Asaju La'aro Bolaji*, Ahamad Tajudin Khader, Mohammed Azmi Al-Betar and Mohammed A. Awadallah

# A Hybrid Nature-Inspired Artificial Bee Colony Algorithm for Uncapacitated Examination Timetabling Problems

**Abstract:** This article presents a Hybrid Artificial Bee Colony (HABC) for uncapacitated examination timetabling. The ABC algorithm is a recent metaheuristic population-based algorithm that belongs to the Swarm Intelligence technique. Examination timetabling is a hard combinatorial optimization problem of assigning examinations to timeslots based on the given hard and soft constraints. The proposed hybridization comes in two phases: the first phase hybridized a simple local search technique as a local refinement process within the employed bee operator of the original ABC, while the second phase involves the replacement of the scout bee operator with the random consideration concept of harmony search algorithm. The former is to empower the exploitation capability of ABC, whereas the latter is used to control the diversity of the solution search space. The HABC is evaluated using a benchmark dataset defined by Carter, including 12 problem instances. The results show that the HABC is better than exiting ABC techniques and competes well with other techniques from the literature.

**Keywords:** Artificial Bee Colony algorithm, Swarm Intelligence, metaheuristics, timetabling problem, examination timetabling problem.

***Corresponding author: Asaju La'aro Bolaji,** School of Computer Sciences, Universiti Sains Malaysia, Penang 11800, Malaysia, e-mail: abl10_sa0739@student.usm.my
**Asaju La'aro Bolaji:** Department of Computer Science University of Ilorin, Ilorin, Nigeria
**Ahamad Tajudin Khader and Mohammed A. Awadallah:** School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia
**Mohammed Azmi Al-Betar:** School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia; and Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, Al-Huson, Irbid, Jordan

## 1 Introduction

Examination timetabling is a taxing and difficult administrative process for educational institutions. The university examination timetabling problem (UETP) can be defined as the assignment of given examinations to a limited number of time periods, subject to a set of hard and soft constraints [43]. The production of a high-quality timetable that schedules all examinations and takes care of all constraints is one of the major concerns in the timetabling community. A hard constraint is a type of constraint that cannot be violated in the timetable solution, while the violation of soft constraints is tolerated. The main objective is to generate a high-quality timetabling solution that satisfies the hard constraints (feasible timetable) and reduces the violations of soft constraints as much as possible. Examination timetabling problems can be classified into capacitated or uncapacitated problems [43]. The major difference between them is that room capacity is not considered in the uncapacitated examination timetabling problem, yet it is considered in the capacitated examination timetabling problem.

Creating effective techniques for tackling examination timetabling problems has been the subject of research in the domain of operational research and artificial intelligence for the past few decades. Generally, these problems are categorized as NP-hard combinatorial problems [30] and considerable efforts have been

exerted by researchers to develop effective procedures to tackle the examination timetabling problems over the last three decades [43]. The complexity of producing near-optimal solutions for the UETPs has led to the introduction of numerous heuristic and metaheuristic techniques: mathematical programming [12, 25], local-based search methods such as Great Deluge [20, 34], Simulated Annealing [46], Tabu Search [26, 33], Variable Neighborhood Search [2, 17], and population-based methods such as Ant Colony [27, 28], Evolutionary Algorithms [24, 37], Particle Swarm Optimization [29], Harmony Search Algorithm [7], and memetic algorithms [8, 35]. More details on the review of techniques that have been used for examination timetabling problems are provided in Ref. [43].

In recent years, a number of hybrid techniques have been efficiently proposed for tackling wide varieties of optimization problems, including university examination timetabling. These hybridization techniques can be based mainly on the combination of heuristic techniques with metaheuristics, or hybridization of two or more metaheuristic techniques. Generally, the basic idea is to capitalize on the advantages of such hybridization in order to produce a method that is able to strike a right balance between the exploration of the problem search space and exploiting the accumulative search. For example, a complementary advantage is achieved by combining the evolutionary algorithm that explores the solution search space with gradient-descent techniques that exploit the small region of the search space to find the local optimum quickly. These kinds of hybrid techniques have been already proven to be efficient and robust methods for solving a wide range of combinatorial optimization problems, including timetabling and scheduling problems [21, 36].

