International Journal of Informatics and Applied Mathematics
e-ISSN:2667-6990 Vol. 4, No. 2, 1-16

A Discrete Bat Algorithm for the Examination Timetabling Problem

Egbert Mujuni

Mathematics Department, University of Dar es Salaam,
Box 35062 Dar es Salaam, Tanzania
emujuni@udsm.ac.tz, egbertmujuni2016@gmail.com

Abstract. Bat Algorithm (BA) is one of the newest and promising nature inspired metaheuristics. Introduced by Yang in 2010, BA is a population method which is based on the echolocation characteristics of microbats. The original BA was proposed only for continuous optimization problems. Different approaches that use BA as basis for solving discrete optimization problem have been proposed. In this paper, a discrete bat algorithm version has been developed to solve the examination timetabling problem. Empirical study of the proposed algorithm was carried using data from the University of Dar es Salaam. The proposed algorithm demonstrated higher performance in comparison to a well known metaheuristic, Tabu Seach (TS).

Keywords: Combinatorial Optimizations · Examination Timetabling Problems · Metaheuristics

1 Introduction

Combinatorial Optimization (CO) is a field of studies that is focused on optimization problems with a finite set of solutions. CO has numerous important real life applications such as resource allocations, network analysis, academic timetabling, machine learning, communication and routing in networks, production planning and artificial intelligence [24, 39]. Unfortunately, most combinatorial optimization problems are NP-hard (see, e.g., [36]). Thus, no any polynomial time algorithm is known to solve them, and it is very unlikely such algorithms exist, unless $P = NP$ [23]. To complicate the matter further, many cases of CO problems from real-life applications come with large size instances, several objectives and many constraints. Thus, their exact solutions cannot be computed with reasonable amount of time. For these reasons, many heuristic, in particular metaheuristic, methods have been developed for CO problems. Heuristics and metaheuristics seek to find solutions fast without a guarantee that the obtained solution is optimal.

Metaheuristic algorithms are considered as higher-level heuristic methodologies that combine several lower-level heuristic processes in solving specific optimization problems (see, e.g., [12]). The key components of metaheuristic algorithms are diversification and intensification [6, 31]. Diversification, also called exploration, ensures that the algorithm to explore, often randomly, many and different regions of the search space. The main idea behind the diversification component is to ensure that the system does not get trapped into local solutions, and thus generate new solutions as widely as possible. Intensification, also called exploitation, is the ability to obtain high quality solution within the explored regions.

Many metaheuristic algorithms have been developed over the last two decades, and many of them are inspired by various phenomena of nature [39]. Examples of such algorithms include: Ant Colony Optimization (ACO) was inspired by the behaviour of ants foraging for food. Artificial Immune Systems (AIS) mimics biological immune systems for optimization. Bacterial Foraging Algorithm (BFA) simulates search and optimal foraging of bacteria. Cat Swarm Algorithm (CSA) simulates the behaviour of cats. Cuckoo Optimization Algorithm (COA) mimics reproduction strategy of cuckoos. Fish School Search (FSS) simulates gregarious behaviour of oceanic fish. Genetic Algorithm (GA) simulates Darwinian evolution concepts. Invasive Weed Optimization (IWO) mimics the ecological behaviour of colonizing weeds. Marriage in Honey Bee Optimization Algorithm (MBO) is based on the processes of reproduction in the honey bee colony. Monkey Search (MS) mimics a monkey in search for food resources. Particle Swarm Optimization (PSO) simulates the social behaviour of a flock of migrating birds trying to reach an unknown destination. Simulated Annealing (SA) is based on the annealing process of metals. The reader is referred to [31] and [38] for detailed information on the nature inspired metaheuristic algorithms.

One of the most promising recently introduced nature inspired metaheuristics is Bat Algorithm (BA). Proposed by Yang [37], BA is based on the echolocation behaviour of microbats. Bats are the only mammals with wings and they have

capability of echolocation, which they can use in finding preys and discriminating different types of objects even in complete darkness by varying pulse rates of emission and loudness [37]. BA uses communication between individuals in the bat population to search solutions. One of the main advantages of BA is that it can balance between the diversification and intensification strategies by adjusting parameters when the global optimality is approaching.

Since its inception, BA and its variations have been applied in many different optimization problems from both continuous and discrete fields. A list of the applications includes: Clustering and Data Mining [13, 11], Diagnosis Breast Cancer [10], Fuzzy Logic [32], Image Processing [4], Items Allocation [30], Multiprocessor Scheduling [16], Transport Network Design [29], Vehicle Routing [21, 22]. Induja and Eswaramurthy in [9] and Wanatchapong in [34] provide overviews and applications of BA.

