Binary Bat Algorithm: On The Efficiency of Mapping Functions When Handling Binary Problems Using Continuous-variable-based Metaheuristics

Zakaria Dahi, Chaker Mezioud, Amer Draa

Binary Bat Algorithm:
On The Efficiency of Mapping Functions When Handling Binary Problems Using Continuous-variable-based Metaheuristics

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

Modeling and Implementation of Complex Systems laboratory
Dept. of New Technologies of Information and Communication
Constantine 2 university
Constantine City, Algeria
{zakaria.dahi,chaker.mezioud}@univ-constantine2.dz
{draa_amer@yahoo.fr}

Abstract. Global optimisation plays a critical role in today's scientific and industrial fields. Optimisation problems are either continuous or combinatorial depending on the nature of the parameters to optimise. In the class of combinatorial problems, we find a sub-category which is the binary optimisation problems. Due to the complex nature of optimisation problems, exhaustive search-based methods are no longer a good choice. So, metaheuristics are more and more being opted in order to solve such problems. Some of them were designed originally to handle binary problems, whereas others need an adaptation to acquire this capacity. One of the principal adaptation schema is the use of a mapping function to decode real-valued solutions into binary-valued ones. The Antenna Positioning Problem (APP) is an NP-hard binary optimisation problem in cellular phone networks (2G, EDGE, GPRS, 3G, 3G$^+$, LTE, 4G). In this paper, the efficiency of the principal mapping functions existing in the literature is investigated through the proposition of five binary variants of one of the most recent metaheuristic called the Bat Algorithm (BA). The proposed binary variants are evaluated on the APP, and have been tested on a set of well-known benchmarks and given promising results.

1 Introduction

Combinatorial problems are problems whose parameters belong to a finit set of integers ($x_i \epsilon \mathbb{N}$). The latter includes a more specific type called *binary optimisation problems* : problems whose parameters can take values from a bi-valued search space called *genotype space* ($x_i \epsilon \{1, 0\}$).

The design of cellular phone networks (2G, EDGE, GPRS, 3G, 3G$^+$, LTE, 4G) is one of the most critical tasks during the network implantation. Any design process that can not deal with this phase may alter the service quality

of the network itself. The *Antenna Positioning Problem* (*APP*) is one of the most challenging optimisation issues in the design phase of cellular networks. The APP is formulated as a binary optimisation problem and was proven to be NP-hard.

Metaheuristics are efficient tools to use when tackling such optimisation problems. Regardless to the source of their inspiration, metaheuristics can be divided into algorithms who are originally designed to tackle continuous problems, and those who are designed to tackle combinatorial ones.

The *Bat Algorithm* (*BA*), is one of the recently proposed metaheuristics [14]. It was inspired by the natural phenomenon of echolocation used by bats. The BA was originally designed to tackle optimisation problems within continuous search space and it has shown encouraging performances.

Generally, when adapting a *continuous-variable-based metaheuristic* (i.e. *a metaheuristic that was designed originally to operate on variables within continuous search space*) to tackle binary problems, many schemas of adaptation exist. One of the most opted one, is the use of a mapping function to map the real-valued solutions into binary ones. Several sub-schemas of mapping exist as well in this last one : *one-to-one*, *many-to-one*, *one-to-many*.

Many questions still surround this schema of adaptation, such as the fact that the efficiency of these mapping functions is still fuzzy and unexplored. In addition, no clear statement exist on whether using *binary-variable-based metaheuristics* (i.e. *metaheuristics that were designed originally to operate on variables within binary-valued search space*) or continuous ones to tackle binary optimisation problems. Does the efficiency of these mapping functions depends on the algorithm used or the problem solved. Finally, no affined study shows if it is worth using this mapping functions and ultimately which kind of metaheuristics is more efficient when solving binary problems.

Through analysing the literature, the five principal mapping functions existing are used to propose new binary variants of the *Bat Algorithm*. The mapping functions used in this work were selected to illustrate different schemas of mapping. These functions are : The *Nearest Integer method* (*NI*), the *normalisation technique*, the *Angle Modulation method* (*AM*), the *Great Value Priority method* (*GVP*) and finally, the *Sigmoid Function* (*SF*). The proposed binary variants of the BA were tested on the APP using well-known benchmark instances, with different sizes and complexity as well. In addition, the last ones were compared to one of the most used binary-variable based algorithm : the *Genetic Algorithm*.