One of the recent metaheuristic population-based techniques is the Artificial Bee Colony (ABC). It was originally proposed in 2005 by Karaboga for numerical optimization problems. Owing to its simplicity and robustness [14], it has been successfully applied to different optimization problems such as the graph coloring problem, flow job–shop scheduling problem, traveling salesman problem, vehicle routing problem, and quadratic assignment problem. It has been proven in the literature that ABC has a better or comparable performance over other well-known population-based algorithms such as Differential Evolution, Genetic Algorithm (GA), and Particle Swarm Optimization on constrained and unconstrained problems because of the following advantages:

- Apart from the solution number (SN) and maximum cycle number (MCN), ABC uses only one control parameter when compared with many other search techniques. For example, the original GA comes with three control parameters (i.e., crossover rate, mutation rate, and generation gap).
- Ease of implementation with basic mathematical, logical operations, and no derivative information is needed in the initial search.

However, owing to the combinatorial nature of optimization problems, the original ABC has been adopted or hybridized with other gradient-descent methods in order to strike a balance between exploration and exploitation of the problem search space. Admittedly, it has been proven that the structure of ABC tends to observe global exploration rather than local exploitation [49].

Studies from the literature have shown that the ABC algorithm has been adopted to tackle examination timetabling problems [4–6]. In these studies, a two-stage solution approach was employed where the largest weighted degree was used to generate an initial feasible solution at the constructive stage. The ABC algorithm is integrated with large neighborhood structures at the improvement stage. The author compared three different selection strategies of the onlooker operator in Ref. [5] and later hybridized simulated annealing with the onlooker operator in Ref. [6]. Another study [13] tested the performance of ABC with small neighborhood structures for the uncapacitated examination timetabling problem. However, the results obtained by these studies were not comparably better than the existing techniques from the literature. Research has also shown that hybridization between metaheuristic techniques often performs better than individual techniques, as they benefit from the advantages of both (or more) techniques. The motivation to improve the performance of the ABC algorithm for the uncapacitated examination timetabling problem leads to two major objectives of this article: (i) hybridizing a simple local search technique (SLST) with the employed bee operator of the ABC, and (ii) replacing the concept of the scout bee operator with random consideration strategy of the Harmony

Search Algorithm (HSA) process. The proposed technique is called the Hybrid ABC (HABC) algorithm. The hybrid technique is analyzed using a benchmark dataset proposed in Ref. [22]. This dataset includes 12 real-world problem instances. The proposed method is able to produce high-quality results in comparison with 25 comparative methods.

The rest of the article is organized as follows: Section 2 gives descriptions and formulations of the uncapacitated examination timetabling problem, while Section 3 presents the fundamentals of ABC. Section 4 describes the HABC technique for examination timetabling, and experimental results are presented in Section 5. The last section is devoted to the conclusion and some future works.

## 2 Problem Descriptions and Formulations

Uncapacitated examination timetabling is a process of scheduling a set of examinations, each taken by a set of students, to a set of time periods (or timeslots) subject to satisfying hard and soft constraints. The main objective is to obtain a feasible timetable solution that satisfies the hard constraint (H1) with reduction in the violations of the soft constraint (S1). The hard and soft constraints are as follows:

- H1: No student can be scheduled to sit for more than one examination at the same time.
- S1: The examinations taken by the same student should be spread out evenly across a timetable.