It is well known that heuristics may perform well in some problems but perform poorly in other problems. That is, every meta-heuristic always encounter a problem on which it performs poorly. Wolpert and Mcready [35] proposed a No-Free-Lunch (NFL) theorem, which states "*for any optimization algorithm, any elevated performance over one class of problems is offset by performance over another class*". The theorem implies that the performance of an optimization algorithm always depends on the problem. This means that there is no optimization algorithm that outperforms all other algorithms for all types of problems. This necessitates a need of rigorously testing new developed methods. In this work we investigate usefulness of BA on a well known combinatorial problem called the examination timetabling problem.

The Examination Timetabling Problem (ETP) falls to a large class of timetabling problems, which arise in various forms. This class includes educational timetabling (teaching timetabling and examination timetabling), sports timetabling, nurse scheduling, and transportation timetabling [27]. Burke et al in [3] define timetabling as a problem consisting with four parameters: (i) a finite set of times T, (ii) a finite set of resources R, (iii) a finite set of meetings M, and (iv) a finite set of constraints C. The problem is to assign times and resources to the meetings so as to satisfy the constraints as far as possible. This work focuses on the examination timetabling problem which is defined as an assignment of a set of examinations into a limited number of ordered timeslots (time periods) and rooms of certain capacity in each timeslot, subject to a set of constraints (see, e.g., [19]).

ETP is known to be NP-hard [3, 27] and it has received much attention in the literature because of its importance, from both practical and theoretical point of view. The task of generating high-quality timetable is now becoming more challenging. Institutions are now enrolling more students into a wider variety of degree programmes. In addition, many universities have introduced the concept of cross-faculty, which gives the greater flexibility to students in enrolling courses that they wish to take, besides offering them more options to choose from different faculties. The NP-hardness of ETP implies that there is no effective (polynomial-time) algorithm to solve it. Thus, a large number of researchers have focused on finding heuristic algorithms to solve the problem. This work

aims at investigating performance of BA on the ETP, using real data from the University of Dar es Salaam. One way of assessing the usefulness of a new heuristic is to compare its results with other successful heuristic methods. Examples of such methods include TS [7, 20], SA [15], ACO [2], POS [17] and Prey-Predator Algorithm (PPA) [33] for the case of ETP. Aldeeb et al [1] provide a comprehensive list of heuristic methods used for ETP. In this work, timetables generated by BA and TS are compared. TS has been opted because it was successfully applied in [20] to solve ETP by using real-life dataset which has similar characteristics as of those used in this work.

The remaining part of this paper is organized as follows: The next section gives a formulation of the problem. Here we also give soft and hard constraints. Then we present a section on bat algorithm. In this section original (basic) bat algorithm and modified bat algorithm for ETP are described. In subsequent sections, the results and comparison are presented and discussed. Conclusions are provided in the section before the references.

2 Problem Formulation

As mentioned earlier, examination timetabling problem involves assigning a set of examinations into timeslots and rooms subject to a set of constraints. These constraints are grouped into hard and soft. The hard constraints must be satisfied in order to produce a feasible timetable, whilst the soft constraints should be satisfied as much as possible [19]. Merlot et al in [18] and Mushi in [20] give lists of possible constraints. Examination timetabling problems can be categorized into capacitated and uncapacitated. Their differences is that we consider capacity of each examination room in the first category which is not the case in the other category [2]. In this paper we focus on the uncapacitated case. Instead of considering individual room capacity, we consider the total available sitting capacity in each timeslot. Hard and soft constraints in this paper are similar to those listed in [1], which are:

Hard constraints:

- Each exam must be scheduled in precisely one timeslot.
- No students can have more than one examination at the same timeslot.
- No timeslot with students more than the maximum available sitting capacity in the timeslot.

Soft constraints

- Back-to-back examinations should be avoided as much as possible. This means that the number of examinations with common students which are scheduled in consecutive timeslots is to be minimized.
- Examinations with large number of students (big examinations) should be scheduled as early as possible. The idea is to give examiners enough time for marking.

Sets, Parameter and Variables

In order to write the model formulation we need the following notations for sets, parameters and variables.

***S*ets:**

E**:** Set of examinations.
T**:** Set of timeslots.

***P*arameters:**

R_t**:** Total sitting capacity available in the timeslot $t \in T$.
C_i**:** The number of students in the examination $i \in E$.
$M(i,j)$**:** Collision matrix with $M(i,j) = 1$ if courses i and j have some common students, otherwise $M(i,j) = 0$.