The remainder of this paper is structured as follows. In Section 2, we introduce basic concepts related to the antenna positioning problem, the bat algorithm, and the mapping functions used in this work. In Section 3, we introduce the proposed binary variants of the bat algorithm. Section 4 is dedicated to experimental results, their interpretation and disscussion. Finally, we present the conclusion of our work in Section 5.

2 Basic Concepts

In this Section, we introduce basic concepts related to the antenna positioning problem, the bat algorithm, and the used mapping functions.

2.1 Antenna Positioning Problem

In this section, we present a widely used formulation of the APP, that was given by Guidec et. al. [3]. This modeling of the antenna positioning problem consider two objectives : maximizing the covered area while minimizing the number of base station used.

The antenna positioning problem recalls NP-hard problems in graph theory such as the Minimum Dominating Set (MDS), the Maximum Independent Set (MIS), or the Unicast Set Covering Problem (USCP), see Sub-figures (a), (b), (c) of Figure 2.

Let L be the set of all potentially covered areas and M the set of all potential locations of base stations. Suppose G is a graph where E is the set of edges in the graph verifying that each transmitter is linked to the area that it covers. One seeks for the minimum subset of transmitters that covers the maximum surface of a given area. In other words, the objective is to find a subset M' such that $|M'|$ is minimised and $|\ Neighbours(M', E)\ |$ is maximised [4]; where $Neighbours$ represents the set of the covered area, and M' represents the set of transmitters used to cover this area.

$$|\ Neighbours(M', E)| \ = \ \{\, u \,\epsilon\, L \,|\, \exists\, v \,\epsilon\, M' \,, (u, v) \,\epsilon\, E \,\} \tag{1}$$

$$M' \ = \ \{\, t \,\epsilon\, B \,|\, x_t \ = \ 1 \,\} \tag{2}$$

A Base Tranceiver Station (BTS) is a radio transmitting device with a specific type of coverage (See Figure 1). In this work, we used three types of coverage introduced in [1]. A cell is a part of a geographical area that is covered by a base station.

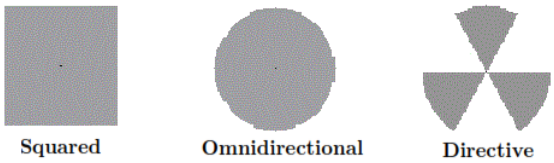

Fig. 1. Antenna Coverage Models

The working area is discretized in a rectangular grid with Dim_x and Dim_y dimensions. Having $Sites = \{site_1 , site_2 , site_3 , \dots , site_N \}$ is the set of potentially preselected sites, where the antennas can be placed. Each potential site location is identified by Cartesian coordinates $\{ site_1 = (x_1 , y_1) , site_2 = (x_2 , y_2) , site_3 = (x_3 , y_3) , \dots , site_N = (x_N , y_N) \}$, see Sub-figures (d), (e), (f) of Figure 2.

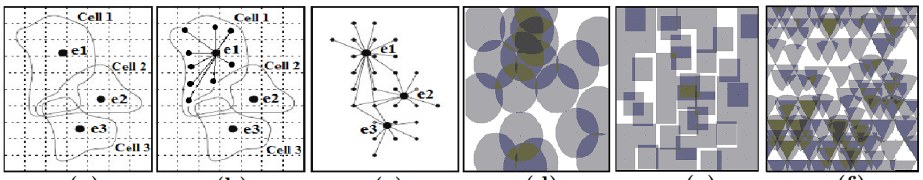

Fig. 2. Representation of The Discretized Area

A potential solution of the APP can be a binary vector described as follows. Each vector $\overrightarrow{X}$ represents a potential configuration of the mobile network. The number of elements of each vector represents the number of potential condidate sites. The rank of each dimsenion represents the rank of the corresponding base station $i = 1 , 2 , \dots , Dimension_{\overrightarrow{x}}$. Each dimension of the vector is strictly binary valued : $x_i \epsilon \overrightarrow{X} / x_i = 1 \vee x_i = 0$. If $x_i = 1$, the i^{th} base station is selected, otherwise it is discarded. The objective function to optimize is defined by the formula 3.

