Journal of Computational Science 5 (2014) 809–818

Contents lists available at ScienceDirect

Journal of Computational Science

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

University course timetabling using hybridized artificial bee colony with hill climbing optimizer

Asaju La'aro Bolaji[a,b,*], Ahamad Tajudin Khader[a], Mohammed Azmi Al-Betar[a,c], Mohammed A. Awadallah[a]

[a] School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia
[b] Department of Computer Science, University of Ilorin, Ilorin, Nigeria
[c] Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, P.O. Box 50, Al-Huson, Irbid, Jordan

ARTICLE INFO

Article history:
Received 31 December 2012
Received in revised form 18 January 2014
Accepted 8 April 2014
Available online 2 May 2014

Keywords:
Artificial bee colony
Timetabling problem
Nature-inspired computing
Swarm intelligence
Hybrid metaheuristic

ABSTRACT

University course timetabling is concerned with assigning a set of courses to a set of rooms and timeslots according to a set of constraints. This problem has been tackled using metaheuristics techniques. Artificial bee colony (ABC) algorithm has been successfully used for tackling uncapaciated examination and course timetabling problems. In this paper, a novel hybrid ABC algorithm based on the integrated technique is proposed for tackling the university course timetabling problem. First of all, initial feasible solutions are generated using the combination of saturation degree (SD) and backtracking algorithm (BA). Secondly, a hill climbing optimizer is embedded within the employed bee operator to enhance the local exploitation ability of the original ABC algorithm while tackling the problem. Hill climbing iteratively navigates the search space of each population member in order to reach a local optima. The proposed hybrid ABC technique is evaluated using the dataset established by Socha including five small, five medium and one large problem instances. Empirical results on these problem instances validate the effectiveness and efficiency of the proposed algorithm. Our work also shows that a well-designed hybrid technique is a competitive alternative for addressing the university course timetabling problem.

1. Introduction

University course timetabling problem (UCTP) is one of the hardest problems faced by academic institutions throughout the world. The UCTP is known to be a difficult combinatorial optimization problem that has been widely studied over the last few decades. The problem involves scheduling a given number of courses to a limited number of periods and rooms subject to satisfying a set of given constraints. The constraints in UCTP are usually classified into hard and soft constraints. The satisfaction of hard constraints is compulsory for the timetabling solutions to be *feasible*, while the satisfaction of the soft constraints is required but not essential. The quality of the UCTP solution is normally determined by the cost of violating the soft constraints. Basically, there are two forms of UCTP: curriculum-based course (CB-CTT) and post-enrolment course timetabling (PE-CTT); the latter is the concern of this research.

The UCTP has been tackled with series of proposition of optimization techniques by the researchers across the fields of operation research and artificial intelligence over the past few years. The problem has been surveyed in [1,2] where metaheuristic-based techniques have shown to be the most frequently used. The metaheuristics can be categorize into local search-based and population-based techniques [3]. Some examples of local search-based techniques that tackled UCTP include tabu search [4], simulated annealing [5,6], great deluge [7,8], variable neighbourhood structures (VNS) [9]. On the other hand, examples of population-based techniques used for UCTP are ant colony optimization [10,11], genetic algorithm (GA) [12,13], harmony search algorithm (HSA) [14,15], particle swarm optimization [16]. Similarly, hyper-heuristic approaches and hybrid metaheuristic approaches have also been used recently to tackle course timetabling problems [17]. Other approaches applied to UCTP include early heuristic approaches derived from graph colouring heurstics [18], constraint-based techniques [19], case-based reasoning [20]. Although the success of the metaheuristic algorithms when applied independently to optimization problems have

* Corresponding author at: School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia. Tel.: +2348146820888.
E-mail addresses: abl10_sa0739@student.usm.my (A.L. Bolaji), tajudin@cs.usm.my (A.T. Khader), mohbetar@cs.usm.my (M.A. Al-Betar), mama10_com018@student.usm.my (M.A. Awadallah).

http://dx.doi.org/10.1016/j.jocs.2014.04.002
1877-7503/

Table 1
The PE-CTP constraints.

	Hard constraints		Soft constraints
H_1	A student should not be assigned to more than one event at a time	S_1	A student should not have a class in the last slot of the day
H_2	The capacity of the room should be big enough for both the students and the event	S_2	A student should not have more than two classes consecutively
H_3	Only one event takes place in each room at any timeslot	S_3	A student should not have a single class in one day

been reported, substantial research efforts come from hybridization especially for large scheduling/timetabling problems. Few examples of hybridization between population-based and local search-based algorithms that were applied to timetabling problem appeared in [21,13,22]. Another hybrid technique which combined two population-based algorithms for real-world course timetabling (i.e. bee algorithm with harmony search algorithm) can be found in [23].

Artificial bee colony (ABC) algorithm is a nature-inspired search mechanism introduced by Karaboga in [24], based on concepts of simulating intelligence foraging behaviour of honey bee. Recently, researches and applications of ABC have been rapidly proposed and widely utilized. The methodology of ABC has been used to solve the complex problem in the field of science and engineering. Its introduction has brought a significant impact on many growing fields, such as artificial intelligence, decision analysis, knowledge management, image processing, pattern matching and recognition, product process design, scheduling and timetabling, and stock market analysis [25,26]. ABC was tailored in our previous work, for both curriculum-based course timetabling and uncapacitated examination timetabling problems with success stories [27–29]. To validate the success of the ABC in other scheduling/timetabling problems, in this paper, PE-CTP is being tackled with a hybridized artificial bee colony with hill climbing optimizer. This hybridization involves the combination of hill climbing optimizer (HCO) with the employed bee phase of original ABC. The main aim of this hybridization is to enhance the exploitation power of ABC by searching the PE-CTP solution space rigourously in order to obtain good solution.

The remainder of this paper is organized as follows: Section 2 gives the background descriptions and formulations of PE-CTP. Section 3 provides an overview of the original ABC, and Section 4 describes the proposed hybridization of HCO with ABC algorithm. Section 5 presents experimental analysis and discussions of the results of the proposed hybridized ABC, and compares the results with 24 other techniques using the Socha dataset. Section 6 presents the conclusion and future directions.

2. Post-enrolment-course timetabling problems

2.1. Problem descriptions

The descriptions of PE-CTP is given as an assignment of a set of courses to a set given timeslots and rooms in accordance with a set of given hard and soft constraints. The problem consists of three hard constraints which *must* be satisfied in order to obtain *feasible* solution i.e. (H_1, H_2, H_3) and three soft constraints (S_1, S_2, S_3) as shown in Table 1. The minimization of the soft constraints in a feasible timetable is the basic objective of PE-CTP.

2.2. Problem formulations

The PE-CTP consists of N events, each of which is attended by a number of students and demand specific room features; a set of M rooms, $\mathbb{R} = \{r_0, r_1, \ldots, r_{M-1}\}$ each with a specific capacity and features; a set of P timeslots[1] $\mathbb{T} = \{t_0, t_1, \ldots, t_{p-1}\}$, a set of Z room features, $\mathbb{F} = \{z_0, z_1, \ldots, z_{z-1}\}$, and a set of V students, $\mathbb{S} = \{s_0, s_1, \ldots, s_{v-1}\}$ each attending one or more events.

