIL has a quicker convergence rate than all the BBAs, while for Instances 549, 749 and 1,000, the PBIL has a constant but swift convergence rate when compared to the GVP-BBA, and a slow convergence rate when compared to the remaining BBAs. It is also worth mentioning that the convergence evolution of the proposed

BBAs is more clear and sudden, while the one of the PBIL is constant and regular.

On the basis of the metrics 'mean' and 'STD' of Tables 4–8 and the fitness evolution displayed in Figures 16–20, one can explain that the good results achieved by the SF-BBA, NI-BBA, N-BBA and AM-BBA is due to a good exploration at the first stages of the search, while the bad results of the GVP-BBA are caused by a lack of diversity in the population (exploration), which made them fall into local optima since the first algorithm iterations.

Figure 17 Fitness value evolution for Instance 349 with circle coverage (see online version for colours)

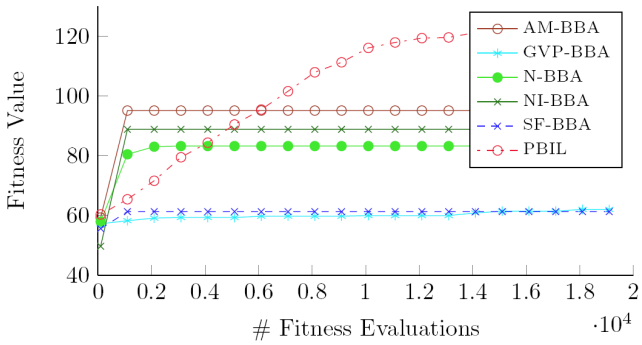


Figure 18 Fitness value evolution for Instance 549 with circle coverage (see online version for colours)

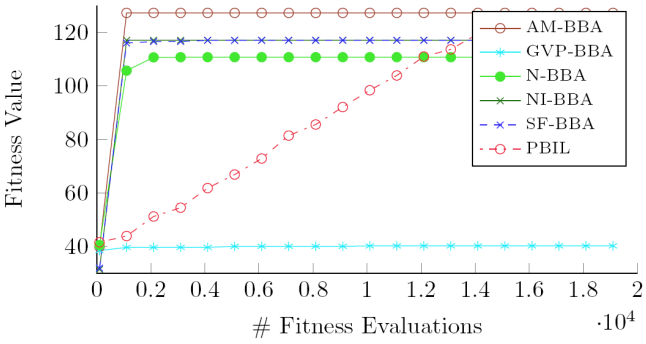


Figure 19 Fitness value evolution for Instance 749 with circle coverage (see online version for colours)

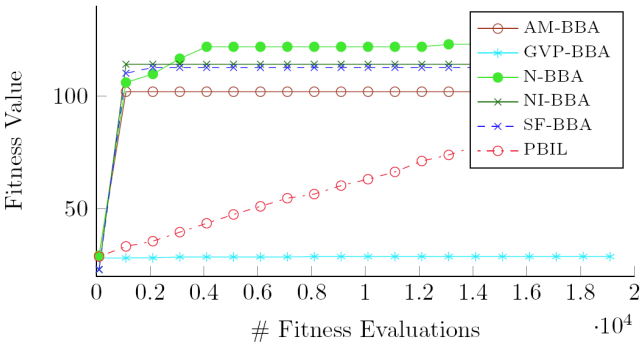


Figure 20 Fitness value evolution for Instance 1,000 (see online version for colours)

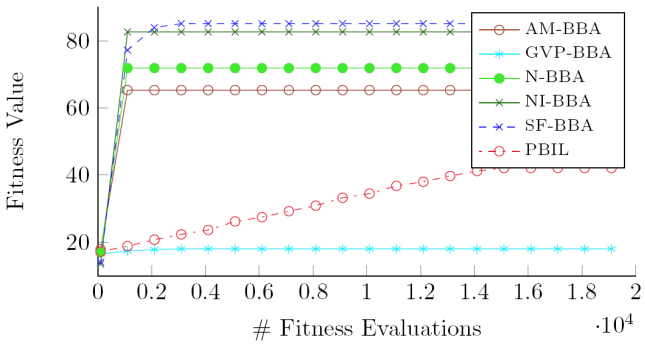


Table 9 Results of the BBAs and DE for networks 1, 2 and 3 of size 4 × 4 cells

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Network 1	AM-BBA	98,535	99,160	98,677.33	207.78
	GVP-BBA	98,535	102,549	99,645.43	1,081.63
	N-BBA	98,535	101,925	99,719.90	1,150.69
	NI-BBA	98,535	103,625	99,958.60	1,508.56
	SF-BBA	98,535	101,419	99,250.13	802.29
	DE	98,535	98,535	*98,535.00*	0.00
Network 2	AM-BBA	97,156	98,065	97,473.47	323.92
	GVP-BBA	97,764	99,972	99,098.60	549.78
	N-BBA	97,156	100,681	98,541.33	996.93
	NI-BBA	97,156	101,334	98,368.40	1,086.91
	SF-BBA	97,156	100,946	98,633.43	1,087.46
	DE	97,156	97,156	*97,156.00*	0.00
Network 3	AM-BBA	95,038	98,995	96,506.97	1,607.84
	GVP-BBA	95,038	102,416	97,918.27	2,332.05
	N-BBA	95,038	104,393	99,453.53	2,729.59
	NI-BBA	95,038	105,474	99,783.07	2,810.97
	SF-BBA	95,038	103,096	98,821.87	1,866.58
	DE	95,038	95,038	*95,038.00*	0.00

