Tanzania Journal of Science 47(2): 674-685, 2021
ISSN 0856-1761, e-ISSN 2507-7961
© College of Natural and Applied Sciences, University of Dar es Salaam, 2021

Solving the University Course Timetabling Problem Using Bat Inspired Algorithm

Ushindi Limota, Egbert Mujuni and Allen Mushi*
Department of Mathematics, Box 35062, University of Dar es Salaam, Tanzania.
E-mail addresses: ushilimo@googlemail.com, emujuni@udsm.ac.tz, allenmushi66@gmail.com, mushi.allen@udsm.ac.tz
**Corresponding author*

Received 6 Feb 2021, Revised 26 Apr 2021, Accepted 27 Apr 2021, Published May 2021
DOI: https://dx.doi.org/10.4314/tjs.v47i2.23

Abstract
Many mathematical optimization problems from real-life applications are NP-hard, and hence no algorithm that solves them to optimality within a reasonable time is known. For this reason, metaheuristic methods are mostly preferred when their size is big. Many meta-heuristic methods have been proposed to solve various combinatorial optimization problems. One of the newly introduced metaheuristic methods is a bat-inspired algorithm, which is based on the echolocation behaviour of microbats. Bat algorithm (BA) and its variants have been used successfully to solve several optimization problems. However, from the No-free Lunch Theorem, it is known that there is no universal metaheuristic method that can solve efficiently all optimization problems. Thus, this study work focused on investigating the usefulness of BA in solving an optimization problem called Course Teaching Problem (CTP). Since BA was originally designed to solve continuous problems, and CTP is a combinatorial optimization problem, a discrete version of BA for CPT has been proposed and tested using real-life data from the Dar es Salaam University College of Education (DUCE). The algorithm has produced promising results, as in each execution test, it generated a timetable in which all hard constraints were met and soft constraints were significantly satisfied within a few iterations.

Keywords: Combinatorial optimization, Timetabling problem, Metaheuristic algorithms, Bat algorithm.

Introduction

Optimization problems arise in different fields such as economics, industries, education, manufacturing systems such as engineering designs and many other sectors. Due to the importance of optimization problems, it is required to have effective and strong computational algorithms to solve large-sized optimization problems. Basically, techniques for solving optimization problems depend on whether the problem is continuous or discrete. While continuous optimization problems contain completely continuous variables, a discrete optimization problem contains some discrete variables, i.e. the domain of the respective variable consists of a finite set of discrete values (Alba 2005). The set of discrete optimization problems forms an optimization class of problems called combinatorial optimization whose decision variables are discrete and has a finite number of feasible solutions (Baghel et al. 2012). Examples of combinatorial optimization problems include the Travelling Salesman Problem (TSP), Machine Scheduling Problems, Timetabling Problems (TP) and Assignment Problems. It is worth noting that besides theoretical significance, combinatorial optimization problems have empirical relevance, as they apply to real-world situations because they arise in several and heterogeneous domains, such as scheduling, location problems, transportation, production planning, decision-making processes, routing and telecommunications, just to name but a few (Festa 2014). Due to the concrete usefulness of combinatorial optimization problems, large numbers of methods for solving them have been established.

Methods for solving combinatorial optimization problems can be categorized as exact or approximate. Exact methods guarantee to obtain an optimal solution for each instance of a combinatorial optimization problem in a limited time, while approximate methods seek to achieve solutions that are as nearly as possible to the optimum values within a reasonable amount of time. One example of approximate methods is heuristic methods. There is a class of combinatorial optimization problems named NP-hard. This class contains problems for which no polynomial-time algorithm to solve them to optimality is known, and it has been conjectured that such algorithms do not exist (Garey and Johnson 1979). Unfortunately, many problems in real-life applications fall into this class of NP-hard problems and have large sizes. For this reason, heuristics and metaheuristics (a combination of heuristic methods) are mostly preferred. Heuristics and metaheuristic methods compute good solutions quickly but without a guarantee of optimality.

The purpose of heuristic methods is to obtain good quality solutions, not necessarily optimal but within a reasonable amount of time. Generally, heuristic methods are based on experience (that is, rule of thumb). Thus, heuristics often have an intuitive justification (Winston et al. 2003). Improvement of heuristic methods has resulted in metaheuristic methods that have a high capacity for solving a large class of problems. Osman and Kelly (1996) defined metaheuristic as an iterative generation process that guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find a near-optimal solution efficiently.