Table 2
The notation used in the PE-CTP formulations.

Symbols	Description
D	Total number of working days per week
H	Total number of timeslots for each working days
M	Total number of rooms
N	Total number of events
P	Total number of timeslots
V	Total number of Students
Z	Total number of features
$\mathbb{F}$	Set of features, $\mathbb{F} = \{z_0, z_1, \ldots, z_{z-1}\}$
$\mathbb{R}$	Set of rooms, $\mathbb{R} = \{r_0, r_1, \ldots, r_{M-1}\}$
$\mathbb{S}$	Set of students, $\mathbb{S} = \{s_0, s_1, \ldots, s_{v-1}\}$
$\mathbb{T}$	Set of timeslots, $\mathbb{T} = \{t_0, t_1, \ldots, t_{p-1}\}$
$\boldsymbol{x}$	The solution to the timetabling problem is give as $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$
b_i	The total of students taking event i
g_i	The capacity of room i
x_i	The room and timeslot of event i in the timetable
$a_{s,j}$	Student-day matrix element: whether student i assigned to only one event in day j $a_{s,j} = \begin{cases} 1 & \sum_{i=0}^{H-1} sa_{s,i+j\times H} = 1, \quad j \in [0, D-1]. \\ 0 & \text{otherwise.} \end{cases}$
$c_{i,j}$	Conflict matrix element: whether events i and j are in conflict with each other $c_{i,j} = \begin{cases} 1 & \text{if } u_{k,i} = 1 \wedge u_{k,j} = 1 \quad \exists k \in \mathbb{S}. \\ 0 & \text{otherwise.} \end{cases}$
$q_{i,j}$	Event-room matrix element: whether events i and room r_j are compatible in terms of features and size $q_{i,j} = \begin{cases} 1 & \text{if}(y_{j,k} \geq w_{i,k} \wedge (b_i \leq g_j) \quad \forall k \in \mathbb{F}. \\ 0 & \text{otherwise.} \end{cases}$
$sa_{j,k}$	Student availability matrix element: whether student s_j is in timeslot t_k in the timetable $\boldsymbol{x}$ $sa_{j,k} = \begin{cases} 1 & \exists x_i \in \boldsymbol{x}, k = x_i \mod P \wedge u_{j,i} = 1. \\ 0 & \text{otherwise.} \end{cases}$
$u_{i,j}$	Student-event matrix element: whether student s_i has attended an event j. $u_{i,j} = \begin{cases} 1 & \text{ifstudent } s_i \text{ attends event } j \\ 0 & \text{otherwise.} \end{cases}$
$w_{i,j}$	Event-feature matrix element: whether event i requires feature z_j. $w_{i,j} = \begin{cases} 1 & \text{if event } i \text{ requires feature } z_j \\ 0 & \text{otherwise.} \end{cases}$
$y_{i,j}$	Room-feature matrix element: whether Room r_i contains feature z_j. $y_{i,j} = \begin{cases} 1 & \text{if room } r_i \text{ includes feature } z_j \\ 0 & \text{otherwise.} \end{cases}$

Table 2 shows the notation for the formulation of the PE-CTP. The solution to PE-CTP is given by the vector $x = (x_1, x_2, \ldots, x_N)$ of courses, where the value of x_i is a map of room r_j with timeslot t_k for course i:

$$x_i = j \times P + k \tag{1}$$

For instance, if $\boldsymbol{x} = (590, \ldots)$, then event 1 is map to room 13 and timeslot 5 (where $P = 45$).

2.3. Definitions

The formal definitions of both hard and soft constraints are as follows:

Definition 1. The position l is feasible for event x_i to be assigned to the timetable $\boldsymbol{x}$ if and only if the following conditions are met:

[1] Generally, $P = 45$ (i.e. $D = 5$ working days, each with $H = 9$ timeslots).

- $H_1 : x_i \bmod P \neq x_j \bmod P, \quad \forall x_i, x_j \in \boldsymbol{x} \wedge c_{ij} = 1 \wedge i \neq j.$
- $H_2 : q_{i,j} = 1, \quad j = \lfloor \frac{x}{P} \rfloor.$
- $H_3 : x_i \neq x_j, \quad \forall x_i, x_j \in \boldsymbol{x} \wedge i \neq j.$

The formulations of hard constraints H_1, H_2, H_3 for each x_i are given in the above definition.

Definition 2. A timetable $\boldsymbol{x}$ is feasible if and only if $\forall x_i \in \boldsymbol{x}$ x_i is feasibly timetabled. It ensures that all events satisfy the hard constraints for defining a feasible timetable.

The following definitions are for the three soft constraints (S_1, S_2, S_3).

Definition 3. Last timeslot of a day (S_1). $\forall s \in \mathbb{S}$

$$f_1(x, s) = \sum_{j=0}^{D-1} sa_{s,H\times(j+1)-1}.$$

Definition 4. More than two events in a row (S_2). $\forall s \in \mathbb{S}$

$$f_2(x, s) = \sum_{i=0}^{D-1}\sum_{j=0}^{H-3} sa_{s,i\times H+j} \times sa_{s,i\times H+j+1} \times sa_{s,i\times H+j+2}.$$

Definition 5. A single event in a day (S_3). $\forall s \in \mathbb{S}$

$$f_3(x, s) = \sum_{j=0}^{D-1} a_{s,j}.$$

2.4. Objective cost

From the notations in Table 2, the timetabling solution $\boldsymbol{x}$ for the PE-CTP can be formally defined using the objective cost $f(x)$ which is given in Eq. (2). The value of $f(x)$ is the total number of soft constraints violated in the feasible timetable:

$$f(x) = \sum_{s\in\mathbb{S}} (f_1(\boldsymbol{x}, s) + f_2(\boldsymbol{x}, s) + f_3(\boldsymbol{x}, s)) \tag{2}$$

The value of $f(x)$ is referred to as the penalty cost of a feasible timetable. Note that the above descriptions and formulations are adopted from [15] with some modifications.

3. Artificial bee colony algorithm

The artificial bee colony (ABC) algorithm is a swarm intelligence meta-heuristic algorithm based on the model first proposed in [30] on the foraging behaviour of honey bee colonies. The model comprises three important elements: employed, unemployed foragers and food sources. The employed and unemployed foragers are the first two elements, while the third element is the rich food source which is closer to their hive. The two leading modes of behaviours are also described by the model. These behaviours are necessary for self-organization and collective intelligence: recruitment of foragers to rich food sources which results into positive feedback and simultaneously, the abandonment of poor sources by foragers, which causes negative feedback [31,32].