$$Maximize : \quad f(x) = \frac{Cover\ ratio^{\alpha}}{Number\ of\ used\ base\ station} \tag{3}$$

with :

$$Number\ of\ used\ base\ station = \sum_{i=1}^{Dimension_{\overrightarrow{x}}} x_i \ , \tag{4}$$

with :

$$cover\ ratio = \left(\frac{Covered\ area}{Total\ area} \right) * 100 \tag{5}$$

And :

$$Covered\ area = \sum_{i=1}^{Dim_x} \sum_{j=1}^{Dim_y} cover(i,j) \ , \tag{6}$$

And :

$$Total\ area = Dim_x * Dim_y. \tag{7}$$

It is worth to mention that other mathematical models of the antenna positioning problem exist like the one proposed in [12]. Generally, these models differ by their mathematical formulations or modeling.

2.2 Bat Algorithm

The Bat Algorithm has been recently proposed. It is a swarm-based metaheuristic [14]. This algorithm is inspired by the natural echolocation behaviour of bats. Microbats use a type of sonar, called echolocation, to detect their preys, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. The bats then adjust the pulse and rate of the sound as they get closer to the obstacles or the prey. This phenomenon has been translated into the newly proposed bat algorithm. The pseudo-code of Algorithm 1 describes the general framework of the bat algorithm.

Algorithm 1 . The Bat Algorithm

Objective function $f(x), x = (x_1, ..., x_d)^T$
Initialize the bat population X_i (i = 1, 2, ..., n) and V_i
Define the pulse rate r_i and the loudness A_i
Input : Initial bat population
while ($t <$ Max number of iterations) **do**
    Generate new solutions by adjusting frequency, and updating velocities and locations/solutions (Equations 8 to 10)
    if ($rand > r_i$) **then**
        Select a solution among the best solutions
        Generate a local solution around the selected best solution
    end if
    Generate a new solution by flying randomly
    if ($rand < A_i \,\&\, f(X_i) < f(X_*)$) **then**
        Accept the new solutions
        Increase r_i and reduce A_i
    end if
    Rank the bats and find the current best X_*
end while
Output : Best bat found (i.e. best solution)

Equations 8, 9 and 10 define how the position X_i and velocity V_i in a d-dimensional search space are updated. The new solution X_i^t and velocity V_i^t at a time step t are given by :

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{8}$$

$$V_i^t = V_i^{t-1} + (X_i^t - X_*)f_i \tag{9}$$

$$X_i^t = X_i^{t-1} + V_i^t \tag{10}$$

Initially, each bat is randomly assigned a frequency which is drawn randomly from $[f_{min}\,, f_{max}]$. $\beta\,\epsilon\,[0,1]$ is a random vector drawn from a uniform distribution. X_* is he current global best location (solution) which is located after comparing all the solutions.

When a local search is performed a solution is selected among the current best solutions. A new solution for each bat is generated using a random walk as described in Equation 11; Where ϵ is a random number from $[-1,1]$, and $A^t =< A_i^t >$ is the average loudness of all bats at this time step.

$$X_{new} = X_{old} + \epsilon A^t \tag{11}$$

Likewise, the loudness A_i and the rate ri of pulse are updated once the new solution is accepted. This is done using Formulas 12 and 13, where α and γ are constants. Initially, each bat is randomly assigned a loudness and a rate drawn respectively from the intervals $[A_{min}\,, A_{max}]$ and $[r_{min}\,, r_{max}]$.

$$A_i^{t+1} = \alpha A_i t \tag{12}$$

$$r_i^{t+1} = r_i^0[1 - exp(-\gamma t)] \tag{13}$$

2.3 From Phenotype to Genotype Space

Several approaches exist for adapting a continuous-variable-based metaheuristic to work also in binary search space. The first schema of adaptation consists in replacing arithmetic operators of the metaheuristic by logical ones to operate directly on the binary solutions. The second aims to find the corresponding operators of the algorithms in the geometric space (*Hamming space*). Finally, the third approach consists in conserving the original operators, architecture and solution representation of the algorithm, and adding a complementary module that maps real-valued solutions into binary ones. The techniques used in this third category are generally known also as *mapping functions*. Several schemas of mapping exist. The pincipal ones are : *one-to-one*, *many-to-one*, and *one-to-many*.