***V*ariables:**

x_{it}**=** 1 if exam i is scheduled at timeslot t, and 0 otherwise.

Mathematical Model Using the above notations, we write integer linear programming model in which constraints represent hard constraints and an objective function constructed in such away that deviations from soft constraint satisfaction are minimized. As mentioned above, this work focuses on two soft constraints. Thus, given a solution s, the objective function has the form:

$$\min f(s) = \lambda_1 f_1(s) + \lambda_2 f_2(s),$$

where f_1 and f_2 are respectively associated with back-to-back and big examinations constraints. A penalty λ_i $(i = 1, 2)$ is given to constraint i to represent the importance of the constraint. Values of λ_1 and λ_2 were determined throng experimental tests.

The following function gives the number of pairs of examinations with common students which are scheduled in consecutive timeslots.

$$f_1(s) = \sum_{t=1}^{|T|-1} \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} x_{it} x_{jt+1} M_{ij}$$

There are different ways of defining functions that give priority to large examinations on the lower-numbered timeslots. For example, Mushi in [20] proposed a function of the form $\frac{t^2}{z}$, where t is the timeslot and z is the size of an examination. However, in this work, a simple function $\frac{tz^2}{p}$, where p is a constant, gives good results. Thus, we set:

$$f_2(s) = \frac{1}{p} \sum_{i=1}^{|E|} \sum_{t=1}^{|T|} t {c_i}^2 x_{it}$$

After performing experiments, we set $p = 1,000,000$.

Therefore, a complete model becomes:

$$\begin{aligned} \text{Min } & f(s) = \lambda_1 f_1(s) + \lambda_2 f_2(s) \\ \text{S.t:} & \sum_{t \in T} x_{it} = 1 && \forall i \in E \\ & \sum_{i,j \in T} x_{it} x_{jt} M_{ij} = 0 && \forall t \in T \\ & \sum_{i \in E} x_{it} c_i \leq R_t && \forall t \in T \end{aligned} \tag{1}$$

The objective cost function minimize deviations from soft constrains satisfactory. The first constraints ensures that each exam is scheduled in precisely one timeslot, the second ensures that examinations with common students are not scheduled in the same timeslot and the third ensures that there is enough room space for each timeslot.

To be able to apply the proposed heuristic efficiently, we transform Model (1) into equivalent model whose objective function is a linear combination of both hard and soft constraints, but hard constraints are penalized much higher than soft constraints. Let

$$D_t = \begin{cases} 1 \text{ if } \sum_{i \in E} x_{it} c_i > R_t \\ 0 \text{ otherwise} \end{cases}$$

The equivalent model is

$$\begin{aligned} \text{Min } & f(s) = M \sum_{t \in T} \Big(\sum_{i,j \in T} x_{it} x_{jt} M_{ij} + D_t \Big) + \lambda_1 f_1(s) + \lambda_2 f_2(s) \\ \text{S.t:} & \sum_{t \in T} x_{it} = 1 \quad \forall i \in E \end{aligned} \tag{2}$$

In solving Model (2), M can be assigned any large number. In our case we set $M = 3000$. Thus, since the expression $\sum_{t \in T} \Big(\sum_{i,j \in T} x_{it} x_{jt} M_{ij} + D_t \Big)$ gives the number of all hard constraints that have been violated, any solution s such that $f(s) < 3000$ must be feasible.

3 Bat Algorithm

3.1 Original Bat Algorithm

As mentioned earlier, BA is a nature inspired optimization algorithm based on the echolocation behaviour of microbats. It mimics the foraging behaviour of microbats when they are hunting and navigating. Microbats emit sound pulse

with different rates and loudness. After the emission of these pulses, they listen to the echoes, and based on the time delay between the emission and detection of the echo behaviour they can locate and identify preys and obstacles [37]. In addition, each bat of the swarm is able to find and move towards a "nutritious" location previously found by the swarm [22].

```
Set Population of Bat P = {x_1, ..., x_n}
Initialize bat x_i, i = 1, ..., n
For each Bat x_i ∈ P do
    Initialize bat x_i, i = 1, ..., n
End for
Repeat
    For each x_i ∈ P do
        Generate new solutions using Equations (3)-(5);
        Generate random number rand
        If (rand > r_i)
            Select one solution among the best ones;
            Generate a local solution around the best one;
        End if
        If (rand < A_i && f(x_i) < f(x_*))
            Accept the new solution
            Increase r_1 and reduce A_i
       End if
       Rank the bats and find the current best x_*
    End for
Until termination criterion is reached
Return the current best bat of the population;
```

Fig. 1. Basic bat algorithm

Yang [37] idealized the following steps of basic BA:

1. All bats use echolocation to sense distance and difference between food and background barriers.
2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency f_i, varying wavelength k and loudness A_0 to search for preys. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their targets.
3. The loudness can vary from a large (positive) A_0 to a minimum constant value A_{min}.