In the ABC algorithm, the artificial foragers are grouped into three categories: the employed, the onlooker and the scout foragers. A forager that is exploiting a food source is classified as employed. The employed foragers share information with the onlookers, which are waiting in the hive and watching the dance of the employed foragers. The onlooker foragers then select a food source with probability proportional to the quality of that food source. In other words, foragers tend to move toward good food sources than the bad ones. Scout foragers known as colony explorer search for a new food source randomly in the vicinity of the hive. When a scout forager finds a good food source, it turn into employed. When a food source has been fully exhausted, the employed forager in charge of it will abandon the position, and automatically turn to a scout. Therefore, it is interesting to note that the scout bee perform the job of "exploration", whereas employed and onlooker bees perform the job of "exploitation". In the optimization context, a food source corresponds to a possible solution to the optimization problem, and the nectar amount of a food source corresponds to the fitness of the associated solution.

The search cycle of ABC consists of three rules: (i) sending the employed foragers to a food source and evaluating the nectar quality; (ii) onlookers choosing the food sources after obtaining visual information from employed foragers and calculating their nectar quality; (iii) determining the scout foragers and then sending them onto possible food sources. The positions of the food sources are randomly selected by the foragers at the initialization stage and their nectar qualities are measured. The employed foragers then share the nectar information of the sources with the bees waiting at the dance area within the hive. At the next stage after sharing information, every employed bee returns to the food source visited at the previous cycle, since the position of the food source exists in its memory and then selects a food source using the visual information in the neighborhood of the present one. At the last stage, an onlooker uses the information obtained from the employed bees at the dance area to select a food source. The probability with which the food source is selected increases as the nectar quality of a food source increases as well. Therefore, the employed forager with higher nectar quality information recruits the onlooker to that food source. It subsequently chooses a food source in the neighborhood of the one in her memory based on visual information (i.e. comparison of food source positions). A new food source is randomly generated by a scout bee to replace a food source which has nectar quality abandoned by the onlooker foragers. This search procedure could be represented in algorithm (1) as follows:

Algorithm 1. Schematic pseudocode of ABC procedure

```
Initialize the ABC and problem parameters
Initialize the Food Source Memory (FSM)
repeat
  Send the employed foragers to the food sources.
  Send the onlookers to select a food source.
  Send the scouts to search possible new food sources.
  Memorize the best food source.
until (termination criterion are met)
```

The detailed description of the ABC search procedure could be found in our previous work on the survey of ABC [25].

4. A hybridized ABC with hill climbing for PE-CTP

This section gives description of hybridized ABC with hill climbing for PE-CTP. Firstly, the description of Basic-ABC as presented in our previous work shall be briefly discussed; then it will be followed by hybridization of hill climbing optimizer with employed bee operator of ABC.

4.1. ABC algorithm for PE-CTP

The implementation of ABC for the PE-CTP includes the six steps as described in following sections.

4.1.1. Initialize the ABC and PE-CTP parameters

In this step, the three control parameters of ABC that are needed for tackling PE-CTP are initialized. These parameters include solution number (SN) which represents the number of food sources in the population and it is similar to the population size in GA; maximum cycle number (MCN) refers to the maximum number of

iterations; and *limit* which is normally used to diversify the search and it is responsible for the abandonment of solution if there is no improvement for a certain number of iterations. Similarly, the PE-CTP parameters like a set of the courses, a set of rooms, a set of timeslots and a set of constraints are also extracted from the dataset. Note that the courses are the main decision variable that can be scheduled to feasible positions (i.e. rooms and timeslots) in the timetable solution.

4.1.2. Initialize the food source memory

The food source memory (FSM) is a memory allocation that contains sets of feasible food source (i.e. solutions) vectors which are determined by *SN* as shown in Eq. (3). Here, the food source vectors are generated with the aid of largest weighted degree (LWD) [33] followed by backtracking algorithm (BA) [34] and they are sorted in ascending order in the FSM in accordance with their objective function values, that is $f(x_1) \leq f(x_2) \ldots f(x_{SN})$

$$FSM = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^N \\ x_2^1 & x_2^2 & \cdots & x_2^N \\ \vdots & \vdots & \ddots & \vdots \\ x_{SN}^1 & x_{SN}^2 & \cdots & x_{SN}^N \end{bmatrix} \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_{SN}) \end{bmatrix} \quad (3)$$

Note that in order to generate a feasible solution (i.e. food source) for FSM, the LWD starts with an empty timetable solution where the course with the largest number of conflicting students in the schedule is assigned first. Whenever LWD fails to assign all courses into the timeslots and rooms because of earlier assignments, then the backtracking algorithm (BA) is applied to re-assign unscheduled courses.

Firstly, all courses that cannot be scheduled to the solution would be entered into a list called (A). The backtracking algorithm would then select each unassigned course (c) from list (A) and explore all courses in conflict in the timetabling. Those courses that are in conflict with course (c) are removed from the timetable and added to (A) again. Subsequently, the backtracking algorithm attempts to re-assign timeslots and rooms to all courses in the list. This process is repeated several times until no further courses could be scheduled, in which case the LWD, followed by the BA processes are repeated from the beginning until all the courses are assigned. It is important to note that this process is similar to what was used in [15]. The pseudocode for generating feasible solutions for the PE-CTP is given in Algorithm (2).

Algorithm 2. The pseudocode for generating feasible solutions

```
while j ≤ SN do
  repeat
    x^j = φ
    LWD(x^j)
    if (x^j is not complete and predefined iterations are not met) then
      Backtracking (x^j)
    end if
  until (x^j is complete)
  save x^j in FSM
  evaluate f(x^j)
end while
```

4.1.3. Send the employed bee to the food sources

Here, the timetable solutions are sequentially selected from the FSM by the employed forager operator. Each solution is perturbed using three neighbourhood structures to produce a new set of solutions. Three neighbourhood structures used by employed bees are:

1 *Neighbourhood Move (NMove)*: moves selected course to a feasible period and room randomly i.e. replace the time period x_i' of course i by another feasible timeslot.

2 *Neighbourhood Swap (NSwap)*: swap two selected course at random i.e. select course i and course j randomly, swap their time periods (x_i', x_j').

3 *Neighbourhood Kempe Chain (NkempeChain)*: Firstly, select the timeslot x_i' of course i and randomly select another q' timeslot. Secondly, all courses that have the same timeslot x_i' that are in conflict with one or more courses timetabled in q_i are entered to chain δ where $\delta = \{j | x'_j = x'_i \wedge t_{i,q'} = 0 \wedge \forall j \in E\}$. Thirdly, all courses that have the same timeslot q' that are conflicting with one or more courses timetabled in x_i' are entered to a chain δ' where $\delta' = \{k | x'_k = q' \wedge t_{k,x_i'} = 0 \wedge \forall k \in E\}$ and Lastly, simply assign the courses in δ to q' and the courses in δ' to x_i'.

The fitness of each new food source is calculated; if it is better than that of old food source, then it replaces the parent food source in FSM. This process is implemented for all solutions (see Algorithm 3 for details).