Tables 9–12 show the results obtained when evaluating the BBAs and the DE on networks of sizes 4 × 4, 6 × 6, 8 × 8 and 10 × 10 cells. On the basis of the metric 'mean', the best results are highlighted in italic.

With respect to the metric 'mean' in Tables 9–12, one can see that the AM-BBA is assessed to be the best variant when tackling networks 1, 2 and 3 of size 4 × 4 cells, networks 1 and 3 of size 6 × 6 cells, network 1 of size 8 × 8 cells and networks 1, 2 and 3 of size 10 × 10 cells. The variant GVP-BBA is proved to be the best variant when solving network 2 of size 8 × 8 and 6 × 6 cells, while the SF-BBA is assessed to be the best when addressing network 3 of size 8 × 8 cells.

On the other hand, the variant GVP-BBA is proved to be the worst variant when solving network 2 and 3 of sizes 4 × 4 and 10 × 10 cells. The variant NI-BBA is proved to be the worst variant when solving network 3 of size 8 × 8 cells and network 1 of size 4 × 4 cells, whereas the worst variant found when tackling networks 1 and 2 of size 6 × 6 cells

and network 2 of size 8 × 8 cells, is the SF-BBA. The variant N-BBA is assessed to be the worst one when solving network 1 of size 8 × 8 cells. Finally, when solving network 3 of size 4 × 4 cells, the NI-BBA is proved to be the worst.

Table 10 Results of the BBAs and DE for networks 1, 2 and 3 of size 6 × 6 cells

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Network 1	AM-BBA	173,753	176,763	174,259.43	662.09
	GVP-BBA	190,841	208,420	198,890.80	3,875.65
	N-BBA	176,722	214,309	197,435.13	8,330.90
	NI-BBA	184,447	209,875	197,752.27	6,215.43
	SF-BBA	186,678	208,885	200,169.70	5,215.91
	DE	173,701	175,241	*174,060.33*	662.48
Network 2	AM-BBA	199,101	214,784	207,934.67	3,270.27
	GVP-BBA	192,455	212,058	204,782.80	4,521.91
	N-BBA	192,933	217,592	206,068.53	6,424.60
	NI-BBA	195,152	212,121	205,482.03	4,379.91
	SF-BBA	195,978	216,691	209,128.47	4,270.34
	DE	182,331	182,331	*182,331.00*	0.00
Network 3	AM-BBA	182,466	198,197	184,696.57	4,148.98
	GVP-BBA	185,606	204,671	195,904.90	4,916.80
	N-BBA	183,153	206,714	195,403.90	5,483.37
	NI-BBA	183,707	209,527	193,594.53	6,177.83
	SF-BBA	181,949	206,283	193,709.30	6,152.20
	DE	174,519	174,519	*174,519.00*	0.00

Table 11 Results of the BBAs and DE for networks 1, 2 and 3 of size 8 × 8 cells

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Network 1	AM-BBA	342,478	356,488	350,065.23	3,461.36
	GVP-BBA	336,923	366,403	350,283.80	8,144.76
	N-BBA	337,268	372,176	353,994.00	8,453.31
	NI-BBA	333,626	367,003	351,544.30	8,625.52
	SF-BBA	332,577	364,886	350,873.47	7,313.59
	DE	308,401	312,355	*310,880.23*	1,520.81
Network 2	AM-BBA	330,278	341,702	336,650.77	3,135.73
	GVP-BBA	321,580	346,785	334,249.77	5,802.83
	N-BBA	312,269	346,652	336,479.70	8,381.56
	NI-BBA	323,408	346,584	336,340.73	5,810.11
	SF-BBA	318,970	349,908	336,792.57	6,217.60
	DE	287,149	295,869	*287,988.73*	2,565.92
Network 3	AM-BBA	305,287	319,267	311,729.37	3,657.77
	GVP-BBA	298,532	319,644	310,979.83	5,754.56
	N-BBA	296,317	324,606	311,401.50	6,728.99
	NI-BBA	301,872	324,648	312,912.03	5,744.11
	SF-BBA	294,817	320,199	310,725.37	5,473.07
	DE	264,204	265,324	*264,475.13*	347.97

Table 12 Results of the BBAs and DE for networks 1, 2 and 3 of size 10 × 10 cells