A thorough description of the examination timetabling problem is surveyed in Ref. [43]. A timetabling solution is represented by a vector, $\boldsymbol{x} = (x_1, x_2, \ldots, x_M)$, of examinations, where the value of $x_i$ is the timeslot for examination $i$. The proximity cost function defined in Ref. [7] is used to evaluate the solution. It computes the ratio of the penalty assigned to the total number of soft constraint violations and the total number of students. The formulation for the proximity cost function is given in eq. (1), while the notation of the variables used is shown in Table 1. Note that the notation is adopted from Ref. [7] with some modifications:

$$\frac{1}{N} \times \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} c_{i,j} \times a_{i,j}. \tag{1}$$

**Table 1.** Symbols Used in the Description of U-UETP.

| Symbols | Definition |
|---|---|
| $M$ | Total number of examinations |
| $N$ | Total number of students |
| $P$ | Total number of time periods |
| $E$ | Set of examinations |
| $S$ | Set of students |
| $T$ | Set of time periods |
| $\boldsymbol{x}$ | A timetable solution is given by $(x_1, x_2, \ldots, x_M)$ |
| $x_i$ | Timeslot of examination $i$ |
| $a_{i,j}$ | Proximity coefficient matrix element: whether the timetable $\boldsymbol{x}$ is penalized on the basis of the distance between the time period of examination $i$ and the time period of examination $j$ $a_{i,j} = \begin{cases} 2^{5-\lvert x_i - x_j \rvert} & \text{if } 1 \le \lvert x_i - x_j \rvert \le 5 \\ 0 & \text{Otherwise.} \end{cases}$ |
| $u_{i,j}$ | Student–examination matrix element: if student $s_i$ is taking examination $j$ $u_{i,j} = \begin{cases} 1 & \text{if student } i \text{ is sitting for exam } j \\ 0 & \text{Otherwise.} \end{cases}$ |
| $c_{i,j}$ | Conflict matrix element: total number of students sharing examination $i$ and examination $j$ $c_{i,j} = \sum_{k=1}^{N} u_{k,i} \times u_{k,j} \forall i, j \in E$ |

H1: No student can sit for two examinations simultaneously

$$x_i \neq x_j \quad \forall x_i, x_j \in X \wedge c_{i,j} \geq 1.$$

Notably, the value of the proximity cost function $f(\boldsymbol{x})$ is referred to as the fitness cost of a feasible timetable.

The Carter dataset [23] used in this study consists of 13 datasets that reflect the real-world examination timetabling problems. For the purpose of our study, 12 datasets that are circulated in the literature were used. The characteristics of Carter datasets, varying in size and complexity, are shown in Table 2. The conflict matrix in the last column illustrates density, which is the ratio between the number of elements of values $c_{i,j} > 0$ and the total number of elements in the conflict matrix [43].

## 3 Artificial Bee Colony Algorithm

The ABC algorithm is a relatively new family of Swarm Intelligence (SI) algorithms that can be employed to obtain "optimal" solutions to general optimization problems. It can be easily adopted to tackle different optimization problems and has proven to be efficient, effective, and fast when used for various optimization problems [31, 32]. This algorithm starts with a population of solution (i.e., food source) and is inspired by the intelligent foraging behavior of a honeybee swarm. In ABC, the colony consists of three groups of artificial foragers: employed foragers, onlookers, and scouts. The first half of the colony consists of the artificial employed foragers, while the second half includes the onlookers. For every solution (i.e., food source), there is an associated artificial employed forager. In other words, the number of artificial employed foragers is equal to the number of food sources. The artificial employed forager whose food source is abandoned turns to a scout. The onlookers are the bees waiting in the hive to study the dance behavior of the artificial employed bees in order to select the desired food source. The "scouts" are those bees that are randomly searching for new food sources within the hive. Analogously in the optimization context, the number of food sources (i.e., the employed or onlooker foragers) in the ABC algorithm is equivalent to the number of individuals in the population. Furthermore, the position of a food source signifies the possible solution for the optimization problem. The nectar amount of a solution (i.e., food source) represents the quality of the food source by that solution [31].