Figure 1 gives a pseudocode of the basic BA.

We assume that a position x_i of bat i of the population represents a candidate solution of the problem to be solved and the objective function to be optimized is already defined. In applying BA, the first step is to initialize all the parameters

related to each bat i. These parameters are: position x_i, velocity v_i and frequency f_i, pulse rate r_i and loudness A_i. The news positions x_i^t and velocities v_i^t at time step t are given by the following equations:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{3}$$
$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right) f_i \tag{4}$$
$$x_i^t = x_i^{t-1} + v_i^t, \tag{5}$$

where the parameter $\beta \in [0, 1]$ is a randomly generated number and x_* denotes the current best solution in the population.

After selecting one solution among the current best solutions, a new solution for each bat is generated locally using random walk [37]:

$$x_{new} = x_{old} + \epsilon \left\langle A^t \right\rangle \tag{6}$$

where a scaling facto $\epsilon \in [-1, 1]$ is a random number and $\langle A^t \rangle$ is the average loudness of bats at the current step. The loudness A_i and the pulse emission rate r_i are updated by using the equations:

$$A_i^{t+1} = \alpha A_i^t \tag{7}$$
$$r_i^{t+1} = r_i^0 \left[1 - exp(-\gamma t)\right] \tag{8}$$

where α and γ are constants.

3.2 Discrete Bat Algorithm for ETP

The original BA was developed to optimize continuous nonlinear functions [28]. This implies that it cannot be used to solve discrete optimization problems directly. Therefore, some modifications of the original BA are needed in order to solve discrete optimization problems. Saji and Riffi [28] suggested to redefine the standard arithmetic operators (addition, subtraction) and the involved parameters. We have adopted this idea in developing Discrete Bat Algorithm (BDA) version for ETP.

The parameters r_i, A_i and f_i have the same form as the basic BA. Similarly, the position x_i of bat i is a possible feasible solution for the examination timetable problem. Below are descriptions of how position x_i and velocity v_i are computed in DBA.

Computation of Position x_i In the basic BA the movement of the bats is updated through the following the Equation (5)

$$x_i^t = x_i^{t-1} + v_i^t.$$

From this formula, the position of a bat i at time step t depends on the velocity v_i and its previous position at time step $t - 1$. This formula cannot be used as it is to solve the ETP. To modify it, we need to define the notion of distance in the discrete space, based on the work of Qui et al. [26].

Definition 1 *A discrete solution space is a pair $(S, \emptyset)$, where S is a set of solutions and $\emptyset$ is an operator which operates on an element in S and generates another one. For a given solution $s \in S$, the set of all solutions that can be obtained by performing one operation $\emptyset$ on s is called to be the neighbourhood set of s, and it is denoted by $N(s)$.*

In solving ETP, an operator $\emptyset$ can use 1-1 and 1-0 moves. The 1-1 move involves with swapping timeslots of two examinations. In the 1-0 move, an examination is removed from its current timeslot and assigned to another. In this work we have used 1-0 move in which both exams and timeslots are randomly selected.

Definition 2 ([26]) *For any solutions s and s', the distance of them, denoted by $s - s'$, is a sequence of least number of consecutive applications of operator that are required to transform s' into s. Difference of two positions is a velocity. That is, a velocity is a sequence of applications of the operator.*

Definition 3 ([26]) *If s is a solution and v is a velocity, the sum of them, $s+v$, is a new position such that $s + v = \emptyset(\ldots\emptyset(\emptyset(s)))$ (v consecutive applications of the operator)*

Thus, through Equation (5), each bat i updates its position by applying the operator $\emptyset$ v_i times and choose the best position.

Computation of v_i A bat i updates velocity through the following Equation (4)

$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right) f_i.$$

Inputs: Partitions $P_i = \{E_1^i, E_2^i, \ldots, E_k^i\}$ and $P_* = \{E_1^*, E_2^*, \ldots, E_k^*\}$.
Step 1: Create a cost matrix $A = [a_{rs}]$ by putting

$$a_{rs} = |E_r^i \cap E_s^*| \;\; r, s \in \{1, \ldots, k\}$$

Step 2: Use Hungarian algorithm to calculate the value of maximum assignment w of the cost matrix A.
Step 3: Return w as the value of $x_i - x_*$

Fig. 2. Computation of $x_i - x_*$

.