The two main components of meta-heuristic algorithms are intensification (also called exploration) and diversification (also called exploitation) (Yang 2010b). Diversification tries to ensure that the algorithm does not get trapped into local solutions, while the idea behind intensification is to obtain a high-quality solution within the explored regions. Thus, for meta-heuristics to be successful in solving optimization problems, intensification and diversification strategies should balance in order to produce an equilibrium between the exploitation of the collected search experience and exploration of the search space to find areas with the best solutions in a specific problem, close to the optimal solution (Birattari et al. 2001). A good combination of the two components improves the achievement of global optimality.

Meta-heuristic algorithms have been effectively applied to solve various real-life problems. A list of such applications includes Travelling Salesman (Wang et al. 2009), Facility Location (Qin et al. 2012), Aggregate Production Planning (APP) (Abu Bakar et al. 2016), Examination Timetabling (Mujuni and Mushi 2015), Shortest Path in Network (Gonen 2006), Agriculture systems for maximizing benefits (Srivastava and Yagyasen 2016), Vehicle routing (Masum et al. 2011), Optimization of the shift schedules for nurses (Augustine et al. 2009), Detection of Epistatic Interactions (Guan and Zhao 2019), Aircraft Recovery (Zegordi and Jafari 2010), Train Timetabling (Su and Huang 2006) and Aircraft Departure Scheduling (Atkin et al. 2008). This study aims at investigating the usefulness of a relatively new metaheuristic, a bat algorithm, on a combinatorial problem called Course Teaching Problem (CTP).

Bat algorithm is one of the new metaheuristic algorithms. Proposed in Yang

(2010a), it mimics the behaviour of bats when they hunt for food. Bats are mammals with wings and they use a process called echolocation in finding preys and differentiate different types of objects by varying pulse rates of emission and loudness (Yang 2010a). Bat algorithm is a population-based metaheuristic, and it uses communication between individuals in searching solution. The key advantage of the bat algorithm is that it can balance between the diversification and intensification strategies by adjusting parameters when the global optimality is approaching.

Bat inspired algorithms have been used to solve various optimization problems from both continuous and discrete fields. A list of such applications includes Scheduling, Allocation, Routing, Facility Layout Design (Kongkaew 2017), Image Processing (Zhang and Wang 2012) and Transport Network Design (Srivastava and Sahana 2019). See Kongkaew (2017) for overviews and applications of bat algorithms.

Course Timetabling Problem

The increasing enrolment numbers in academic institutions over the years and the complexity of course choices among students pose huge challenges to the timetabling process. The need to ensure that the teaching space is optimally used is extremely important for academic planning and hence increasing interest in timetabling research.

Course Timetabling Problem (CTP) falls into a wide class of Timetabling Problems (TTP). The timetabling problem involves the allocation of resources to a limited number of timeslots, subject to the given constraints (Wren 1996). Educational timetabling can be classified into three, namely, school timetabling, course timetabling and examination timetabling (Schaerf 1999). All three problems have some common characteristics of the general TTP but still can have significant differences between them (Mushi 2011).

CTP is defined as an assignment problem in which students, teachers (or faculty members) are assigned to courses, course sections or classes; events (individual meetings between students and teachers), classrooms and timeslots (Carter and Laporte 1998). CTP is a complex and time-consuming task. It involves the allocation of classes to times and space subject to a set of constraints, which are classified as hard and soft. Hard constraints are those which must be met, while soft constraints are not necessarily met but must be satisfied as much as possible (see e.g. Mushi 2011). A timetable is said to be feasible if it satisfies all hard constraints.

CTP is known to be NP-hard and therefore metaheuristic methods are widely used to solve it. Examples of such metaheuristics include Tabu Search (Mushi 2006, Aladag and Hocaoglu 2007), Simulated Annealing (Aycan and Ayav 2009, Basir et al. 2013), Genetic Algorithm (Al-Jarrah et al. 2017), Constraint Programming (Abdennadher and Marte 2000). To the best of the authors' knowledge, very little has been done on applications of a newly proposed Bat-Inspired algorithm in solving the course timetabling problem using real data. This has motivated this work.

Mathematical Programming Formulation

As mentioned above, a teaching timetable consists of hard and soft constraints. This paper considered the following hard constraints:

1. There is no collision of two or more courses, i.e. no two or more courses are assigned into the same timeslot in the same room.
2. The rooms must have enough size which can accommodate registered students.
3. A student cannot attend more than one course at the same timeslot.