In the ABC algorithm, the search cycle consists of three processes: (i) assigning the artificial employed foragers to the food sources and evaluating their nectar amounts; (ii) onlookers select the food sources after obtaining information from artificial employed foragers and calculating their nectar amount; and (iii)

**Table 2.** Characteristics of Uncapacitated Examination Dataset.

| Dataset | Time periods | Examinations | Student | Density |
|---|---|---|---|---|
| CAR-S-91-I | 35 | 682 | 16,925 | 0.13 |
| CAR-F-92-I | 32 | 543 | 18,419 | 0.14 |
| EAR-F-83-I | 24 | 190 | 1125 | 0.27 |
| HEC-S-92-I | 18 | 81 | 2823 | 0.42 |
| KFU-S-93 | 20 | 461 | 5349 | 0.06 |
| LSE-F-91 | 18 | 381 | 2726 | 0.06 |
| RYE-S-93 | 23 | 481 | 11,483 | 0.07 |
| STA-F-83-I | 13 | 139 | 611 | 0.14 |
| TRE-S-92 | 23 | 261 | 4360 | 0.18 |
| UTA-S-92-I | 35 | 622 | 21,266 | 0.13 |
| UTE-S-92 | 10 | 184 | 2750 | 0.08 |
| YOR-F-83-I | 21 | 181 | 941 | 0.29 |

determining the scout bees 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 onlookers waiting at the dance area within the hive. At the next stage after sharing information, every employed forager returns to the food source visited at the previous cycle, as the position of the food source exists in its memory and then selects a new 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 foragers at the dance area to select a food source. The probability with which the food source is selected increases with the increase in the nectar quality of a food source. Therefore, the employed forager with higher nectar quality information recruits the onlookers 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 forager to replace a food source that has nectar quality abandoned by the onlookers. The search process of the ABC algorithm is repeated until the MCN is reached. Note that the MCN is the number of iterations (i.e., cycles) that is predetermined at the initial stage of the search process. The ABC procedure could be represented in Algorithm 1.

**Algorithm 1:** Schematic Pseudocode of the ABC Procedure

```
1: Initialize the food sources and calculate the fitness (nectar amount) of food sources
2: Send the employed foragers to the current food sources
3: MCN = 0;
4: repeat
5:   /*Employed Bees' Phase*/
6:   for each employed bee do
7:     Generate a new food source in its neighborhood
8:     Evaluate the fitness of the new food source
9:     Apply greedy selection on the original food source and the new one
10:  end for
11:  Calculate the probability p for each food source
12:  /*Onlookers' Phase*/
13:  Send onlooker bees on the food sources depending on their nectar qualities
14:  Generate a new food source in its neighborhood
15:  Evaluate the fitness of the new food source
16:  Apply greedy selection on the food source with a higher fitness value
17:  Abandon the exploitation process of the food sources, if the limit is exceeded
18:  Send the scout forager to generate a random food source
19:  Memorize the best food source
20:  MCN = MCN+1
21: until (termination criterions are met)
```

## 4 The Proposed Hybrid Algorithm

In this section, the HABC algorithm is proposed for the uncapacitated examination timetabling problem. The proposed method hybridizes the search capabilities of three powerful operators of metaheuristic techniques, i.e., ABC algorithm, SLST, and random consideration from harmony. It is noteworthy that hybridizing the ABC algorithm with an SLST has important advantages in enhancing the local exploitation capability of ABC, while replacing the concept of the scout bee operator with random consideration is to control the diversity and slow convergence of the proposed HABC. The next two subsections provide a brief review of the proposed SLST followed by a detailed description of the proposed HABC algorithm.

## 4.1 An SLST

The local search technique is used to direct the search toward the local optimum. Its process often begins with a solution randomly generated and its fitness cost is calculated. Then, the solution undergoes random changes and its fitness is re-evaluated. The new solution will replace the old one under the condition that it has a better or equal fitness. This process is repeated until a termination condition is achieved. In this article, three different neighborhood structures are used by the SLST to explore the timetabling solution to enhance its quality. The descriptions of these neighborhood structures are given below, while the pseudo-code of SLST is stated in Algorithm 2.