Algorithm 3. Employed bee phase

```
for i = 1 ··· SN do
  i = RND() {RND generates a random integer number in range 1–3}
  if i = 1 then
    x^{i(new)} = NMove( x^i )
  else
    if i = 2 then
      x^{i(new)} = NSwap( x^i )
    else
      if i = 3 then
        x^{i(new)} = NKempeChain( x^i )
      end if
    end if
  end if
  if x^{i(new)} is better than x^i then
    x^i = x^{i(new)}
  end if
  next i
end for
```

4.1.4. Send the onlooker bee

The onlookers have the same number of food sources (timetabling solution) as the employed foragers. It initially calculates the selection probability of each food source generated by the employed forager in the previous step. The fittest food source is selected by the onlooker using roulette wheel selection mechanism. The process of selection in the onlooker phase works thus:

- assign for each employee bee a selection probability as follows:

$$p_j = \frac{f(\boldsymbol{x}^j)}{\sum_{k=1}^{SN} f(\boldsymbol{x}^k)}$$

  Note that the $\sum_{i=1}^{SN} p_i$ is unity.
- the onlooker samples and selects the fitness of the food source of each employed forager and selects the one with the highest probability. It then adjusts the selected food source to its neighbourhood using the same strategy as the employed bee. The fitness of the new solution is calculate and if it is better, then it will replace the current one.

4.1.5. Send the scout to search for possible new food sources

This is known to be the colony explorer. It works once a solution is abandoned, i.e. if a solution in the FSM has not improved for certain number of iterations. ABC generates a new solution randomly to substitutes the abandoned one. Memorize the fitness of the best food source x_{best} found so far in FSM.

4.1.6. Stopping condition

Steps 3–5 are repeated until a stop criterion is met. This is originally determined using *MCN* value.

4.2. Hybridization of hill climbing optimizer with ABC

Here, a brief description of hill climbing optimizer (HCO) shall be discussed in the next bsection, followed by procedure of its hybridization with the employed forager of the classical ABC.

4.2.1. Hill climbing optimizer (HCO)

Hill climbing is a member of the local search-based techniques which can be used to direct the search toward the local optima. For example, in HCO, the search direction of an individual should be moved towards local optimum, and computation of the fitness is done at each step. The hybridization of HCO with ABC presented in this paper is similar to the memetic algorithms (MAs), which have been mostly applied to timetabling problems by hybridizing hill-climbing optimizer into the operators of the genetic algorithm (GA) [13]. It is noteworthy to mention that the hybridizing HCO to population-based techniques for timetabling problems seems to be computationally expensive. However, if there is reduction in computation time and/or there is enhancement in the quality of solution, then the hybridization can be preferable.

4.2.2. Hybridized ABC algorithm with HCO

In this study, hill climbing optimiser (HCO) is hybridized with the ABC as a new operator to improve the food source vector relative to the maximum cycle number (MCN). A hybridized ABC follows similar to what was done in our previous works on the basic ABC [28,29] except on the employed forager phase where a hybridized ABC utilized HCO in its operation to optimize the navigation of the search. It is worth noting that in this research the feasibility of the solution search space is highly considered.

The ABC starts with a population of feasible solutions (food sources) generated at initialization stage. In each cycle, each employed forager applied the HCO to exploit the neighbourhood of the existing solutions sequentially and moves to another neighbouring solution that has a better or equivalent objective cost value. Indeed, HCO randomly selects three neighbourhood structures described in Section 4.1.3 to produce a new food source from the neighbourhood of the existing one.

The fitness of each offspring solution produced by the HCO operator is calculated; if it is better than that of parent food source, then it replaces the parent food source in FSM. This process is implemented for all solutions. Algorithm 4 shows how the HCO is being invoked within the employed forager of ABC as a new operator. Each employed forager uses HCO to exploit the neighbourhood of the food source in their memory by fine-tuning it in order to obtain a new one with probability defined by the "hill climbing rate (HCR)" parameter. Note that the HCR parameter is used to control the usage of hill climbing optimizer. The pseudocode of the HCO operator as used in Algorithm 4, is given in Algorithm 5. During the search process, the function *Search* ($\mathfrak{N}(x)$) explores the neighbouring solutions $\mathfrak{N}(x)$ of $\boldsymbol{x}$, and accepts the neighbouring solution $x' \in N(x)$ which has an equal or lower penalty cost, i.e.,$f(x') \leq f(x)$. The search process is repeated until no further improvement is achieved. The function *Search* ($\mathfrak{N}(x)$) exploits the UCTP search space of $\boldsymbol{x}$ with aid neighbourhood structures discussed in Section 4.1.3.

Table 3
Characteristics of PE-CTP dataset.

Class	Small	Medium	Large
Number of events	100	400	400
Number of rooms	5	10	10
Number of features	5	5	10
Number of students	80	200	400
Number of timeslots	45	45	45
Approximate feature per room	3	3	5
Percentage of feature use	70	80	90
Max. no. of events per student	20	20	20
Max. no. of students per event	20	50	100

Algorithm 4. The ABC invoked HCO procedure

```
Initialize (FSM)
repeat
  for i = 1 ··· SN do
    if (f(U(0, 1) < HCR)) then
      x_i(new) = HCO( x_i )
    end if
  end for
  Memorize the best food source found so far
until (the Maximum Cycle Number is met)
```

Algorithm 5. The HCO procedure

```
Input ( x) solution
repeat
  x' = Search(N(x))
  if (f( x') ≤ f( x)) then
    x = x'
  end if
until (local minima solution is achieved)
```

5. Experimental results and discussion

The performance of the proposed hybridized-ABC for PE-CTP is evaluated in this section. The proposed technique is coded in Microsoft Visual C++6.0 on Windows 7 platform on Intel 2 GHz Core 2 Quad processor with 2 GB of RAM. The performance of the proposed method is tested using the dataset established by Socha et al. [10]. This dataset is published at website.[2] The dataset comprises 100–400 courses that are needed to be assigned to a timetable with 45 timeslots corresponding to 5 days of 9 hours each, at the same time satisfying room features and room capacity constraints. They are divided into three types: small, medium and large (i.e. 5 small, 5 medium and 1 large instances) as shown in Table 3.

5.1. Experimental design

In this section, the experiments showing the effectiveness of proposed hybridized-ABC are presented to study the influence of the HCO. Note that the role of HCO is important in fine-tuning the search space region of the food sources (solutions) toward the local minima by enhancing the local exploitation capability of the proposed technique. The study that investigates the effect of varying the hill climbing rate (HCR) parameter on the performance of hybridized-ABC is conducted. The five values of HCR experimented are 0%, 10%, 25%, 50% and 90%, respectively. The details of the parameter settings for the proposed hybridized-ABC for PE-CTP are given in Table 4. The settings of these parameters were selected based on our preliminary experiments over PE-CTP. Note that the solution number *(SN)* was set to 10 and limit to 100. It is worthy to mention that these settings should not be considered as best optimal values of the parameters. However, they can be considered as a generalised set of parameter values since the performance of the hybridized-ABC is fairly well over the PE-CTP.

[2] http://iridia.ulb.ac.be/ msampels/tt.data/.

Table 4
The Hybridized-ABC convergence cases.