Soft constraints considered are:

1. Minimize the number of lectures of the same course in a day. The idea is to spread courses over the teaching week.
2. A student should not have consecutive sessions in a day.

There are various models for CTP. Chacha and Mushi (2013) gave examples of such models. In this work, we have opted to use a binary model to represent a schedule of lecture and courses after performing experimentations. The mathematical model will have the following form:

$$\text{Minimize} \quad f(s)$$
$$\text{Subject to} \quad h(s) \le 0,$$

where s is a solution, $f(s)$ is the objective function, which consists of soft constraints and $h(s)$ consists of hard constraints. First, we need to define sets, parameters and variables to be used in our formulation.

Sets:
T: Set of all time slots,
R: Set of all rooms,
L: Set of lectures.
Parameters:
$M_{i,j} = 1$ if lectures i and j have some common students; 0 otherwise.
$L_{i,j,} = 1$ if i and j are lectures of the same course; 0 otherwise.
S_i: The number of students in lecture i.
C_r: Sitting capacity of room R.
Variables:
$x_{i,t,} = 1$ if lecture i is scheduled in timeslot t.
$y_{i,r,} = 1$ if lecture i is scheduled in room r.
$z_{r,t,} = 1$ if room r is to be used in timeslot t.

Model:
Minimize $f(s) = \lambda_1 f_1(s) + \lambda_2 f_2(s)$ (1)
Subject to:
$\sum_{i\in L}\sum_{j\in L} M_{i,j}.x_{i,t}.x_{j,t} = 0 \ \forall t \in T$ (2)
$\sum_{t\in T} x_{i,t} = 1 \ \forall i \in L$ (3)
$\sum_{r\in R} y_{i,r} = 1 \ \forall i \in L$ (4)
$\sum_{i\in L} y_{i,r} + z_{r,t} \le 1 \ \forall i \in L$ (5)
$x_{i,t}y_{i,t} - z_{r,t} \le 0 \ \forall i \in L, \forall r \in R, \forall t \in T$ (6)
$S_i y_{i,r} - C_r \le 0 \ \forall i \in L$ (7)
$x_{i,t}, y_{i,r}, z_{r,t} \in \{0,1\}$

Equation (1) is the objective function that needs to be minimized and represents soft constraints as detailed in the next section. Equation (2) ensures that lectures with some common students are not scheduled in the same timeslot. Equations (3) and (4) guarantee that each lecture is scheduled precisely once in a timeslot and room, respectively. Equation (5) ensures that at most one lecture is assigned to a room in each timeslot. Equation (6) establishes a connection between the variables $x_{i,t}, y_{i,r}$ and $z_{r,t}$. Equation (7) guarantees that the total number of students in a lecture i does not exceed the capacity of the assigned room.

Objective Function
The objective function is formulated to minimize violations of soft constraints. The hard constraints are also penalized in the objective function so that all constraints are represented. Thus, it is a linear combination of functions of the form; $f(s) = \lambda_1 f_1(s) + \lambda_2 f_2(s) + \lambda_3 f_3(s)$, where $f_1(s)$ is the number of lectures of the same course offered on the same day, $f_2(s)$ is the number of conflicting lectures that are offered in consecutive timeslots, $f_3(s)$ is the number of hard constraints that have not been satisfied, and λ_1, λ_2, λ_3 are weights indicating the importance of the associated constraint in relation to other constraints. Since λ_3 is associated with hard constraints, it is given a much higher value compared to those of λ_1 and λ_2. Let

$$L(i,j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are lectures of the same course} \\ 0 & \text{Otherwise} \end{cases}$$

$$D(i,j) = \begin{cases} 1 & \text{if lectures } i \text{ and } j \text{ are offered in the same day} \\ 0 & \text{Otherwise} \end{cases}$$

Then,

$$f_1(s) = \sum_{i\in L}\sum_{j\in L} L(i,j)D(i,j) \text{ and } f_2(s) = \sum_{i\in L}\sum_{j\in L} x_{i,t}\, x_{i,t+1} M_{i,j} \qquad (8)$$

Methodology
Original Bat Algorithm
Bat Algorithm (BA) is one of the relatively new meta-heuristic algorithms which is nature-inspired. It was proposed by Yang in 2010 (Yang 2010a). BA mimics the echolocation behaviour of bats when they are searching for food and navigation. Echolocation is an exceptional hearing based navigation system used by bats and some other animals to sense and identify preys or any obstacle in their surrounding environment through releasing a

sound (Yang 2010a). For simplicity, Yang (2010) suggested the following idealized rules:

1. Bats use echolocation to determine distance, and they can differentiate between food and barriers.
2. Bats fly randomly with velocity v_i at position x_i having a fixed frequency f_{min}, varying wavelength λ and loudness A_i to search a prey. In this rule, it is assumed that bats can adjust automatically the frequency (or wavelength) of their emitted pulses as well as the rate of pulse emission $r \in [0,1]$. The automatic adjustment relies on the proximity of the target.
3. The loudness of the bats can vary in many ways, however, it is assumed that the loudness can vary from between positive values A_0 and A_{min}.

Figure 1 gives basic pseudo-code for BA.

```
Define the Objective function $f(x)$.
Generate Initial population of the bat $X = \{x_1, \dots, x_n$
For each bat $x_i \in X$
   Initialize the pulse rate $r_i$, velocity $v_i$ and loudness $A_i$
   Define the pulse frequency $f_i$;
End for
While termination criterion not reached;
   For each bat $x_i \in X$
      Generate new solutions by using Equations (9), (10, and (11)
      Generate a random number $rand$
      If $(rand > r_i)$
         Select one solution among the best one
         Generate a local solution around one of the best
      End if
      If $(rand < A_i)$ &$(f(x_i) < f(x_*))$
         Accept the new solution
         Increase $r_i$ and reduce $f(x_i)$
      End if
   End for
End while
Ranks the bats and obtain the current best bat
```

Figure 1: Pseudo-code of the original bat-inspired algorithm (BA).

The first step involves initialization of bat population: position x_i, velocity v_i and frequency f_i. At each generation, each bat moves by adjusting its velocity v_i^t and position x_i^t at a time t using the following equations:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \qquad (9)$$
$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_* \ ) f_i \qquad (10)$$
$$x_i^t = x_i^{t-1} + v_i^t \qquad (11)$$

where, β is a random number generated from the interval $[0,1]$ and x_* represents the current global best solution obtained after comparing all the solutions among all n bats in the swarm.

Additionally, for the local search part, even if a solution is selected among the best solutions, a new solution for each bat is generated using a random walk.

$$x_{new} = x_{old} + \epsilon < A^t > \qquad (12)$$

where ϵ is a randomly generated number in the interval $[0,1]$ and $< A^t >$ is the average loudness of the bats in the swarm at a specific time step t. Moreover, the loudness A_i and the rate of pulse emission r_i of each bat have to be updated consequently using the following equations:

$$A_i^t = \alpha A_i^{t-1}, r_i^t = r_i^0[1 - e^{-\gamma t}] \qquad (13)$$

where α and γ are constant.

Proposed Bat Algorithm for CTP

The basic bat algorithm discussed above was basically established for solving continuous optimization problems. Since Course Timetabling Problem (CTP) is a discrete combinatorial optimization problem, some modifications on the basic BA are necessary.

In the proposed BA, a position x_i of the bat, i presents a candidate solution for the CTP. In solving the CTP using the proposed algorithm, each position x_i is initialized by using a heuristic that randomly chooses a lecture and then tries to find timeslots without collision and a room big enough to accommodate the lecture. The idea is to start with a good initial timetable.

Considering the basic parameters of the original Bat algorithm, which are pulse rate r_i, loudness A_i, frequency f_i and speed, the first two parameters (r_i and A_i will be used in the way as in the basic BA and initialized randomly. But, frequency f_i has been fixed to 1 in the proposed algorithm to reduce the complexity of the algorithm. Velocity v_i has been modified by relating it with the value of the objective function. This approach had been used previously to solve the Travelling Salesman Problem (Osaba et al. 2016). Thus, we have adapted v_i using the value of the objective function in the following way:

$$v_i^t = f(x_i) - f(x_*) \qquad (14)$$

In Equation (14), v_i can be viewed as the number of operations that bat i will perform in updating the current position.

In the basic BA, a new position x_i^t is calculated through Equation (11).

$$x_i^t = x_i^{t-1} + v_i^t \qquad (15)$$

Equation (15) implies that the current position depends on the previous position and velocity. Thus, this equation can now be interpreted as that the current position is obtained by performing v_i^t moves.