The mapping functions used in this work have been chosen to illustrate several mapping schemas and several mathematical properties. In the following, the mapping functions used in this study are introduced.

Nearest Integer (NI) : This technique consists of assigning a real number to the nearest integer by rounding or truncating it up or down [2, 6].

Normalisation : This approach was proposed in [9, 10]. It consists of the normalisation of the solution by linearaly scaling it using the Formula 14. Then the condition 15 is applied to get the corresponding binary solution.

$$x_{ij}^{'} = \frac{(\, x_{ij} \, + \, x_i^{min} \,)}{(\, |\, x_i^{min} \,| \, + \, x_i^{max} \,)} \tag{14}$$

$$x_{ij} = \begin{cases} 1\,, & If \; x_{ij}^{'} \geqslant 0.5 \\ 0\,, & Otherwise \end{cases} \tag{15}$$

Assuming that $i = 1 \cdots N$ and $j = 1 \cdots D$. Where : N is the population size, D is the size of the solution vector. x_i^{min} and x_i^{max} are respectively the minimum and the maximum values in the i^{th} vector at the iteration t.

Angle Modulation (AM) : The idea is to use a trigonometric function to map real-valued solutions into binary ones. The generator function is used for signal processing in telecommunications and defined as follows [11,13].

$$g(x_{ij}) \, = \, sin\,(\, 2\Pi\,(\, x_{ij} \, - \, a) \, * \, b \, * \, cos\,(\, 2\Pi\,(\, x_{ij} \, - \, a) \, * \, c\,)\,) \, + \, d \tag{16}$$

Where $i = 1 \cdots N$ and $j = 1 \cdots D$. N is the population size, D is the size of the solution vector. $g(x)$ is the generator function, and x_{ij} is a single element from a potential solution vector. Instead of optimizing a D-dimensional binary string solution, the search space is reduced to a 4-dimensional search space. Each vector of solution $\overrightarrow{G}$ represents potential values of the coefficients (a, b, c, d) in the generator function. At each iteration, every solution vector is applied to a sample vector $\overrightarrow{X}$ with the original D-dimensions of the problem. The sample vector is drawn from a uniform distribution and has equaly spaced intervals between each dimension and another. Finally, one has to apply the following formula on the resulting vector :

$$x_{ij} = \begin{cases} 1\,, & If \; g(x_{ij}) \geqslant 0 \\ 0\,, & Otherwise \end{cases} \tag{17}$$

Sigmoid Function (SF) : In this technique, each real valued dimension of the solution vector is mapped into a strictly binary valued one [7,8]. The probability of each dimension to flip to one state or another is computed according to the real value of the dimension itself by applying Formula 18.

$$x_{ij} = \begin{cases} 1\,, & If \; Rand[0,1] \leq \frac{1}{1+e^{x_{ij}}} \\ 0\,, & Otherwise \end{cases} \tag{18}$$

Where $i = 1 \cdots N$ and $j = 1 \cdots D$. N is the population size, D is the size of the solution vector, and $Rand[0,1]$ is a randomly generated positive number, drawn from a uniform distribution in the interval $[\,0\,,1\,]$.

Great Value Priority (GVP) : Recently, authors in [5] have introduced this technique. Starting from a given real valued solution vector $\overrightarrow{X}$, a permutation vector $\overrightarrow{P}$ is created. The first element of the permutation vector p_1 will contain the position of the largest element in the original vector, the second element of the permutation vector p_2 will receive the position of the second largest element of the real valued vector, and so on. The procedure will be repeated until all the elements of the original vector are browsed. Finally, having the permutation vector $\overrightarrow{P}$, the following formula will be applied to recover back a binary valued vector.

$$x_{ij} = \begin{cases} 1\,, & If\ p_j\ > p_{j+1} \\ 0\,, & Otherwise \end{cases} \tag{19}$$

3 The proposed Binary Bat Algorithm

The inclusion of the discretisation step using one of the mapping functions after line code 10 in pseudo-code of Algorithm 1 results in giving birth to new variants of the bat algorithm. The first variant, using the nearest ineteger method as a discretising technique is called NI-BBA (for Nearest Integer based Binary Bat Algorithm). The second variant is called N-BBA (for Normalisation based Binary Bat Algorithm) is based on the normalisation method. The third variant uses the sigmoid function and is called SF-BBA (for Sigmoid Function based Binary Bat Algorithm). The fourth variant is the AM-BBA (for Angle Modulation based Binary Bat Algorithm) is based on the angle modulation method. Finally, GVP-BBA (for Great Value Priority based Binary Bat Algorithm) uses the great value priority technique.