	Algorithm	*Best*	*Worst*	*Mean*	*STD*
Network 1	AM-BBA	436,293	451,680	445,647.20	4,105.03
	GVP-BBA	439,565	489,433	465,656.53	14,663.74
	N-BBA	424,821	462,765	450,950.13	8,071.37
	NI-BBA	431,059	465,506	449,076.10	8,527.40
	SF-BBA	432,652	460,630	446,434.60	6,564.68
	DE	386,474	402,531	*390,149.70*	2,996.00
Network 2	AM-BBA	401,295	410,274	405,181.67	2,423.76
	GVP-BBA	398,786	435,933	417,884.50	10,401.66
	N-BBA	388,642	432,558	411,115.37	8,428.72
	NI-BBA	392,425	423,856	408,718.27	7,023.80
	SF-BBA	391,362	415,614	405,985.73	5,333.75
	DE	358,167	367,092	*362,130.57*	2,390.40
Network 3	AM-BBA	404,228	416,827	412,802.23	2,870.56
	GVP-BBA	406,333	451,745	420,978.13	10,482.13
	N-BBA	404,193	428,568	415,464.60	6,259.95
	NI-BBA	400,993	426,889	415,771.70	5,911.04
	SF-BBA	403,750	422,209	413,477.80	5,173.29
	DE	371,924	385,701	*378,903.03*	3,597.84

Figure 21 Fitness value evolution for network 1 of size 4 × 4 cells (see online version for colours)

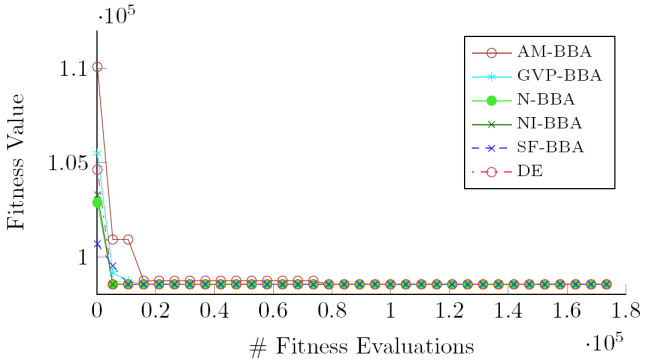
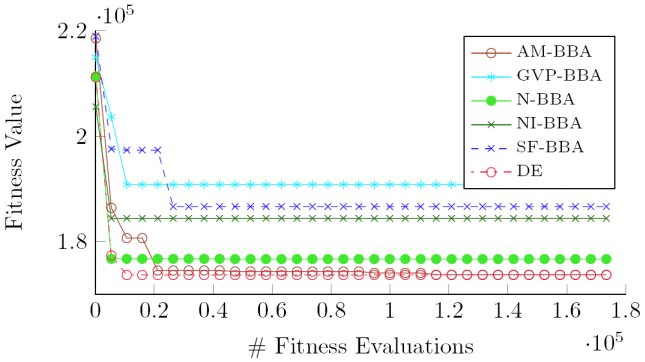


Figure 22 Fitness value evolution for network 1 of size 6 × 6 cells (see online version for colours)



It is to be also noted that the BBAs have close if not similar results when tackling all instances of the RCP. However, when considering the metric 'STD', the AM-BBA has generally a smaller deviation, while the remaining BBA

variants have all higher deviation when compared to the one of AM-BBA.

Again, when considering the metric 'Mean' of Tables 9–12, the state-of-the-art algorithm DE is able to outperform the BBAs in all instances. However, when considering the metric 'Best', all the BBAs are able to achieve the same results as the ones obtained by the DE when tackling networks 1, 2 and 3 of size 4 × 4 cells.

Figure 23 Fitness value evolution for network 1 of size 8 × 8 cells (see online version for colours)

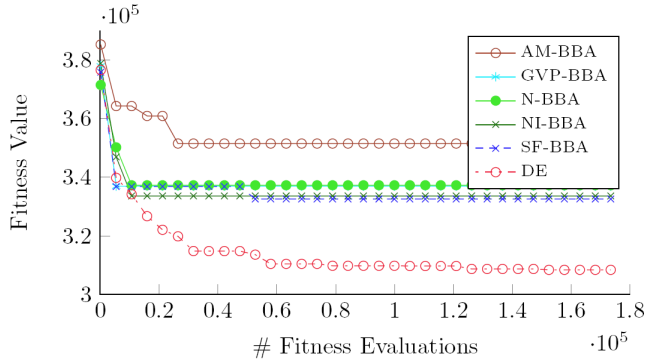
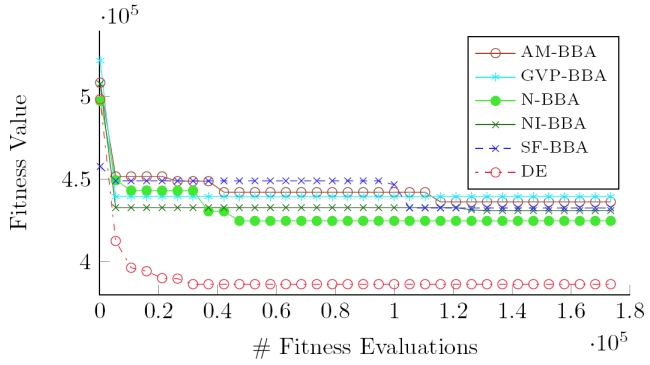


Figure 24 Fitness value evolution for network 1 of size 10 × 10 cells (see online version for colours)



Figures 21–24 represent the fitness evolution of the proposed BBAs and the DE when tackling network 1 of sizes 4 × 4, 6 × 6, 8 × 8 and 10 × 10 cells.