To update the current solution of an instance of the CTP, the most commonly used strategies are 1-0 and 1-1 moves. For the case of 1-0 move, a lecture is moved from one timeslot to another. In the other case timeslots of two lectures are swapped. Because of its simplicity, in this work, we have opted to use 1-0 move, in which both lectures and timeslots are randomly selected.

A pseudo-code of the modified bat algorithm for CTP is given in Figure 2.

Implementation of the method

A modified BA was implemented by using Java programming language. Table 1 and Table 2 give the used parameters of BA and weights of the problem, respectively.

```
Define the Objective function $f(x)$.
Generate Initial population of the bat $X = \{x_1, \dots, x_n\}$
For each bat $x_i \in X$
    Initialize the pulse rate $r_i$, velocity $v_i$ and loudness $A_i$
    Define the pulse frequency $f_i$;
End for
While termination criterion not reached;
    For each bat $x_i \in X$
        $v_i = f(x_i) - f(x_*)$
        Perform 0-1Moves $v_i$ times to update $x_i$
        Generate a random number $rand$
        If $(rand > r_i)$
            Select one solution among the best one
            Generate a local solution around one of the best
        End if
        If $(rand < A_i) \& (f(x_i) < f(x_*))$
            Accept the new solution
            Increase $r_i$ and reduce $f(x_i)$
        End if
    End for
End while
Ranks the bats and obtain the current best bat
```

Figure 2: Pseudo-code of a modified BA for CTP.

Table 1: Parameters of BA

Parameter	Value
Population size; n	5
Emission rate; r_i	$r_i \in [0, 1]$
Loudness; A_i	$A_i \in [0, 1]$
Maximum iteration	500
α	0. 98
γ	0. 98

Table 2: Weight of the problem

Parameter	Value
γ_1	10
γ_2	1
γ_3	1000

Table 2 implies that since λ_3 is associated with hard constraints, any solution s with $f(s) < 1000$ must be feasible.

To generate timetables, the program used 6 input text files, namely: (i) *Conflict.txt* - containing courses conflict as per students registration, (ii) *CourseName.txt* containing names of the courses to be scheduled, (iii) *CourseSize.txt* containing the number of students registered for each course, (iv) *NumberOfLectures.txt* - containing the number of lectures for each course, (v) *RoomName.txt* - containing names of lecture rooms and (vi) *RoomSize.txt* containing the capacity of each lecture room. The file *Conflict.txt* was used to create a lecture conflict matrix. These input data were collected from Timetabling Section at DUCE and The Academic Registration Information System (ARIS) office.

Results and Analysis

A modified BA was tested by using real data from Dar es Salaam University College of Education (DUCE) which is a constituent College of the University of Dar es Salaam in Tanzania. Data for the first semester for the Academic Year 2018/2019 were used. Data included lectures excluding tutorials for science and mathematics as well as seminars for humanities. This is due to the fact that the Bat algorithm is a relatively new metaheuristic algorithm, thus the study focused on investigating the usefulness of the algorithm in solving combinatorial optimization problems. Small size instances of the problem are therefore sufficient for the research work.

The study creates a course timetable for undergraduate courses only which is a major current project. The data set consisted of 5551 students (from 1st to 3rd year) with a total of 115 undergraduate courses for the semester, 10 lecture rooms of different capacities excluding laboratories which were used for the teaching process. The largest room had a sitting capacity of 1000. There were seven (7) courses with more than 1000 students, hence exceeding the maximum room capacity of the available rooms. Those seven (7) big courses were split into either two or three sub courses. This gives a total of 124 courses that were used in the timetable construction. In addition, these courses need two (2) or three (3) lectures per week. Thus, in total there were 273 lectures for scheduling. There are 12 timeslots per day which makes a total of 60 timeslots for the five teaching days of a week (Monday to Friday). Table 3 provides a summary of the size of input data, Table 4 gives a sample course data, and Table 5 gives data on rooms (source: Timetable Section at DUCE). Table 6 shows a sample representation of the lecture conflict matrix.