4 Experimental Results and Discussion

The experiments where carried using an Intel I3 core with 2 GB Ram and a Windows 7 OS. The implementation was done using Matlab 7.12.0 (R2011a).

Two scenarios were randomly generated. Both are representing a working area of 20.25 Km^2. The first instance contains 549 available locations, whereas the second instance contains 749 available locations. Other instances of 149 and 349 preselected positions, are used here. They were provided by the university of Malaga, Spain. We used three types of coverage : squared, omnidirectional and directive [1]. It is worth to mention that directive antennas cover one sixth of the area of omnidirectionnal antennas having the same radius. Table 1 shows all the features of the used instances.

The proposed binary variants of the bat algorithm were also compared to one of the most used binary-variable-based metaheuristic and whose the efficiency is well established : the canonical Genetic Algorithm (GA). The GA uses a wheel selection and a two-point crossover with a probability equal to 0.7 and a bit-flip

Table 1. Instances : Size and Coverage

Instance Type	Grid Dimension	Instance	Coverage	Radius
Synthetic	287 x 287	149	Omnidirectional	22
			Squared	20
			Directive	22
		349	Omnidirectional	22
			Squared	20
			Directive	22
Random	300 x 300	549	Omnidirectional	26
			Squared	24
			Directive	26
		749	Omnidirectional	26
			Squared	24
			Directive	26

mutation with a probability of 0.05. The percentage of chromosomes used to create the matting pool is 50 %. The parameters of the bat algorithm used in this experiment are shown in Table 2.

Table 2. Bat Algorithm Parameters

Prameter	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 experiments were performed till reaching 20.000 evaluations, and each one is repeated for 20 runs. Several results are reported such as : the *best* and the *worst* fitness, and also the *mean* and *standard deviation* of fitness value over 20 runs.

Tables 3, 4, 5 and 6 show the results obtained when evaluating the five binary variants of the bat algorithm using the instances 149, 349, 549, 749 and this for each type of coverage : squared, omnidirectional and directive.

Based on the results shown in Tables 3, 4, 5 and 6 many observations can be made. The performances of the variants for small instances (149, 349) are close, but one can note that as the instance size increases (549, 749), the difference in the efficiency of the variants is more obvious. So, for some variants like the AM-BBA, N-BBA, GVP-BBA their efficiency depends highly on the size of the problem treated. Whereas for other variants such as the NI-BBA and the SF-BBA the efficiency is maintained even if the size of problem increases.

In general, NI-BBA and SF-BBA variants has shown better results than the other variants when solving the APP, especially the NI-BBA. The scalability of both variants is similar since both succeeded to solve the different sizes of

Table 3. Results of the Bat Algorithm Variants For Instance 149

Instance	Coverage	Algorithm	Best	Worst	Mean	Std
149	Squared	NI-BBA	120.582	120.582	**120.582**	1.458E-14
		N-BBA	106.361	95.006	100.501	2.70580121
		AM-BBA	111.120	97.564	104.250	4.63919122
		SF-BBA	103.012	102.112	102.157	0.20130794
		GVP-BBA	113.548	113.400	113.489	0.0745577
		GA	110.495	99.044	104.332	3.60183932
	Circle	NI-BBA	97.701	97.701	97.701	2.916E-14
		N-BBA	94.310	87.932	90.455	1.7646704
		AM-BBA	99.283	85.004	90.940	3.79249425
		SF-BBA	100.366	100.366	**100.366**	2.916E-14
		GVP-BBA	98.747	98.747	98.747	2.916E-14
		GA	97.282	85.472	90.832	2.99547967
	Directive	NI-BBA	41.473	41.473	41.473	1.458E-14
		N-BBA	40.354	37.315	38.905	0.81358109
		AM-BBA	42.560	40.963	41.594	0.44465264
		SF-BBA	42.388	41.859	**42.362**	0.11845141
		GVP-BBA	40.639	40.639	40.639	0
		GA	41.543	36.904	38.924	1.10100813