Case	Solution number (SN)	Limit	Hill climbing rate (HCR)	MCN
Case 1	10	100	0%	2000
Case 2	10	100	10%	2000
Case 3	10	100	25%	2000
Case 4	10	100	50%	2000
Case 5	10	100	90%	2000

The proposed hybridized-ABC with HCO took between 6 min and 7 h to obtain the recorded results. This is also in line with time taken by HBMO for ETP proposed by [35]. It is worth mentioning here that this runtime is not a major factor in university timetabling problems because the productions of timetables usually come once or twice a year [36]. However, the minimum and maximum time taken by the proposed hybrid technique is recorded in Table 5.

Table 6 shows the results of the proposed Hybridized-ABC with varying HCR value by showing the best, mean, worst and standard deviation over ten runs. The best solution for each problem instance is highlighted in bold. The results showed that the HCR with lower value generally improves the solutions that are obtained. As shown in Table 6, the results produced by the Hybridized-ABC when the value HCR is small (i.e. HCR = 10%), the results produced are more promising than when its value is higher (i.e. HCR= 90%) especially for medium and large problem instances. This is because balance between the exploitation and exploration during the search might properly achieved. However, when the value of HCR is set at 90%, the results seems to be worst because the proposed algorithm tend to be more exploiter which allowed it to be attracted to the best solution in the population and thus prevent it from escaping the local optima".

Fig. 1(a–f) shows convergence characteristic plots for varying HCR parameter in terms of the penalty cost for medium 1 through large problem instances. The *x*-axis represents the number of iterations, while the *y*-axis represents the penalty cost. Each point of the graphs corresponds to the penalty cost and the number of generations (refer to the curve of the best food source). As shown in this figure, the penalty cost is mostly decreased until reaches the near optimal value. The penalty cost is rapidly reduced at the beginning of the search due to the availability of the rooms and timeslots in the beginning which helps to improve the solution. It also can be shown from these figures; the Hybridized-ABC algorithm is able to achieve a high improvement at the beginning of the search for the medium and large problem instances, while towards end of runs the proposed method almost stagnated. This is because the hill climbing optimizer attracted the solutions into the local optima during the search and the most advantage of hill climbing is gained at the initial stage of the search.

Table 5
The minimum and maximum runtime taken by the proposed ABC on each problem instance on all experiment (in seconds).

Dataset	Minimum runtime (s)	Maximum runtime (s)
Small 1	312	1211
Small 2	292	1661
Small 3	320	1584
Small 4	331	1709
Small 5	211	1203
Medium 1	2813	7642
Medium 2	2944	8011
Medium 3	2715	9324
Medium 4	2977	8983
Medium 5	3310	10,081
Large	6712	19,189

Table 6
The Hybridized-ABC convergence cases.

Dataset		Case 1	Case 2	Case 3	Case 4	Case 5
Small 1	Best	2	**0**	**0**	1	**0**
	Mean	6.60	0.60	0.70	1.70	1.50
	Worst	13	2	2	3	3
	Stdv	4.25	0.70	0.67	0.67	0.97
Small 2	Best	2	**0**	**0**	**0**	**0**
	Mean	6.60	0.90	1.00	1.30	2.60
	Worst	15	2	3	3	4
	Stdv	3.84	0.74	1.15	1.06	1.43
Small 3	Best	4	**0**	**0**	**0**	**0**
	Mean	7.40	1.50	0.90	2.50	2.00
	Worst	13	3	3	4	3
	Stdv	2.80	1.35	1.10	1.43	1.05
Small 4	Best	7	**0**	1	1	**0**
	Mean	9.70	1.70	2.30	3.00	2.60
	Worst	15.00	3	3	5	5
	Stdv	2.98	1.16	0.95	1.33	1.78
Small 5	Best	1	**0**	**0**	**0**	**0**
	Mean	5.30	0.30	0.00	0.10	0.00
	Worst	8	1	0	1	0
	Stdv	2.50	0.48	0.00	0.32	0.00
Medium 1	Best	171	76	80	**73**	93
	Mean	199.50	100.20	108.60	111.70	113.70
	Worst	251	109	130	143	143
	Stdv	23.35	11.04	17.64	19.21	15.39
Medium 2	Best	166	88	96	**79**	103
	Mean	192.00	104.70	113.20	102.50	115.40
	Worst	220	125	143	118	143
	Stdv	17.80	13.28	16.23	14.98	14.39
Medium 3	Best	186	**137**	159	151	145
	Mean	213.80	171.10	169.50	176.90	171.90
	Worst	238	211	199	205	211
	Stdv	15.78	27.29	11.52	19.39	26.28
Medium 4	Best	169	**69**	86	81	85
	Mean	200.00	92.50	108.90	108.60	171.90
	Worst	237	107	133	127	211
	Stdv	21.98	12.00	14.04	17.39	26.28
Medium 5	Best	113	**61**	76	69	65
	Mean	140.10	87.00	113.30	84.40	95.20
	Worst	173	128	150	132	120
	Stdv	18.80	23.81	19.66	19.89	16.65
Large	Best	559	**462**	491	487	521
	Mean	604.00	509.70	553.60	570.10	569.90
	Worst	653	601	577	638	635
	Stdv	32.95	38.07	28.35	52.34	47.52

5.2. Comparative analysis with previous techniques

The proposed hybridization of HCO with ABC is compared with some previous techniques that work on Socha dataset, known and available to the authors. They consist of 11 swarm intelligence related techniques and 13 other heuristic methods as shown in Table 7. The results of the hybridized-ABC in comparison with previous techniques from literature are given in Tables 8 and 9. The numbers in the table signify the number of soft-constraint violations which define the penalty cost. The sign "+" indicates where the technique could not provide a feasible timetable solution (e.g., satisfaction of hard constraints was not achieved). The numbers highlighted in bold show the best solution that is obtained for that problem instance (lowest is best). As shown in Tables 8 and 9, the proposed method achieves a zero penalty cost for all five small problem instances, as achieved by a number of other previous techniques. For the medium problem instance, out of the 25 techniques that are evaluated (including hybridized ABC), the proposed method has ranked first in medium 1 and 5, second in medium 2 and 4, and fourth in medium 3. The experimental results show

Table 7
Key to the comparator techniques.

No.	Key	Technique	Reference
1	Hybridized-ABC	Hybridization of ABC with HCO	Proposed method
Swarm intelligence techniques			
2	MMAS	MAX-MIN Ant System	[10]
3	HMBO-ETP	Honey Bee Mating Optimisation Algorithm	[35]
4	HEA	Hybrid Evolutionary Approach	[37]
5	ENGD	Evolutionary Non-Linear Great Deluge	[38]
6	EMGD	Electromagnetism Mechanism Great Deluge	[39]
7	DHCA	Die Hard Cooperative Ant	[40]
8	EGSGA	Extended Guided Search Genetic Algorithm	[13]
9	GGA	Guided Genetic Algorithm	[41]
10	EASA	An Elitist-Ant System	[42]
11	ACS-SA	Hybrid Ant Colony Systems	[43]
12	HMA	Hybrid Metaheuristic Approach	[22]
Other heuristic techniques			
13	HHSA	Hybrid Harmony Search Algorithm	[21]
14	RRLS	Random Restart Local Search	[10]
15	VNS	Variable Neighbourhood Search	[9]
16	THH	Tabu search Hyper-heuristics	[44]
17	FMHO	Fuzzy Multiple Heuristic Ordering	[45]
18	GHH	Graph-based Hyper-heuristics	[17]
19	DCFHH	Distributed Choice Function Hyper-heuristics	[46]
20	RII	Randomized Iterative Improvement	[47]
21	GD	Great Deluge	[48]
22	NGD	Non Linear Great Deluge	[48]
23	GDTS	Great Deluge and Tabu Search	[8]
24	PCA	Particle Collision Algorithm	[49]
25	LARD	Late Acceptance Randomized Descent	[50]

Table 8
The best results achieved by hybridized-ABC and other swarm intelligence techniques.