Table 3: Size of input data

Courses Involved	Number of lectures	Number of rooms	Students registered
124	273	10	5551

Table 4: Size of input data

Courses Names	Number of students	Number of lectures per week
BL111	470	3
BT130	295	3
BT225	230	3
CH118	470	3
CH121	404	3
CL106	1584	2
CT201	1712	2

Table 5: Size of input data

Name	Size
MTR	100
RoomA	160
RoomB	160
RoomC	160
RoomD	160
RoomE	160
TheaterA	460
TheaterB	460
Hall	500
TheaterC	1000

Table 6: A sample conflict matrix (M_{ij})

	BL111 Lec1	BL111 Lec2	BL111 Lec3	BT130 Lec1	BT130 Lec2	CH116 Lec1	CH116 Lec2
BL111 Lec1	0	1	1	1	1	0	0
BL111 Lec2	1	0	1	1	1	0	0
BL111 Lec3	1	1	0	1	1	0	0
BT130 Lec1	1	1	1	0	1	0	0
BT130 Lec2	1	1	1	1	0	0	0
CH116 Lec1	0	0	0	0	0	0	1

As mentioned earlier, the algorithm was implemented using Java programming language, on a PC machine with Intel(R) Celeron(R) 2955U processor of 1.4 GHz speed in Windows 10.

Computational Results

The proposed algorithm was able to generate a feasible solution of good quality. Table 7 gives a sample of solutions that were obtained.

Table 7: A sample of a course timetable

Day	07:00-08:00	08:00-09:00	. . .	13:00-14:00	. . .	17:00-18:00	18:00-19:00
Mon	CH243-ThtA CT225-RmB CT226-MTR	CT107-ThtC DS101-MTR HI103-ThtB	. . .	CH219-ThtA CT237-RmA CT302-RmB	. . .	BL111-ThtC CH290-ThtA LL212-Hall	EC371-RmB EF200-RmA KF302-ThtA
Tues	BT225-ThtC CH121-ThtB CH243-ThtA	EF200-RmA EP101A-ThtC	. . .	GE142-Hall GE352-ThtC	. . .	CH290-ThtA EC373-RmB KI310-ThtC	C117-RmB KF202-ThtA ZL236-ThtC
Wed	CH118-Hall CT106-ThtC LL201-ThtA	CH377-RmC CL106A-ThtC	. . .	CM105-MTR DS101-RmB	. . .	CT209-RmA DS112A-ThtC	CH248-ThtB EF100B-ThtC KI208-ThtA
Thur	CH116-RmC GE352-ThtC	DS112B-ThtC KI208-ThtB	. . .	CH118-Hall CM209-RmA EA200-RmC	. . .	CT229-RmB EP101A-ThtC IS272-MTR	BT225-ThtB T228-RmA EC371-RmB
Frid	CT229-MTR C116-RmB EP101B-	CT302-RmA EP306A-ThtC	. . .	CH118-Hall EA200-RmA EA300B-	. . .	BT130-ThtA CT225-MTR EC217-RmA	EC373-RmB EF100A-ThtC LT310-ThtA

Performance analysis

To assess the performance and the usefulness of the proposed algorithm, Table 8 presents a summary of results obtained after running the proposed algorithm 4 times.

Table 8: Performance of the algorithm

Description	**Run/Execution**			
	1	**2**	**3**	**4**
Initial Objective function value	12200	9860	11900	12110
Number of lecture collisions	0	0	0	0
Number of lectures exceeding room capacity	0	0	0	0
Number of lecturers of the same course that offered on the same day	0	0	9	11
Number of back to back lectures	71	68	74	75
Number of courses not assigned a room	0	0	0	0
Final Objective function value	71	68	74	75

It is worth noting that, in all 4 runs, all hard constraints were satisfied. That is, there is no lecture collision and each lecture is allocated in an appropriate room. Also, the table indicates that there were no lectures of the same course which were allocated on the same day. Furthermore, it can be noted that there were only a few conflicting courses which were allocated in consecutive timeslots. Since there were a total of 273 lectures to be scheduled, the table shows only between 24.9% and 27.5% lectures with some common students which were scheduled in consecutive timeslots. This has an implication that many students will have enough time to move from one lecture to another, in case they have another lecture on the same day.

On convergence, Table 9 and Figure 3 give the improvement on the number of soft constraints that are violated (f_1 and f_2)) and values of the objective function, respectively, against iterations for the four (4) runs of the algorithm. Table 9 and Figure 3 show that there is a sharp drop in target values within the first few iterations, followed by a slow convergence. This is a normal situation for metaheuristic methods, where improvement in solution quality is expected to slow down when convergence is approaching. Furthermore, Figure 3 indicates that feasible solutions were obtained only during the first few iterations. The quality of the solution is very good since the final objective function value is much smaller compared to the initial value.