Table 4. Results of the Bat Algorithm Variants For Instance 349

Instance	Coverage	Algorithm	Best	Worst	Mean	Std
349	Squared	NI-BBA	95.371	95.371	95.371	2.916E-14
		N-BBA	61.664	58.335	59.981	1.03311284
		AM-BBA	188.758	89.671	**135.391**	30.2152069
		SF-BBA	102.880	98.142	102.643	1.05964286
		GVP-BBA	63.551	62.123	63.479	0.319142
		GA	63.643	57.858	61.196	1.60235842
	Circle	NI-BBA	95.081	95.081	**95.081**	2.916E-14
		N-BBA	60.350	56.624	58.753	0.98804881
		AM-BBA	127.577	75.920	88.332	13.1198896
		SF-BBA	90.144	88.803	88.911	0.3445047
		GVP-BBA	61.270	60.903	61.251	0.08199193
		GA	62.199	57.224	59.614	1.48976652
	Directive	NI-BBA	41.464	41.464	**41.464**	7.29E-15
		N-BBA	39.659	37.102	38.221	0.54525135
		AM-BBA	42.283	38.420	39.408	0.91636124
		SF-BBA	39.899	39.899	39.899	7.29E-15
		GVP-BBA	39.723	39.723	39.723	7.29E-15
		GA	39.450	37.199	38.304	0.66880293

the problem. No clear conclusion can be made about how the proposed variants behave when dealing with a specific type of coverage (squared, omnidirectional, directive), or a specific type of data (random, synthetic). One can note also that all the binary variants of the BA were able to outperform the results obtained by the canonical GA for all the instances and for all the sizes of the instances. But one can note also that the difference between the GA and the other variants decreases as the size of the instance decreases.

The conclusion that can be made concerning the impact of the mapping functions on the efficiency of an algorithm, is that the adequate use of a mapping function depends in some cases on the size of the problem engaged and in other cases on the type of the problem. Furthermore, one can note that in reality the bat algorithm do not need complex mapping functions since the basic *rounding function* has shown better results than the other complex mapping functions.

Table 5. Results of the Bat Algorithm Variants For Instance 549

Instance	Coverage	Algorithm	Best	Worst	Mean	Std
549	Squared	NI-BBA	139.973	139.973	139.973	2.916E-14
		N-BBA	40.668	38.939	40.063	0.46911297
		AM-BBA	134.322	96.348	112.442	8.45490438
		SF-BBA	147.445	147.445	**147.445**	2.916E-14
		GVP-BBA	40.529	40.365	40.521	0.03669093
		GA	42.777	38.927	41.070	1.19535659
	Circle	NI-BBA	127.289	126.025	**126.910**	0.59402311
		N-BBA	41.311	38.693	39.740	0.77547919
		AM-BBA	116.780	89.794	102.404	6.32663866
		SF-BBA	117.067	116.067	117.017	0.223514
		GVP-BBA	41.411	41.411	41.411	7.29E-15
		GA	42.640	39.027	40.620	1.06845162
	Directive	NI-BBA	49.046	49.046	**49.046**	1.458E-14
		N-BBA	36.371	34.536	35.468	0.4751343
		AM-BBA	52.156	45.135	48.721	1.83706437
		SF-BBA	51.874	51.808	51.870	0.01464278
		GVP-BBA	35.853	35.814	35.852	0.00873771
		GA	37.913	34.444	35.851	0.89470107

Table 6. Results of the Bat Algorithm Variants For Instance 749

Instance	Coverage	Algorithm	Best	Worst	Mean	Std
749	Squared	NI-BBA	135.888	135.888	**135.888**	2.92E-14
		N-BBA	29.847	28.486	29.175	0.40036354
		AM-BBA	126.827	90.586	109.403	11.4041269
		SF-BBA	130.915	130.915	130.915	0
		GVP-BBA	29.586	29.586	29.586	3.65E-15
		GA	30.788	28.477	29.437	0.52843974
	Circle	NI-BBA	101.870	101.870	101.870	1.46E-14
		N-BBA	29.917	28.275	29.018	0.45477513
		AM-BBA	110.941	87.658	97.607	6.66552368
		SF-BBA	114.077	114.077	**114.077**	0
		GVP-BBA	29.367	29.142	29.356	0.05020659
		GA	30.579	27.929	29.198	0.82131334
	Directive	NI-BBA	50.405	50.024	50.386	0.08519864
		N-BBA	28.674	27.113	27.730	0.4651209
		AM-BBA	50.920	45.767	48.605	1.36744246
		SF-BBA	51.189	49.877	**51.123**	0.29334724
		GVP-BBA	27.578	27.415	27.503	0.02651541
		GA	29.608	26.802	27.785	0.66481778