	Small 1	Small 2	Small 3	Small 4	Small 5	Medium 1	Medium 2	Medium 3	Medium 4	Medium 5	Large
Hybridized-ABC	**0**	**0**	**0**	**0**	**0**	**73**	**79**	132	69	**61**	**462**
MMAS	1	3	1	1	**0**	195	184	284	164.5	219.5	851.5
HMBO-ETP	**0**	**0**	**0**	**0**	**0**	75	88	129	74	**61**	523
HEA	**0**	**0**	**0**	**0**	**0**	221	147	246	165	130	529
ENGD	**0**	1	**0**	**0**	**0**	126	123	185	116	129	821
EMGD	**0**	**0**	**0**	**0**	**0**	96	96	135	79	87	683
DHCA	5	5	3	3	**0**	176	154	191	148	166	798
EGSGA	**0**	**0**	**0**	**0**	**0**	139	92	**122**	98	116	615
GGA	**0**	**0**	**0**	**0**	**0**	240	160	242	158	124	801
EASA	**0**	**0**	**0**	**0**	**0**	84	82	123	**62**	75	690
ACS-SA	**0**	**0**	**0**	**0**	**0**	117	121	158	124	134	647
HMA-SA	**0**	**0**	**0**	**0**	**0**	96	96	135	79	87	683

Table 9
The best results achieved by hybridized-ABC and other heuristic techniques.

	Small 1	Small 2	Small 3	Small 4	Small 5	Medium 1	Medium 2	Medium 3	Medium 4	Medium 5	Large
Hybridized-ABC	**0**	**0**	**0**	**0**	**0**	**73**	79	**132**	**69**	**61**	**462**
HHSA	**0**	**0**	**0**	**0**	**0**	99	**73**	135	112	87	498
RRLS	8	11	8	7	5	199	202.5	+	177.5	+	+
THH	1	2	**0**	1	**0**	146	173	267	169	303	1166
VNS	**0**	**0**	**0**	**0**	**0**	317	313	375	247	292	+
FMHO	10	9	7	17	7	243	325	249	285	132	1138
DCFHH	1	3	1	1	**0**	182	164	250	168	222	+
GHH	6	7	3	3	4	372	419	359	348	171	1068
RII	**0**	**0**	**0**	**0**	**0**	242	161	265	181	151	+
GD	17	15	24	21	5	201	190	229	154	222	1066
NGD	3	4	6	6	4	140	130	189	112	141	876
GDTS	**0**	**0**	**0**	**0**	**0**	78	92	135	75	68	556
MPCA	**0**	**0**	**0**	**0**	**0**	105	108	156	84	141	719
LARD	**0**	**0**	**0**	**0**	**0**	149	132	200	138	173	855
EGD	**0**	**0**	**0**	**0**	**0**	80	105	139	88	88	730
MA	**0**	**0**	**0**	**0**	**0**	227	180	235	142	200	+

that the hybridized-ABC has been able to achieve comparably good results for this class of dataset. Similarly, for the hardest dataset, the hybridized ABC is able to provide a feasible timetable and ranked best among the comparative techniques.

Note that our proposed method had a better performance on medium 1, 2, 4, 5 and large instances. This is because the hybridization of hill climbing optimizer (HCO) within the ABC operator is to provide a more structured exploitation through the gradient

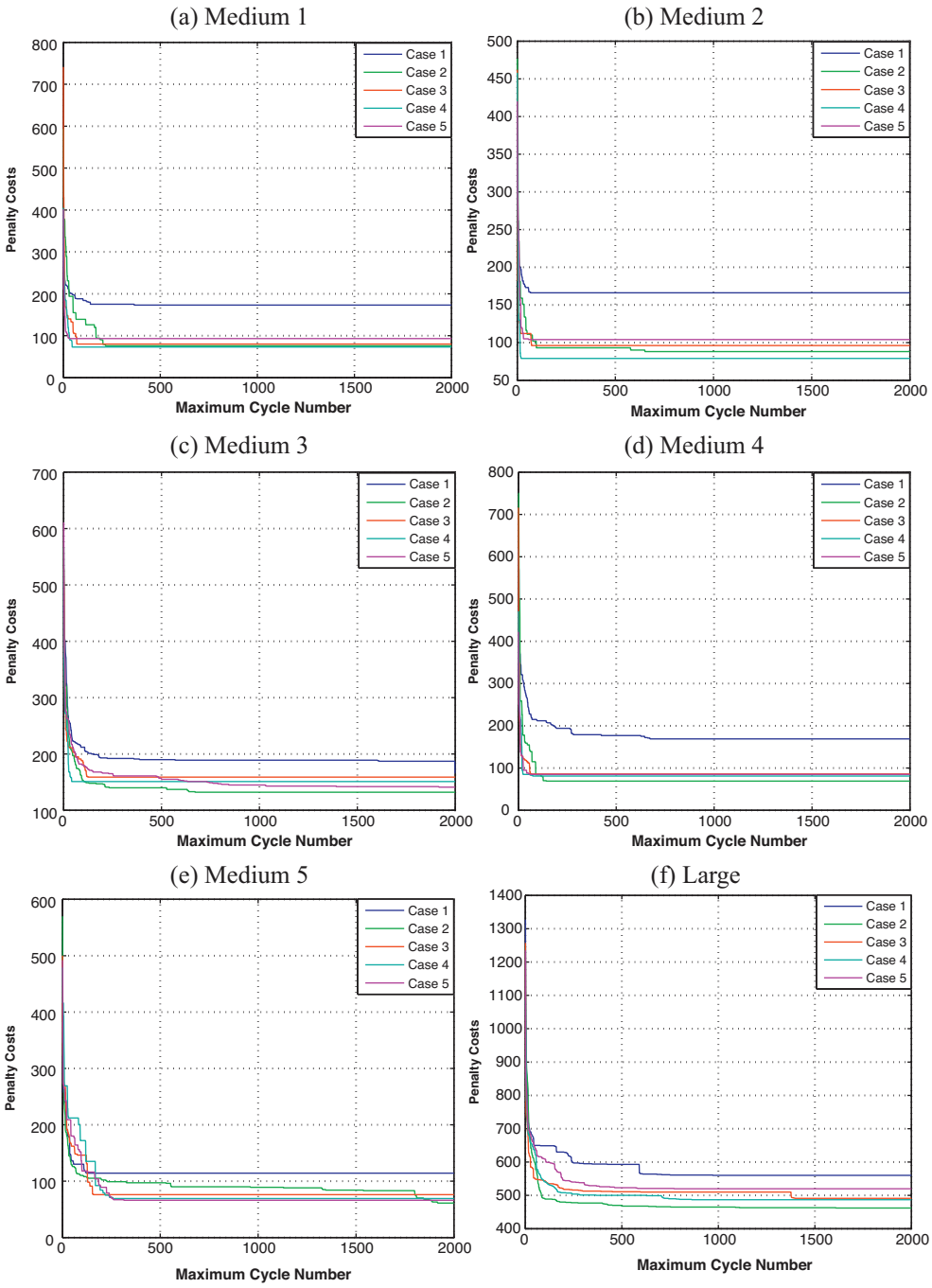

Fig. 1. Comparing the average penalty costs of varying the HCR parameter against 2000 *MCN* for medium 1 through large problem instances.