Table 9: Improvements of soft constraints by iterations

Iteration	Run 1		Run 3		Run 3		Run 4	
	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
1	79	338	88	324	84	325	98	323
5	7	176	12	171	12	179	11	204
10	4	142	6	146	5	170	5	137
20	2	114	4	124	7	106	3	112
100	2	73	1	78	1	83	1	84
200	2	65	0	72	0	82	0	80
300	0	80	0	70	0	77	0	77
400	0	74	0	69	0	75	0	75
500	0	71	0	68	0	74	0	75

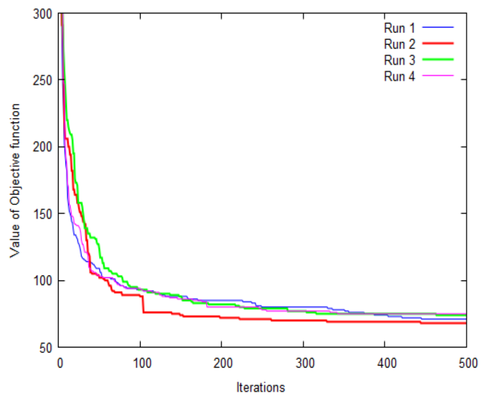

Figure 3: Convergence to solutions.

Conclusion and Recommendations

This study aimed at investigating the usefulness of the bat algorithm in solving the course teaching timetabling problem. Since the original bat-inspired algorithm was basically prepared for solving continuous problems, modifications have been made in order to solve a problem that is defined on a discrete domain. We have developed a discrete version of the bat algorithm for CTP, which was tested using real data from DUCE. During the implementation of the proposed algorithm, five bats were used and each bat's position represented a solution to the problem. The obtained results show that the Bat algorithm is capable of generating timetables with good quality (Table 8). Moreover, the results show that the proposed algorithm computed feasible solutions very fast (see Figure 3).

Future Research Directions

The study was able to produce a feasible course timetable at DUCE using the bat algorithm but some recommendations could be valuable for further studies on the Course Timetabling Problem at DUCE. It is recommended to do the following in future:

(i) The study only involved solving the problem by assigning lectures to timeslots and rooms. It is recommended to include seminars, tutorials and experiments for science subjects.

(ii) In this study, the bat algorithm was applied as a benchmark when solving the Course Timetabling Problem at DUCE. For further research, there is a need to apply hybrid heuristic techniques by combining the bat algorithm with other heuristics.

(iii) Performing further fine-tuning of parameters might reduce the running time of the algorithm.

(iv) It is worth doing a comparative analysis between the bat algorithm and other recent algorithmic techniques.

References

Abdennadher S and Marte M 2000 University course timetabling using constraint handling rules. *Appl. Artific. Intel.* 14(4): 311-325.

Abu Bakar MR, Bakheet AJK, Kamil F, Kalaf BA, Abbas IT and Soon LL 2016 Enhanced simulated annealing for solving aggregate production planning. *Mathematical Probl. Engin.* 2016: 1-9.

Aladag CH and Hocaoglu G 2007 A tabu search algorithm to solve a course timetabling problem. *Hacettepe J. Math. Statis.* 36(1): 53-64.

Alba E 2005 Parallel Metaheuristic: A New Class of Algorithms. John Wiley and Sons, Inc., Hoboken, New Jersey, Canada.

Al-Jarrah MA, Al-Sawalqah AA and Al-Hamdan SF 2017 Developing a course timetable system for academic departments using genetic algorithm. *Jordan J. Comput. Inform. Technol.* 3(1): 25-36.

Atkin JA, Burke EK, Greenwood JS and Reeson D 2008 A metaheuristic approach to aircraft departure scheduling at London Heathrow airport. *Comput-aided Syst. Publ. Transport* 600: 235-252.

Augustine L, Faer M, Kavountzis A and Patel R 2009 A Brief Study of the Nurse Scheduling Problem (NSP). Pittsburgh, Carnegie Mellon School of Computer Science.

Aycan E and Ayav T 2009 Solving the course scheduling problem using simulated annealing. In 2009 *IEEE Int. Adv. Comput. Conf.* Patiala 2009: 462-466.

Baghel M, Agrawal S and Silakari S 2012 Survey of metaheuristic algorithms for combinatorial optimization. *International J. Comput. Applicat.* 58(19): 21-31.

Basir N, Ismail W and Norwawi NM 2013 A simulated annealing for Tahmidi course timetabling. *Proc. Technol.* 11: 437-445.