The equation indicates that the current velocity of each bat i is influenced by two components: one is its previous velocity and the other is the product of the difference between positions of the bat i and the best bat in the population, and the frequency f_i. We deal with these two components as follows: We assign

two probabilities ρ_1 and ρ_2 for the first and second components, respectively, such that $\rho_1 + \rho_2 = 1$. That is, in any step, the update of a bat is influenced by exactly one of the two components. One can easily define ρ_1 and ρ_2 as follows:

$$\rho_1 = \frac{v_i^{t-1}}{v_i^{t-1} + (x_i^t - x_*)\, f_i} \quad \text{and } \rho_2 = \frac{(x_i^t - x_*)\, f_i}{v_i^{t-1} + (x_i^t - x_*)\, f_i}$$

Thus, Equation (4) is replaced by the following equation in the proposed DBA for ETP.

$$v_i = \begin{cases} v_i^{t-1} & \text{if } 0 \le \sigma \le \rho_1 \\ (x_i^t - x_*) f_i & \text{if } \rho_1 < \sigma \le 1 \end{cases} \tag{9}$$

where $\sigma \in [0, 1]$ is randomly generated.

Equation (9) requires computation of distance between two positions (candidate solutions).

```
Set Population of Bat P = {1, ..., n}
For each Bat i ∈ P do
    Initialize: Position x_i, Velocity v_i, Pulse rate r_i
    Loudness A_i
End for
Repeat
    For each x_i ∈ P do
        f_i = f_min + (f_max − f_min)β
        w = x_i − x_* //Use algorithm in Figure 2
        ρ = v_i^t / (v_i^t + w f_i)
        Generate random number σ ∈ [0, 1]
        If 0 ≤ σ ≤ ρ then v_i^{t+1} = v_i^t
        If ρ ≤ σ ≤ 1 then v_i^{t+1} = w f_i
        Perform v_{i+1} operations to improve position of x_i^{t+1}
        and choose the best one.
        Generate random number rand
        If rand > r_i
            Generate a local solution around
            the selected best solution
        End if
        If rand < A_i and f(x_i) < f(x_*)
            Accept the new solution
            Increase r_i and reduce A_i
       End if
       Rank the bats and find the current best x_*
    End for
Until termination criterion is reached
Return the current best bat of the population;
```

Fig. 3. A pseudocode of Discrete Bat Algorithm for ETP

We use set partition-distance [8] based approach to compute $x_i - x_*$. Given a set S, a partition P is a collection of subsets of S such that each element of S appears in precisely one element of P. That is, $P = \{S_i | \bigcup_{i=1} S_i = S \wedge \forall i \neq j \rightarrow S_i \cap S_j = \emptyset\}$.

The partition-distance between the two partitions P and P' of a set S is defined as the minimum number of elements which must be moved between subsets in P' such that the resulting partition equals P.

For a given instance of ETP with examination set E, since each examination appears in precisely one timeslot, any feasible solution s induces a partition P of E. Thus, two solutions x_i and x_* of ETP respectively induce partitions P_i and P_* of E. Therefore, $x_i - x_*$ can be equated with the distance between these two partitions. The following results of Gusfield [8] indicates that the distance between partitions can be computed in polynomial time.

Theorem 4 ([8]) *The problem of solving distance of two partitions can be reduced into Assignment Problem.*

The Assignment Problem [14] is well known optimization problem, and it can be solved by Hungarian algorithm with $\mathcal{O}(n^3)$ time complexity (see, e.g., [25]). The value of $x_i - x_*$ can be computed in polynomial time by using the algorithm in Figure 2.

A pseudocode of the proposed discrete bat algorithm for ETP is given in Figure 3.

4 Numerical Results and Performance Comparison

The proposed algorithm was applied and tested using real-world examination timetabling dataset at the University of Dar es Salaam, Semesters I and II in the academic year 2017/2018. The algorithm was implemented by using Java programming language and ran on Celeron(R) Dual-core CPU T300@ 1.80 GHz 1.79GHz machine with 3.00 GB RAM and Window 10. Table 1 shows characteristics of the data for the problem that we have tested.