descent which aided a balance between global exploration and local exploitation of the timetabling solution search space. Thus hybridization of HCO with the employed bee component of the ABC improves the solution in FSM greatly. Note that in hybridized-ABC, the determination of the rate of gradient descent is carried out by hill climbing optimizer (HCO) whereas in the ABC algorithm, the rate of gradient descent is determined by neighbourhood structures.

6. Conclusion

This paper tackled the PE-CTP using a hybridization of the hill climbing optimizer (HCO) within the employed forager operator of ABC algorithm, named hybridized-ABC. Experimental analysis of the proposed technique showed some improvement in performance over previous techniques. The technique was evaluated with 11 problem instances introduced by Socha et al. [10] comprises 5 small, 5 medium, and 1 large. The experimental results showed that the proposed hybridized-ABC achieved a feasible timetable solution with no constraint violations for all the small problem instances. Similarly, for the medium and large datasets, the technique is able to obtain high quality solutions in comparison with other 25 techniques that tackled the problems. Future research work will include investigating the performance of the proposed hybridized-ABC to other scheduling/timetabling problems such as examinations, nurse scheduling etc. The performance of the proposed method can be improved by trying other efficient and recent local search methods such as Simulated Annealing, Great Deluge,

Tabu Search to escape local optima in order to improve the produced results.

Acknowledgements

The authors wish to express their since gratitude to the anonymous referees for their helpful and insightful comments, which have greatly improved the clarity of the paper. The first author would like to thank Universiti Sains Malaysia for the financial support under USM fellowship scheme for his PhD study and USM Postdoctoral Research Fellowship awarded to the third author.

References

[1] B. McCollum, A perspective on bridging the gap between theory and practice in university timetabling, Pract. Theory Automat. Timetabling VI (2007) 3–23.
[2] R. Lewis, A survey of metaheuristic-based techniques for university timetabling problems, OR Spectrum 30 (1) (2008) 167–190.
[3] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2003) 268–308.
[4] C. Aladag, G. Hocaoglu, M. Basaran, The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem, Exp. Syst. Appl. 36 (10) (2009) 12349–12356.
[5] S. Abdullah, K. Shaker, B. McCollum, P. McMullan, Dual sequence simulated annealing with round-robin approach for university course timetabling, Evol. Comput. Comb. Opt. (2010) 1–10.
[6] P. Kostuch, The university course timetabling problem with a three-phase approach, in: E. Burke, M. Trick (Eds.), in: Practice and Theory of Automated Timetabling (PATAT) V, Vol. 3616, 2005, pp. 109–125.
[7] P. Mcmullan, An extended implementation of the great deluge algorithm for course timetabling, Comput. Sci. – ICCS 2007 (2007) 538–545.
[8] S. Abdullah, K. Shaker, B. McCollum, P. McMullan, Construction of course timetables based on great deluge and tabu search, in: Metaheuristics Int. Conf., VIII Meteheuristic, 2009, pp. 13–25.
[9] S. Abdullah, E. Burke, B. Mccollum, An investigation of variable neighbourhood search for university course timetabling, in: The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), 2005, pp. 413–427.
[10] K. Socha, J. Knowles, M. Sampels, A max–min ant system for the university course timetabling problem, Ant Algorithms (2002) 63–77.
[11] K. Socha, M. Sampels, M. Manfrin, Ant algorithms for the university course timetabling problem with regard to the state-of-the-art, Appl. Evol. Comput. (2003) 334–345.
[12] W. Erben, J. Keppler, A genetic algorithm solving a weekly course-timetabling problem, Pract. Theory Automat. Timetabling (1996) 198–211.
[13] S. Yang, S. Jat, Genetic algorithms with guided and local search strategies for university course timetabling, IEEE Trans. Syst. Man Cybernet. C, Appl. Rev. 41 (1) (2011) 93–106.
[14] M.A. Al-Betar, A.T. Khader, M. Zaman, University course timetabling using a hybrid harmony search metaheuristic algorithm, IEEE Trans. Syst. Man Cybernet. C, Appl. Rev. (2014), http://dx.doi.org/10.1109/TSMCC.2011.2174356.
[15] M.A. Al-Betar, A.T. Khader, A harmony search algorithm for university course timetabling, Ann. Oper. Res. 194 (1) (2012) 3–31.
[16] S. Irene, S. Deris, M. Zaiton, A study on PSO-based university course timetabling problem, in: IEEE International Conference on Advanced Computer Control, 2009, pp. 648–651.
[17] E. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyper-heuristic for educational timetabling problems, Eur. J. Oper. Res. 176 (1) (2007) 177–192.
[18] E. Burke, G. Kendall, M. Mısır, E. Özcan, E. Burke, G. Kendall, E. Özcan, M. Mısır, Applications to timetabling, in: Handbook of Graph Theory, Citeseer, 2004 (chapter 5).
[19] S. Daskalaki, T. Birbas, E. Housos, An integer programming formulation for a case study in university timetabling, Eur. J. Oper. Res. 153 (1) (2004) 117–135.
[20] E. Burke, S. Petrovic, R. Qu, Case-based heuristic selection for timetabling problems, J. Sched. 9 (2) (2006) 115–132.
[21] M.A. Al-Betar, A.T. Khader, A hybrid harmony search for university course timetabling, in: Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2009), 10–12 August 2009, Dublin, Ireland, 2009, pp. 157–179.
[22] S. Abdullah, H. Turabieh, B. McCollum, P. McMullan, A hybrid metaheuristic approach to the university course timetabling problem, J. Heurist. 18 (1) (2012) 1–23.
[23] K. Nguyen, P. Nguyen, N. Tran, A hybrid algorithm of harmony search and bees algorithm for a university course timetabling problem, Int. J. Comput. Sci. Issues 9 (1) (2012) 12–17.
[24] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Techn. Rep. TR06, Erciyes Univ. Press, Erciyes, 2005.
[25] A. Bolaji, A. Khader, M. Al-Betar, M. Awadallah, Artificial bee colony, its variants and applications: a survey, J. Theoret. Appl. Inform. Technol. 47 (2) (2013) 434–459.
[26] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (abc) algorithm and applications, Artif. Intell. Rev. (2012) 1–37.
[27] A. Bolaji, A. Khader, M. Al-Betar, M. Awadallah, An improved artificial bee colony for course timetabling, in: Sixth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), IEEE, 2011, pp. 9–14.
[28] A. Bolaji, A. Khader, M. Al-Betar, M. Awadallah, Artificial bee colony algorithm for solving educational timetabling problems, Int. J. Nat. Comput. Res. 3 (2) (2012) 1–21.
[29] A. Bolaji, A. Khader, M. Al-betar, M. Awadallah, The effect of neighborhood structures on examination timetabling with artificial bee colony, in: Pract. Theory Automat. Timetabling IX, 2012, pp. 131–144.
[30] D. Teodorović, M. DellOrco, Bee colony optimization – a cooperative learning approach to complex transportation problems, in: Proceedings of the 10th Meeting of the EURO Working Group on Transportation Advanced OR and AI Methods in Transportation, Poznan, Poland, Citeseer, 2005, pp. 51–60.
[31] D. Karaboga, B. Akay, Artificial bee colony (ABC) algorithm on training artificial neural networks, in: IEEE 15th Signal Processing and Communications Applications (SIU 2007), 2007, pp. 1–4.
[32] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[33] T. Arani, V. Lotfi, A three phased approach to final exam scheduling, IIE Trans. 21 (1) (1989) 86–96.
[34] M. Carter, G. Laporte, S. Lee, Examination timetabling: algorithmic strategies and applications, J. Oper. Res. Soc. 47 (3) (1996) 373–383.
[35] N. Sabar, M. Ayob, G. Kendall, R. Qu, A honey-bee mating optimization algorithm for educational timetabling problems, Eur. J. Operat. Res. 216 (3) (2012) 533–543.
[36] E. Burke, A. Eckersley, B. McCollum, S. Petrovic, R. Qu, Hybrid variable neighbourhood approaches to university exam timetabling, Eur. J. Operat. Res. 206 (1) (2010) 46–53.
[37] S. Abdullah, E. Burke, B. McCollum, A hybrid evolutionary approach to the university course timetabling problem, in: IEEE Congress on Evolutionary Computation (CEC), 2007, 2007, pp. 1764–1768.
[38] D. Landa-Silva, J. Obit, Evolutionary non-linear great deluge for university course timetabling, Hybrid Artif. Intell. Syst. (2009) 269–276.
[39] H. Turabieh, S. Abdullah, B. Mccollum, Electromagnetism-like mechanism with force decay rate great deluge for the course timetabling problem, Rough Sets Knowl. Technol. (2009) 497–504.
[40] N. Ejaz, M. Javed, Approach for course scheduling inspired by die-hard co-operative ant behavior, in: Proceedings of the IEEE International Conference on Automation and Logistics, 2007, pp. 18–21.
[41] S. Jat, S. Yang, A guided search genetic algorithm for the university course timetabling problem, in: The 4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2009), Dublin, Ireland, 10–12 August 2009, 2009, pp. 180–191.
[42] G. Jaradat, M. Ayob, An elitist-ant system for solving the post-enrolment course timetabling problem, Database Theory Appl. Biosci. Biotechnol. (2010) 167–176.
[43] M. Ayob, G. Jaradat, Hybrid ant colony systems for course timetabling problems, in: 2nd Conference on Data Mining and Optimization, 2009 (DMO'09), IEEE, 2009, pp. 120–126.
[44] E. Burke, G. Kendall, E. Soubeiga, A tabu-search hyperheuristic for timetabling and rostering, J. Heurist. 9 (6) (2003) 451–470.
[45] H. Asmuni, E. Burke, J. Garibaldi, Fuzzy multiple heuristic ordering for course timetabling, in: Proceedings of the 5th United Kingdom Workshop on Computational Intelligence (UKCI 2005), Citeseer, 2005, pp. 302–309.
[46] P. Rattadilok, A. Gaw, R. Kwan, Distributed choice function hyper-heuristics for timetabling and scheduling, Pract. Theory Automat. Timetabling V (2005) 51–67.
[47] S. Abdullah, E. Burke, B. McCollum, Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem, Metaheuristics (2007) 153–169.
[48] D. Landa-Silva, J. Obit, Great deluge with non-linear decay rate for solving course timetabling problems, in: 4th International IEEE Conference on Intelligent Systems (IS'08), 2008, vol. 1, 2008, pp. 8–11.
[49] A. Abuhamdah, M. Ayob, Multi-neighbourhood particle collision algorithm for solving course timetabling problems, in: 2nd IEEE Conference on Data Mining and Optimization (DMO'09), 2009, 2009, pp. 21–27.
[50] A. Abuhamdah, Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems, Int. J. Comput. Sci. Netw. Secur. 10 (2010) 192–200.