5 Conclusion

In this paper we conducted a comparative study on the impact of mapping functions on the efficiency of the continuous-variable-based metaheuristic. This was done by proposing five new variants of a recent metaheuristic which is the Bat Algorithm (BA). The proposed binary variants, were tested on an NP-hard optimisation problem in cellular phone networks which is the Antenna Positioning Problem (APP). The results showed that the impact of such mapping functions on the efficiency of an algorithm depends on two factors : the size of the problem, or the complexity of the problem. The best mapping functions found for the bat algorithm are the nearest integer and sigmoid function techniques.

This work illustrates a simple comparative study, and no deep and general conclusion can be made about the efficiency of the mapping functions, the con-

trolling factor of these last ones or the usefulness of these mapping functions when solving binary problems. So, we seek to conduct a more deep statistical comparative study using several continuous-variable-based metaheuristics and compare them with more powerful binary-variable-based metaheuristics.

References

1. Alba, E., Molina, G., Chicano, F.: Optimal placement of antennae using metaheuristics. In: Proceedings of the 6th International Conference on Numerical Methods and Applications, (NMA). pp. 214–222 (2006)
2. Burnwal, S., Deb, S.: Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. The International Journal of Advanced Manufacturing Technology 64(5-8), 951–959 (2013)
3. Calegari, P., Guidec, F., Kuonen, P.: A parallel genetic approach to transceiver placement optimisation. In: Proceedings of the SIPAR Workshop : Parallel and Distributed Systems. pp. 21–24 (1996)
4. Calegari, P., Guidec, F., Kuonen, P., Kobler, D.: Parallel island-based genetic algorithm for radio network design. J. Parallel Distrib. Comput. 47(1), 86–90 (1997)
5. Congying, L., Huanping, Z., Xinfeng, Y.: Particle swarm optimization algorithm for quadratic assignment problem. In: Proceedings of the International Conference on Computer Science and Network Technology, (ICCSNT). vol. 3, pp. 1728–1731 (2011)
6. Costa, M., Rocha, A.A.M., Francisco, B.R., Fernandes, M.E.: Heuristic-based firefly algorithm for bound constrained nonlinear binary optimization. Advances in Operations Research 1(215182), 12 (2014)
7. Liu, Q., Lu, W., Xu, W.: Spectrum allocation optimization for cognitive radio networks using binary firefly algorithm. In: Proceedings of the International Conference on Innovative Design and Manufacturing, (ICIDM). pp. 257 – 262 (2014)
8. Palit, S., Sinha, S., Molla, M., Khanra, A.: A cryptanalytic attack on the knapsack cryptosystem using binary firefly algorithm. In: Proceedings of the 2nd International Conference on Computer and Communication Technology, (ICCCT). pp. 428 – 432 (2011)
9. Pampara, G., Engelbrecht, A.: Binary artificial bee colony optimization. In: Proceedings of the IEEE Symposium on Swarm Intelligence, (SIS). pp. 1 – 8 (11-15 April 2011)
10. Pampara, G., Engelbrecht, A., Franken, N.: Binary differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, (CEC'06). pp. 1873–1879 (2006)
11. Pampara, G., Franken, N., Engelbrecht, A.: Combining particle swarm optimisation with angle modulation to solve binary problems. In: Proceedings of the Congress on Evolutionary Computation, (CEC'05). pp. 89–96 (2005)
12. Segura, C., Segredo, E., González, Y., León, C.: Multiobjectivisation of the antenna positioning problem. In: International Symposium on Distributed Computing and Artificial Intelligence, (DCAI). pp. 319–327 (2011)
13. Swagatam, D., Rohan, M., Rupam, K., Thanos, V.: Multi-user detection in multicarrier CDMA wireless broadband system using a binary adaptive differential evolution algorithm. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, (GECCO '13). pp. 1245–1252 (2013)
14. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization, (NICSO), vol. 284, pp. 65–74. springer (2010)