Birattari M, Paquete L, Strutzle T and Varrentrapp K 2001 Classification of Metaheuristics and Design of Experiments for the Analysis of Components. *Tech. Rep.* AIDA-01-05.

Carter MW and Laporte G 1998 Recent developments in practical course Timetabling. *Practice and Theory of Automated Timetabling*, Springer Verlag Berlin, 3-19.

Chacha S and Mushi A 2013 Optimal solution strategy for University Course timetabling problem. *International Journal of Advanced Research in Computer Science* 4(1): 35-40.

Festa P 2014 A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In *2014 16th International Conference on Transparent Optical Networks (ICTON)* IEEE, 1-20.

Garey MR, Johnson D 1979 Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Company.

Gonen B 2006 Genetic Algorithm finding the shortest path in Networks. Reno, University of Nevada.

Guan B and Zhao Y 2019 Self-adjusting ant colony optimization based on information entropy for detecting epistatic interactions. *Genes* 10(2): 114.

Kongkaew K 2017 Bat algorithm in discrete optimization: A review of recent applications. *Songklanakarin J. Sci. Technol.* 39(5): 641-650.

Masum AKM, Shahjalal M, Faruque F and Sarker IH 2011 Solving the vehicle routing problem using genetic algorithm. *International Journal of Advanced Computer Science and Applications* 2(7): 126-131.

Mujuni E and Mushi AR 2015 Solving the Examination Timetabling Problem Using a Two-Phase Heuristic: the case of Sokoine University of Agriculture. *J. Inform. Comput. Sci.* 10(3): 220-227.

Mushi AR 2006 Tabu search heuristic for university course timetabling problem. *Afr. J. Sci. Technol.* 7(1): 34-40.

Mushi AR 2011 Two-phase heuristic algorithm for the university course timetabling problem: the case of University of Dar Es Salaam. *Tanz. J. Sci.* 37(1): 73-83.

Osaba E, Yang XS, Diaz F, Lopez-Garcia P and Carballedo R 2016 An Improved Discrete Bat Algorithm for Symmetric and Asymmetric Traveling Salesman Problems. *Engin. Applicat. Artific. Intel.* 48: 59-71.

Osman IH and Kelly JP 1996 Meta-heuristics: an overview. *Meta-heuristics*, 1-21. Springer, Boston, MA.

Qin J, Ni LL and Shi F 2012 Combined simulated annealing algorithm for the discrete facility location problem. *Sci. World J.* 2012: 1-7.

Schaerf A 1999 A survey of Automated Timetabling. *Artificial Intelligence Review* 13(2): 87-127.

Shukl AN and Garg ML 2018 A List based Approach to Solve Graph Coloring Problem. In 2018 *International Conference on System Modeling and Advancement in Research Trends (SMART)* IEEE 265-267.

Srivastava S and Sahana SK 2019 Application of Bat Algorithm for Transport Network Design Problem. *Appl. Computat. Intel. Soft Comput.* 2019(5): 1-12.

Su JM and Huang JY 2006 Using ant colony optimization to solve train timetabling problem of mass rapid transit. In *9th Joint International Conference on Information Sciences (JCIS-06),* Atlantis Press, 1-4.

Wang Z, Geng X and Shao Z 2009 An effective simulated annealing algorithm for solving the travelling salesman problem. *J. Computat. Theoret. Nanosci.* 6(7): 1680-1686.

Winston WL, Venkataramanan M and Goldberg JB 2003 Introduction to mathematical programming, Duxbury; Pacific Grove, CA: Thomson/Brooks/Cole.

Wren A 1996 Scheduling, timetabling and rostering a special relationship? In *International Conference on the Practice and Theory of Automated Timetabling* (46-75), Springer, Berlin, Heidelberg.

Yang XS 2010a A new metaheuristic bat-inspired algorithm. Nature-inspired cooperative strategies for optimization (NICSO 2010), 65-74.

Yang XS 2010b Nature-inspired metaheuristic algorithms, Luniver Press, University of Cambridge, United Kingdom.

Zegordi SH and Jafari N 2010 Solving the airline recovery problem by using ant colony optimization. *Int. J. Industr. Engin. Product. Res.* 21(3): 121-128.

Zhang JW and Wang GG 2012 Image matching using a bat algorithm with mutation. In *Applied Mechanics and Materials* 203(1): 88-93). Trans Tech Publications Ltd.