On the basis of the fitness value evolution displayed in Figures 21–24, one can observe that when tackling instance of size 4 × 4 cells, all BBAs have a quick and sudden convergence since the first algorithm iterations, while when solving instance of size 6 × 6 cells, the convergence rate differs from one variant to another. Indeed, the SF-BBA and the GVP-BBA have a slow convergence, whereas the AM-BBA and N-BBA have a quicker one. Finally, the NI-BBA has a moderate convergence when compared to the remaining BBAs. Taking now the DE algorithm, one can see that for all sizes of problems, the DE has a quicker convergence rate when compared to the BBAs.

With regards to results in Tables 9–12 and the fitness value displayed in Figures 21–24, one can explain the fact that no particular BBA variants clearly outperform the other BBAs, since for all sizes of the instances, the BBAs have almost similar convergence rate, which tend to affirm that they have the same exploration/exploitation balance.

Considering results of Tables 4-12 and the fitness value evolution displayed in Figures 16-24, one can conclude that the efficiency of the mapping techniques is driven by the type (i.e., minimisation and maximisation) and the size of the problem tackled. The latter influence their convergence rate. In fact, the mapping techniques angle modulation and nearest-integer are assessed to be the best when addressing a maximisation problem, while the angle modulation is assessed to be the best when tackling a minimisation problem. In addition, it can be noticed that for small instances the mapping techniques have close efficiency. However, as the size of the problem increases, their efficiency starts to differ. This is mainly due to a change in the convergence rate of the BBAs. Again with respect to results of Tables 4–12 and the fitness value evolution displayed in Figures 16–24, one can see that the BA has some shortfalls when compared to top-ranked algorithms. However, the proposed BBAs could achieve very promising results and even outperform state-of-the-art algorithms for some instances.

5.3 Statistical analysis tests

Both experiments conducted on the APP and RCP are statistically significant. Before deciding what type of statistical test to perform, two assumptions have to be verified: distribution normality and variance homogeneity. We perform first the one-sample Kolmogorov-Smirnov as a normality distribution test. Then, we conduct the Brown-Forsythe as variance homogeneity test. Since both assumptions are not fulfilled, parametric distribution equality tests cannot be applied. So, we perform the Kruskal-Wallis as a variance analysis test. The Kruskal-Wallis test succeeds to reject the null-hypothesis, which means that a difference exist in the distribution of the results obtained by the algorithms.

It is worth mentioning that all our statistical tests are hypothesis tests based on the acceptance or the rejection of the null-hypothesis (i.e., assumptions). The latter is specific to each test (i.e., distribution normality, variance homogeneity, distribution equality, etc.). In addition, all tests are conducted between samples of data. A given sample represents the results obtained by a given algorithm over the whole set of executions conducted on a given instance. Furthermore, all tests are performed using a level of significance of 5%.

Finally, in order to find where the difference occurs and which algorithm is the best, we perform a post-hoc test. For a given problem instance, the post-hoc performs a series of pairwise comparison between a reference algorithm and another algorithm. The assumptions of this test are that there is no difference between the distribution of both algorithms being compared. Mathematically speaking, it ranks the algorithms on the basis of their results in each execution. Then, the mean of these ranks is taken.

It is also to be noted that the null-hypothesis of the post-hoc test is not rejected for most comparisons made between state-of-the-art PBIL or DE and the remaining BBAs. Statistically speaking, this proves that the BBAs

gave results as good as the ones achieved by the PBIL or the DE.