‘

Table 1. Characteristics of dataset

Sem	**no. of exams**	**average no. of students per exam**	**Number of rooms**	**Number of timeslots**
Sem. I	924	138	82	28
Sem. II	901	116	82	28

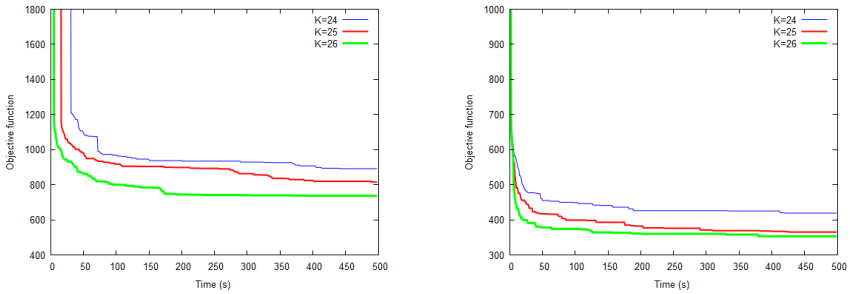
Through experience and after experiments the parameters of DBA as applied in this work are given in Table 2.

Table 2. The parameters for discrete for Bat Algorithm for ETP

Parameter	**Value/Range**
Population Size N	5
Constants α and γ	0.99
Emission rate r_i	$[0, 1]$
Loudness A_i	$[0, 1]$
Minimum frequency $f_{\min}$	0
Maximum frequency $f_{\max}$	10

There are different stopping conditions for optimization algorithms. Examples include the maximum number of iterations, number of iterations without improvements and CPU time. In this work, after performing experiments, we set CPU time =500 seconds as the stopping criteria.

As indicated in Table 1, there are 28 timeslots for timetable in each semester. However, experiences show that in generating earlier versions of examination timetables, one or two timeslots are left free (unused) so that they can be used when unforeseen problems arise. Thus, in this work, we have tested the algorithm using the number of timeslots $K = 24,\ 25$ and 26 in each semester. Figure 4(a) and 4(b) show convergence behaviour of the objective function for the above values of K in Semester I and II, respectively. From the figures we observe that the values of the objective function decrease very fast in the first few seconds. For example, in Figure 4(a), when $K = 26$ the value of objective function dropped from the initial value of 220,501 to 948 at time= 20 sec. Similarly, in Figure 4(b) for $K = 26$, the value of the objective function dropped from 175,128 to 503 at time=20 sec. This and the equation of the objective function in Model (2) indicate that DBA generated feasible solutions within few seconds. In addition, the figures clearly indicate that good solutions are obtained more rapidly when the value of K is higher.



(a) Semester 1 (b) Semester 1

Fig. 4. Convergence behaviour of modified BA for various values of time slots K.

The true value of a new algorithm is known once it has been compared with the other algorithms which solve the same problems [5]. Therefore, in order to test the computational performance of DBA, a well known heuristic Tabu Search (TS) was implemented in the same platform and using same to dataset. Table 3 gives final values of objective functions after running DBA and TS, each for 500s. The table clearly shows that DBA performs better in all cases. For example, in Semester II, $K = 25$, a timetable that was generated by DBA had 24 back-to-back examinations, while the timetable generated by TS had 45 such examinations. In the same semester, for $K = 26$, DBA produced a timetable with 17 back-to-back examinations while TS produced a timetable with 45 back-to-back examinations.

Table 3. Comparison of objective functions between DBA and TS after running each heuristic for 500s for timeslots $K = 25$ and $K = 26$.

Semester	K	Function	DBA	TS
I	25	$f_1(s)$	66	113
		$f_2(s)$	484	543
		$f(s)$	814	1108
	26	$f_1(s)$	56	91
		$f_2(s)$	457	603
		$f(s)$	737	1058
II	25	$f_1(s)$	24	55
		$f_2(s)$	285	321
		$f(s)$	405	596
	26	$f_1(s)$	17	45
		$f_2(s)$	317	36
		$f(s)$	402	591

Table 4. Comparison of convergance between DBA and TS after running each heuristic for 500s for timeslots $K = 25$ and $K = 26$.

Time (Sec)	Semester I				Semester II			
	K=25		K=26		K=25		K=26	
	DBA	TS	DBA	TS	DBA	TS	DBA	TS
1	22501	774099	13561	765155	4371	752734	4049	668754
20	1095	473797	948	440859	502	416340	503	356334
50	982	254589	862	203502	476	205971	459	152041
100	917	92209	800	8273	452	37833	422	1784
200	983	1743	744	1494	439	1048	419	1067
300	863	1335	740	1194	412	6810	417	733
400	824	1118	737	1061	406	637	411	618
500	814	1108	737	1058	405	596	402	604

Table 4 compares converges solutions versus execution time. This table shows that, in all cases, DBA generated good solutions faster than TS. It can be noted that in this table and Figure 4, in both semesters, there are quick drop in objective function values within the first 50 seconds, followed by a slow convergence. This behaviour is normal for global heuristics techniques, where improvement in solution quality is expected to slow down when approaching optimal value (see, e.g., [19]).