Asaju La'aro Bolaji received his B. Tech. in Physics from Federal University of Technology, Minna, Nigeria and the M.Sc. in Mathematics from the University of Ilorin, Nigeria in 2001 and 2006 respectively. He was awarded PhD degree in Computer Science at Universiti Sains Malaysia in April 2014. He is a member of the Academic staff in the Department of Computer Science, University of Ilorin, Ilorin, Nigeria. His research interests include evolutionary algorithms, nature-inspired computation, and their applications to optimization problems etc.

Ahamad Tajudin Khader obtained his B.Sc. and M.Sc. degrees in Mathematics from the University of Ohio, USA in 1982 and 1983, respectively. He received his PhD in Computer Science from the University of Strathclyde, UK in 1993. He is currently working as a professor and Dean in the School of Computer Sciences, Universiti Sains Malaysia. He has authored and co-authored hundreds of high quality papers in international journals, conferences, and book. He is a member of the Institute of Electrical and Electronics Engineers (IEEE) as an international professor, Association of Computing Machinery (ACM), Special Interest Group for Genetic and Evolutionary Computation (SIGEVO), and Computational Intelligence Society. His research interests mainly focus on optimization and scheduling.

Mohammed Azmi Al-Betar is an assistant professor in the Department of Computer Science at Al-Huson University College, Al-Balqa Applied University, Irbid, Jordan and a senior member of the computational intelligence research group in the school of computer science at University Sains Malaysia (USM). He received his B.Sc. and M.Sc. degrees from Computer Science Department at Yarmouk University, Irbid, Jordan in 2001 and 2003 respectively. He obtained his PhD from the School of Computer Sciences at the Universiti Sains Malaysia (USM), Pulau Penang, Malaysia in October 2010. The author published a number of high quality papers in international journals and conferences. His research interests are mainly directed to metaheuristic optimization methods and hard combinatorial optimization problems including scheduling and timetabling.

Mohammed A. Awadallah received his BSc degree in Computer Science from Islamic University of Gaza, Palestine, and M.Sc. in Computer Information System (CIS) from Arab Academy for Banking and Financial Sciences, Amman, Jordan in 2002 and 2005, respectively. Since 2005, he works as a lecturer in the Department of Computer Science, College of Science, Al-Aqsa University-Gaza, Palestine. He is currently working toward achieving the PhD from the School of Computer Sciences, Universiti Sains Malaysia. His current research interests include evolutionary algorithms, nature-inspired computing, and their applications to timetabling problems etc.