Table 13 Statistical ranking for the APP instances

	Algorithm	*Squared*	*Circle*	*Directive*
Instance 149	AM-BBA	63.93	51.30	85.23
	GVP-BBA	92.23	104.77	23.63
	N-BBA	75.57	74.00	81.67
	NI-BBA	97.17	99.93	94.17
	SF-BBA	49.60	47.50	92.80
	PBIL	*164.50*	*165.50*	*165.50*
Instance 349	AM-BBA	138.03	145.13	104.03
	GVP-BBA	15.50	15.50	36.30
	N-BBA	68.13	69.67	87.03
	NI-BBA	86.87	88.03	86.97
	SF-BBA	86.23	78.50	63.17
	PBIL	*148.23*	*146.17*	*165.50*
Instance 549	AM-BBA	88.63	98.10	97.27
	GVP-BBA	15.50	15.50	15.50
	N-BBA	89.97	85.23	85.57
	NI-BBA	*121.97*	106.90	85.37
	SF-BBA	120.83	97.93	93.80
	PBIL	106.10	*139.33*	*165.50*
Instance 749	AM-BBA	112.27	110.60	86.13
	GVP-BBA	15.50	15.50	15.50
	N-BBA	121.93	116.17	90.57
	NI-BBA	*130.47*	*129.07*	108.37
	SF-BBA	117.33	125.03	85.40
	PBIL	45.50	46.63	*157.03*
Instance 1,000	AM-BBA	/	101.10	/
	GVP-BBA	/	15.50	/
	N-BBA	/	119.00	/
	NI-BBA	/	*134.73*	/
	SF-BBA	/	127.17	/
	PBIL	/	45.50	/

Tables 13 and 14 represent results of the statistical ranking performed on both instances of the APP and RCP, respectively. It is worth mentioning that the post-hoc gives higher scores to the algorithm with the highest fitness. This means that for maximisation problems such as the APP, the algorithm with the highest mean of ranks is the best one, while for minimisation problems such as the RCP, the algorithm with smaller mean of ranks is the best one. Under the light of these explanations, the best algorithm is highlighted in italic in Tables 13 and 14.

6 Conclusions

In this paper, we conducted a systematic study to investigate several aspects of the discretisation of continuous metaheuristics. This was done by proposing five new BBAs. These variants were tested for solving two optimisation problems in cellular phone networks which are the APP and the RCP. The experiments were performed on different types and sizes of benchmarks. The proposed BBAs were also compared to two top-ranked state-of-the-art algorithms: the DE algorithm and the population-based incremental learning algorithm, designed to solve the APP and the RCP, respectively. Several statistical tests have been conducted as well.

Table 14 Statistical ranking for the RCP instances

	Algorithm	*Network 1*	*Network 2*	*Network 3*
4 × 4 cells	AM-BBA	62.00	63.63	63.48
	GVP-BBA	120.68	140.38	96.15
	N-BBA	114.50	114.47	119.15
	NI-BBA	112.23	104.88	124.40
	SF-BBA	94.58	99.73	103.32
	DE	*39.00*	*19.90*	*36.50*
6 × 6 cells	AM-BBA	39.00	124.17	53.23
	GVP-BBA	121.55	93.57	123.60
	N-BBA	112.65	107.78	120.27
	NI-BBA	112.90	100.80	105.93
	SF-BBA	134.87	101.18	124.47
	DE	*22.03*	*15.50*	*15.50*
8 × 8 cells	AM-BBA	101.20	107.87	104.33
	GVP-BBA	101.63	90.20	102.17
	N-BBA	122.80	115.13	101.77
	NI-BBA	112.70	106.03	112.73
	SF-BBA	89.17	108.27	106.50
	DE	*15.50*	*15.50*	*15.50*
10 × 10 cells	AM-BBA	79.12	77.95	86.77
	GVP-BBA	148.43	139.03	130.67
	N-BBA	112.37	114.93	109.02
	NI-BBA	97.62	102.70	111.43
	SF-BBA	89.97	92.88	89.62
	DE	*15.50*	*15.50*	*15.50*

The results demonstrated that the efficiency of the mapping techniques is driven by the nature and the size of the problem being solved. The experiments also showed that depending on the type of problem addressed, the nearest-integer and the angle modulation are the best mapping techniques and the most scalable ones. In addition, it was proven that the BA has some shortcomings when compared to the top-ranked algorithms, but it still showed some promising results and even outperformed the state-of-the-art algorithms for some instances.

As perspective of research, we intend to design smart mapping techniques that can dynamically switch between binary and grey-coded solution representations. Moreover, we think of conceiving a universal technique that allow mapping continuous, binary and integer-coded solutions.

References

'Matlab code of the bat algorithm' [online] https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm--demo- (accessed 27 May 2017).

'The random benchmark instances of the antenna positioning problem: instances of 549 and 749 positions' [online] http://www.fichier-rar.fr/2014/10/03/random-instance/ (accessed 27 May 2017).

'The realistic benchmark instance of the antenna positioning problem: instance of 1000 positions' [online] http://oplink.unex.es/rnd/ (accessed 27 May 2017).

'The realistic benchmark instances of the reporting cell problem' [online] http://oplink.lcc.uma.es/problems/mmp.html (accessed 27 May 2017).

'The synthetic benchmark instance of the antenna positioning problem: instance of 149 positions' [online] http://neo.lcc.uma.es/software/more/index.php (accessed 27 May 2017).

Abdelsalam, H.M. and Al-shaar, A. (2013) 'An enhanced binary particle swarm optimization algorithm for channel assignment in cognitive radio networks', in *Proceedings of International Conference on Modelling and Identification Control, (ICMIC)*, IEEE, pp.221–226.