5 Concluding Remarks

In this work, we have presented a discrete version (DBA) of Bat Algorithm (BA) for the examination timetabling problem. Since the original BA was designed for continuous problems, we have proposed modifications on BA in order to solve the ETP. The proposed DBA was tested using real data from the University of Dar es Salaam. The experimental results show that it generates feasible solutions of high quality very fast. When compared with Tabu Search (TS) heuristic, DBA has demonstrated superior performance. In particular, DBA has outperformed TS in terms of time and the quality of solutions.

There are other mathematical models for ETP with different objectives and constraints. In this work we have developed a model with two soft constraints. Thus, a possible future work is to extend the model by adding more soft constraints, and then solve the resulting model with DBA. In addition, there are different moves possible for the ETP. In this paper we have used 1-1 move in the implementation of DBA. It would be interesting to investigate the impacts of other moves such as 1-1 on DBA. Furthermore, we have compared the performances of DBA and TS. It is worth to compare the performance of DBA with other heuristics such as ACO, PSO and SA. Finally, it would be profitably interesting to investigate the impact of hybridizing the DBA with other metaheuristics on solving the examination timetabling problem.

References

1. B. Aldeeb, M. Al-Betar, A. Abdelmajeed, M. Younes, M. Alkenani, W. Alomoush, K. Alissa, and M. Alqahtani. A comprehensive review of uncapacitated university examination timetabling problem. *A Comprehensive Review of Uncapacitated University Examination Timetabling Problem.* International Journal of Applied Engineering Research, 14(24):4524–4547, (2019).
2. A.L. Bolaji, A.T.A. Khader, M.A. Al-Betar, and M.A. Awadallah. A hybrid nature-inspired artificial bee colony algorithm for uncapacitated examination timetabling problems. *Journal of Intelligent Systems*, 24(1):37–54, s(2014).
3. E.K. Burke, J.H. Kingston, and D. deWerra. Applications to timetabling. In J. Gross and J. Yellen, editors, *The Handbook of Graph Theory*, pages 445–474. Chapman Hall/CRC Press, (2004).
4. Z.Y. Du and Liu B. Image matching using a bat algorithm with mutation. *Applied Mechanics and Materials*, 203(1):88–93, (2012).

5. I. Fister, S. Fong, J. Brest, and I. Fister. A novel hybrid self-adaptive bat algorithm. *Scientific World Journal*, 2014.
6. A.H. Gandomi, X. Yang, A.H. Alavi, and S. Talatahari. Bat algorithm for constrained optimization tasks, neural computing and applications. *Neural Computing and Applications*, 22(6):1239–1255, (2013).
7. L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III. PATAT 2000. Lecture Notes in Computer Science*, volume 2079, pages 104–117. Springer, Berlin, Heidelberg, (2001).
8. G. Gusfield. Partition-distance: A problem and class of perfect graphs arising in clustering. *Information Processing Letters*, 82(3):159–164, (2002).
9. S. Induja and V.P. Eswaramurthy. Bat algorithm: An overview and its applications. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(1):448–451, (2016).
10. S Jeyasingh and M. Veluchamy. Modified bat algorithm for feature selection with the wisconsin diagnosis breast cancer (wdbc) dataset. *Asian Pacific journal of cancer prevention*, 18(5):1257–1264, (2017).
11. K. Khan, A. Nikov, and A. Sahai. A fuzzy bat clustering method for ergonomic screening of office workplaces. In D. Dicheva et al, editor, *Software, Services & Semantic Technologies, Advances in Intelligent and Soft Computing*, volume 101, pages 59–66. Springer-Verlag, Berlin, Heidelberg, (2011).
12. R. Khosravanian, V. Mansouri, D.A. Wood, and R.A Masood. A comparative study of several metaheuristic algorithms for optimizing complex 3-d well-path designs. *Journal of Petroleum Exploration and Production Technology*, 8(4):1487–1503, (2018).
13. G. Komarasamy and A Wahi. An optimized k-means clustering technique using bat algorithm. *European Journal of Scientific Research*, 84(2):263–273, (2012).
14. H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2 (1-2):83– 97, (1955).
15. N. Leite, F. Melicio, and A.C. Rosa. A fast simulated annealing algorithm for the examination timetabling problem. *Expert Systems with Applications*, 122(1):137–151, (2019).
16. M.K. Marichelvam, T. Prabaharan, X.S. Yang, and M. Geetha. Solving hybrid flow shop scheduling problems using bat algorithm. International Journal of Logistics Economics and Globalisation, 5(1):15–29, (2013).
17. S.L Marie-Sainte. A new hybrid particle swarm optimization algorithm for real-world university examination timetabling problem. In *2017 Computing Conference, London, UK*, pages 157–163, 07 (2017).
18. L.T.G. Merlot, N. Boland, B.D. Hughes, and P.J. Stuckey. A hybrid algorithm for the examination timetabling problem. *Lecture Notes in Computer Science*, 2740:207–231, (2003).
19. E. Mujuni and A. Mushi. Solving the examination timetabling problem using a two-phase heuristic: the case of sokoine university of agriculture. *Journal of Information and Computing Science*, 10 (3):220–227, (2015).
20. A.R. Mushi. Implementation of a tabu search heuristic for the examinations timetabling problem. *Tanzania Journal of Science*, 37:84–93, (2011).
21. C. Ochoa-Zezzatti, L. Margain, J. Arreola, DeLuna A., Garca , E. Soto, S. Gonzalez, Haltaufoerhyde K., and Scarandangotti V. Improved solution based on bat algorithm to vehicle routing problem in a caravan range community. *Proceedings of the* 13^{th} *International Conference on Hybrid Intelligent Systems*, pages 18–22, (2013).