Alba, E. and Chicano, F. (2005) 'On the behavior of parallel genetic algorithms for optimal placement of antennae in telecommunications', *International Journal of Foundations of Computer Science*, Vol. 16, No. 2, pp.343–359.

Alba, E., García-Nieto, J., Taheri, J. and Zomaya, A. (2008) 'New research in nature inspired algorithms for mobility management in GSM networks', in *Proceedings of the EvoWorkshops on the Applications of Evolutionary Computing: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM, EvoSTOC, and EvoTransLog*, Springer, pp.1–10.

Alba, E., Molina, G. and Chicano, F. (2006) 'Optimal placement of antennae using metaheuristics', in *Proceedings of the 6th International Conference on Numerical Methods and Applications (NMA)*, IEEE, pp.214–222.

Almeida-Luz, S.M., Vega-Rodríguez, M.A., Gómez-Púlido, J.A. and Sánchez-Pérez, J.M. (2011) 'Differential evolution for solving the mobile location management', *Applied Soft Computing*, Vol. 11, No. 1, pp.410–427.

Al-Muraeb, A. and Abdel-Aty-Zohdy, H. (2016) 'Optimal design of short fiber Bragg grating using bat algorithm with adaptive position update', *IEEE Photonics Journal*, Vol. 8, No. 1, pp.1–11.

Aratsu, Y., Mizuno, K., Sasaki, H. and Nishihara, S. (2013) '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, pp.134–139.

Bäck, T. (1996) *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, Oxford, UK.

Bäck, T., Hammel, U. and Schwefel, H.P. (1997) 'Evolutionary computation: comments on the history and current state', *IEEE Trans. Evolutionary Computation*, Vol. 1, No. 1, pp.3–17.

Bar-Noy, A. and Kessler, I. (1993) 'Tracking mobile users in wireless communications networks', *IEEE Transactions on Information Theory*, Vol. 39, No. 6, pp.1877–1886.

Berrocal-Plaza, V., Vega-Rodríguez, M.A. and Sánchez-Pérez, J.M. (2015) 'A multiobjective study of the gaussian cluster paging in the reporting cells strategy', *Applied Soft Computing*, Vol. 28, pp.332–344.

Calegari, P., Guidec, F. and Kuonen, P. (1996) 'A parallel genetic approach to transceiver placement optimisation', in *Proceedings of the SIPAR Workshop: Parallel and Distributed Systems*, pp.21–24.

Calegari, P.R. (1999) *Parallelization of Population-based evolutionary algorithms for Combinatorial Optimization Problems*, PhD thesis, Claude Bernard University.

Caruana, R. and Schaffer, J.D. (1988) 'Representation and hidden bias: gray vs. binary coding for genetic algorithms', in *Proceedings of the 5th International Conference on Machine Learning (ICML)*, pp.153–161.

Chawla, M. and Duhan, M. (2015) 'Bat algorithm: a survey of the state-of-the-art', *Appl. Artif. Intell.*, Vol. 29, No. 6, pp.617–634.

Congying, L., Huanping, Z. and Xinfeng, Y. (2011) 'Particle swarm optimization algorithm for quadratic assignment problem', in *Proceedings of the International Conference on Computer Science and Network Technology (ICCSNT)*, IEEE, Vol. 3, pp.1728–1731.

Crawford, B., Soto, R., Olea, C., Johnson, F. and Paredes, F. (2015) 'Binary bat algorithms for the set covering problem', in *Proceedings of the 10th Iberian Conference on Information Systems and Technologies (CISTI)*, IEEE, pp.1–4.

Dahi, Z.A.E.M., Chaker, M. and Draa, A. (2016a) 'A quantum-inspired genetic algorithm for solving the antenna positioning problem', *Swarm and Evolutionary Computation*, Vol. 31, pp.24–63.

Dahi, Z.A.E.M., Mezioud, C. and Draa, A. (2016b) 'On the efficiency of the binary flower pollination algorithm: application on the antenna positioning problem', *Applied Soft Computing*, Vol. 47, pp.395–414.

Dahi, Z.A.E.M., Mezioud, C. and Draa, A. (2015) 'Binary bat algorithm: On the efficiency of mapping functions when handling binary problems using continuous-variable-based metaheuristics', in *Proceedings of the 5th IFIP-TC International Conference on Computer Science and Its Applications (CIIA)*, Springer, Vol. 456, pp.3–14.

Deng, C., Zhao, B., Yang, Y. and Zhang, H. (2013) 'Binary encoding differential evolution with application to combinatorial optimization problem', in *Proceedings of the Chinese Intelligent Automation Conference*, Springer, Vol. 255, pp.77–84.

Djelloul, H., Sabba, S. and Chikhi, S. (2014) 'Binary bat algorithm for graph coloring problem', in *Proceedings of the 2nd World Conference on Complex Systems (WCCS)*, IEEE, pp.481–486.

Enache, A.C. and Sgarciu, V. (2015) 'A feature selection approach implemented with the binary bat algorithm applied for intrusion detection', in *Proceedings of the 38th International Conference on Telecommunications and Signal Processing (TSP)*, IEEE, pp.11–15.

Enache, A.C., Sgarciu, V. and Petrescu-Nita, A. (2015) 'Intelligent feature selection method rooted in binary bat algorithm for intrusion detection', in *Proceedings of the 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics (SACI)*, IEEE, pp.517–521.

Gandomi, A.H., Yang, X.S., Alavi, A.H. and Talatahari, S. (2012) 'Bat algorithm for constrained optimization tasks', *Neural Computing and Applications*, Vol. 22, No. 6, pp.1239–1255.

Gao, M.L., Shen, J., Yin, L.J., Liu, W., Zou, G.F., Li, H.T. and Fu, G.X. (2016) 'A novel visual tracking method using bat algorithm', *Neurocomputing*, Vol. 177, pp.612–619.

Ghaleb, M.A. and Alharkan, I.M. (2015) 'Efficient metaheuristics for multi-stage no-wait flexible flow-shop scheduling problem', in *Proceedings of the International Conference on Operations Excellence and Service Engineering*, pp.92–97.

Hac, A. and Zhou, X. (1997) 'Locating strategies for personal communication networks: a novel tracking strategy', *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 8, pp.1425–1436.

Jaddi, N.S., Abdullah, S. and Hamdan, A.R. (2015) 'Multi-population cooperative bat algorithm-based optimization of artificial neural network model', *Information Sciences*, Vol. 294, pp.628–644.

Jiang, D., Peng, C., Fan, Z. and Chen, Y. (2013) '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, pp.23–28.

Kang, M., Kim, J. and Kim, J.M. (2015) 'Reliable fault diagnosis for incipient low-speed bearings using fault feature analysis based on a binary bat algorithm', *Information Sciences*, Vol. 294, pp.423–438.

Keles, A. (2007) 'Binary differential evolution for the unit commitment problem', in *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO)*, ACM, pp.2765–2768.

Kennedy, J. and Eberhart, R.C. (1997) 'A discrete binary version of the particle swarm optimization algorithm', in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, IEEE, pp.41404–4108.

Kimiyaghalam, A., Mahdavi, M., Ashouri, A. and Bagherivand, M. (2013) '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, pp.1–4.

Kora, P. and Kalva, S.R. (2015) 'Improved bat algorithm for the detection of myocardial infarction', *SpringerPlus*, Vol. 4, No. 1, pp.1–18.

Krause, J., Cordeiro, J., Parpinelli, R.S. and Silvério Lopes, H. (2013) 'A survey of swarm algorithms applied to discrete optimization problems', in *Swarm Intelligence and Bio-Inspired Computation*, Elsevier, pp.169–191.

Laamari, M.A. and Kamel, N. (2014) 'A hybrid bat based feature selection approach for intrusion detection', in *Proceedings of the 9th International Conference on Bio-Inspired Computing – Theories and Applications (BIC-TA)*, Springer, pp.230–238.

Laamari, M.L. and Kamel, N. (2015) 'A multi-objective binary bat algorithm', in *Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication (IPAC)*, ACM, pp.75:1–75:5.

Liu, Q., Lu, W. and Xu, W. (2014) 'Spectrum allocation optimization for cognitive radio networks using binary firefly algorithm', in *Proceedings of the International Conference on Innovative Design and Manufacturing (ICIDM)*, IEEE, pp.257–262.

Martinez Cruz, A., Barron Fernandez, R. and Molina Lozano, H. (2013) 'Automated functional coverage for a digital system based on a binary differential evolution algorithm', in *Proceedings of the 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC)*, IEEE, pp.92–97.

Mirjalili, S., Mirjalili, S.M. and Yang, X.S. (2013) 'Binary bat algorithm', *Neural Computing and Applications*, Vol. 25, No. 3, pp.663–681.

Moraglio, A. and Silva, S. (2010) 'Geometric differential evolution on the space of genetic programs', in *Genetic Programming*, Vol. 6021, pp.171–183, Springer.

Moraglio, A., Togelius, J. and Silva, S. (2013) 'Geometric differential evolution for combinatorial and programs spaces', *Evol. Comput.*, Vol. 21, No. 4, pp.591–624.

Nakamura, R.Y.M., Pereira, L.A.M., Costa, K.A., Rodrigues, D., Papa, J.P. and Yang, X.S. (2012) 'BBA: a binary bat algorithm for feature selection', in *Proceedings of the 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, IEEE, pp.291–297.

Nguyen, T.T., Shieh, C.S., Horng, M.F., Ngo, T.G. and Dao, T.K. (2015) 'Unequal clustering formation based on bat algorithm for wireless sensor networks', in *Proceedings of the 6th International Conference on Knowledge and Systems Engineering (KSE)*, Springer, Vol. 326, pp.667–678.