22. E. Osaba, X.S. Yang, F. Diaz, P. Lopez-Garcia, and R. Carballedo. An improved discrete bat algorithm for symmetric and asymmetric travelling salesman problems. *Engineering Applications of Artificial Intelligence*, 48(1):59–71, (2016).
23. C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, (1998).
24. V.T Paschos. *Applications of Combinatorial Optimization. Mathematics and Statistics*. Wiley-ISTE, (2014).
25. C.R. Pedersen, L.R. Nielsen, and K.A. Andersen. An algorithm for ranking assignments using reoptimization. *Computers and Operations Research*, 35(11):3714–3726, (2008).
26. J. Qin, Y. Yin, and X. Ban. Hybrid discrete particle swarm algorithm for graph coloring problem. *Journal of Computers*, 6(6):1175–1182, (2011).
27. Qu R., E.K. Burke, B. McCollum, L.T.G. Merlot, and S.Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12 (1):55–89, (2009).
28. Y. Saji and M.E. Riffi. A novel discrete bat algorithm for solving the travelling salesman problem. *Neural Computing and Applications*, 27(7):1853–1866, (2016).
29. S. Srivastava and S.K. Sahana. Application of bat algorithm for transport network design problem. *Applied Computational Intelligence and Soft Computing Applied Computational Intelligence and Soft Computing*, 2019:1–12, (2019).
30. T.P. Talafuse and E.A. Pohl. A bat algorithm for the redundancy allocation problem. *Engineering Optimization*, 48(5):900–910, (2016).
31. E. Talbi. *Metaheuristics: From Design to Implementation*. John Wiley & Sons, (2009).
32. A.L. Tamiru and F.M. Hashim. Application of bat algorithm and fuzzy systems to model exergy changes in a gas turbine. In X.S. Yang, editor, *Artificial Intelligence, Evolutionary Computing and Metaheuristics. Studies in Computational Intelligence*, volume 427, pages 685–719. Springer, Berlin, Heidelberg, (2013).
33. S.L. Tilahun. Prey-predator algorithm for discrete problems: a case for examination timetabling problem. *Turkish Journal of Electrical Engineering and Computer Sciences*, 27(2):950–960, (2019).
34. K. Wanatchapong. Bat algorithm in discrete optimization: A review of recent applications. *Songklanakarin Journal of Science and Technology (SJST)*, 39(5):641–650, (2017).
35. D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, (1997).
36. L.A Wolsey and G.L. Nemhauser. *Integer and combinatorial optimization*. Wiley, London, Hoboken, (2014).
37. X.S Yang. A new metaheuristic bat-inspired algorithm. In J. R. González, D.A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74. Springer Berlin Heidelberg, Berlin, Heidelberg, (2010).
38. X.S. Yang. Metaheuristic optimization: Nature-inspired algorithms and applications. In X.S. Yang, editor, *Artificial Intelligence, Evolutionary Computing and Metaheuristics. Studies in Computational Intelligence*, volume 427, pages 405–420. Springer Berlin Heidelberg, Berlin, Heidelberg, (2013).
39. M. Yazdani and F. Jolai. Lion optimization algorithm (loa): A nature-inspired metaheuristic algorithm. *Journal of Computational Design and Engineering*, 3(1):24–36, (2015).