Pampara, G. and Engelbrecht, A.P. (2011) 'Binary artificial bee colony optimization', in *Proceedings of the IEEE Symposium on Swarm Intelligence (SIS)*, IEEE, 11–15 April, pp.1–8.

Pampara, G., Engelbrecht, A.P. and Franken, N. (2006) 'Binary differential evolution', in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, IEEE, pp.1873–1879.

Priem-Mendes, S., Gomez-Pulido, J.A., Vega-Rodriguez, M.A., Sanchez-Perez, J.M., Saez, Y. and Isasi, P. (2009a) 'The radio network design optimization problem benchmarking and state-of-the-art solvers', *Studies in Computational Intelligence*, Vol. 210, Springer.

Priem-Mendes, S., Molina, G., Vega-Rodriguez, M.A., Pulido, J.A.G., Saez, Y., Miranda, G., Segura, C., Alba, E., Isasi, P., Leon, C. and Sanchez-Perez, J.M. (2009b) 'Benchmarking a wide spectrum of metaheuristic techniques for the radio network design problem', *Evolutionary Computation*, Vol. 13, No. 5, pp.1133–1150.

Rafiei, M. and Zadeh, M.S. (2012) 'Optimization of a typical biomass fuelled power plant using genetic algorithm and binary particle swarm optimization', in *Proceedings of 17th Conference on Electrical Power Distribution Networks (EPDC)*, IEEE, pp.1–10.

Ramos, C.C.O., de Souza, A.N., Chiachia, G., Falcão, A.X. and Papa, J.P. (2011) 'A novel algorithm for feature selection using harmony search and its application for non-technical losses detection', *Computers & Electrical Engineering*, Vol. 37, No. 6, pp.886–894.

Rashedi, E., Nezamabadi-pour, H. and Saryazdi, S. (2010) 'BGSA: binary gravitational search algorithm', *Natural Computing*, Vol. 9, No. 3, pp.727–745.

Razavi, S.M. (2011) *Tracking Area Planning in Cellular Networks*, PhD thesis, Department of Science and Technology Linkoping University Norrkoping, Sweden.

Rodrigues, D., Pereira, L.A.M., Almeida, T.N.S., Papa, J.P., de Souza, A.N., Ramos, C.C.O. and Yang, X.S. (2013) 'BCS: a binary cuckoo search algorithm for feature selection', in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, pp.465–468.

Sambariya, D.K., Gupta, R. and Prasad, R. (2016) 'Design of optimal input-output scaling factors based fuzzy PSS using bat algorithm', *Engineering Science and Technology, An International Journal*, No. 2.

San Chua, P., Salleh, A.H.M., Mohd Saberi, M., Safaai, D., Sigeru, O. and Michifumi, Y. (2015) 'Identifying a gene knockout strategy using a hybrid of the bat algorithm and flux balance analysis to enhance the production of succinate and lactate in Escherichia coli', *Biotechnology and Bioprocess Engineering*, Vol. 20, No. 2, pp.349–357.

Severino, A.G.V., Linhares, L.L.S. and de Araujo, F.M.U. (2015) 'Optimal design of digital low pass finite impulse response filter using particle swarm optimization and bat algorithm', in *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, IEEE, Vol. 1, pp.207–214.

Thafasal Ijyas, V.P. and Sameer, S.M. (2015) 'A reduced complexity bat algorithm for joint CFO estimation in OFDMA uplink', in *Proceedings of the IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, IEEE, pp.1–5.

Tsai, P.W., Pan, J.S., Liao, B.Y., Tsai, M.J. and Istanda, V. (2012) 'Bat algorithm inspired algorithm for solving numerical optimization problems', *Applied Mechanics and Materials*, Vols. 148–149, pp.134–137.

Wang, X. and Guo, P. (2012) 'A novel binary adaptive differential evolution algorithm for Bayesian network learning', in *Proceedings of the 8th International Conference on Natural Computation (ICNC)*, IEEE, pp.608–612.

Yang, X.S. (2010) 'A new metaheuristic bat-inspired algorithm', in *Nature Inspired Cooperative Strategies for Optimization (NICSO)*, Vol. 284, pp.65–74, Springer.

Yang, X.S. (2011) 'Bat algorithm for multiobjective optimisation', *Int. J. Bio-Inspired Comput*., Vol. 3, No. 5, pp.267–274.

Yang, X.S. (2012) 'Metaheuristic optimization with applications: Demonstration via bat algorithm', in *Proceedings of 5th Bio-inspired Optimization Methods and Their Applications (BIOMA)*, pp.23–34.

Yang, X.S. and Gandomi, A.H. (2012) 'Bat algorithm: a novel approach for global engineering optimization', *Engineering Computations*, Vol. 29, No. 5, pp.464–483.

Yang, X.S. and He, X. (2013) 'Bat algorithm: literature review and applications', *Int. J. Bio-Inspired Comput*., Vol. 5, No. 3, pp.141–149.