- NL-Move: Moves a selected examination to a feasible period randomly, i.e., replace the time period $x'_i$ of examination $i$ by another feasible timeslot. For example, Figure 1 illustrates the NL-Move neighborhood where an examination ($e_3$) is moved from a timeslot ($t_6$) to another timeslot ($t_4$).
- NL-Swap: Swap two selected examinations at random, i.e., select examination $i$ and examination $j$ randomly, swap their time periods $(x'_i, x'_j)$. Here, Figure 2 shows the example of an NL-Swap neighborhood where two examinations are selected randomly (i.e., $e_5$ from timeslot $t_3$ and $e_1$ from timeslot $t_9$) and then swap their timeslots.
- NL-KempeChain: First, select the timeslot $x'_i$ of examination $i$ and randomly select another $q'$ timeslot. Second, all examinations that have the same timeslot $x'_i$ that are in conflict with one or more examinations timetabled in $q_i$ are entered to chain $\delta$ where $\delta = \{ j \mid x'_j = x'_i \wedge t_{i,q'} = 0 \wedge \forall j \in E \}$. Third, all examinations that have the same timeslot $q'$ that are conflicting with one or more examinations timetabled in $x'_i$ are entered to a chain $\delta'$ where $\delta' = \{ k \mid x'_k = q' \wedge t_{k,x'_i} = 0 \wedge \forall k \in E \}$, and lastly, simply, assign the examinations in $\delta$ to $q'$ and the examinations in $\delta'$ to $x'_i$. Figure 3 shows the example of the NL-KempeChain

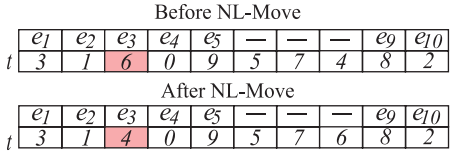


**Figure 1.** NL-Move Example.

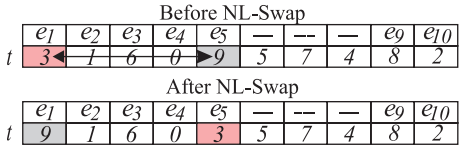


**Figure 2.** NL-Swap Example.

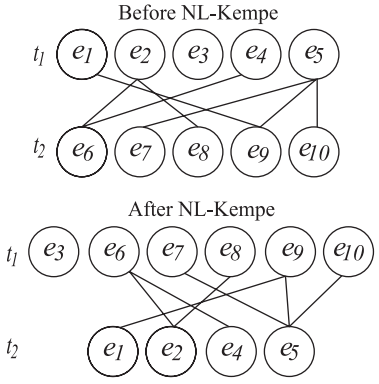


**Figure 3.** NL-Kempe Example.

neighborhood that allows a subset of the examinations in one timeslot to be moved to another timeslot while the feasibility of the solution is preserved. For instance, Figure 3 consists of two timeslots $t_1$ and $t_2$, and each timeslot contains five examinations. Examination $e_5$ cannot be moved from timeslot $t_1$ to timeslot $t_2$ because it clashes with examinations $e_7$, $e_9$, and $e_{10}$ in $t_2$. This means that examinations $e_7$, $e_9$, and $e_{10}$ need to be moved from timeslot $t_2$ and then examination $e_1$ will have to move across to timeslot $t_2$ in order to maintain the feasibility of the timetabling solution.

**Algorithm 2:** SLST Phase

```
1: INPUT (x) {x is the current solution}
2: x' {x' is the improved solution}
3: repeat
4:   i = RND(){RND: generate a random integer number between 1 and 3}
5:   if i = = 1 then
6:     x' = NL-Move (x)
7:   else
8:     if i = = 2 then
9:       x' = NL-Swap (x)
10:    else
11:      if i = = 3 then
12:        x' = NL-KempeChain (x)
13:      end if
14:    end if
15:  end if
16:  if f(x')<f(x) then
17:    x = x'
18:  end if
19: until SLST cycle number (SCN) is reached
20: OUTPUT (x')
```

## 4.2 The Proposed HABC Algorithm

The proposed HABC hybridizes an SLST within the employee forager operator of ABC to improve its search capability and replace the concept of the scout bee operator with the random consideration procedure of HSA. The HABC consists of two main search disciplines: a global one (i.e., ABC algorithm), which is responsible for the global improvement, and a local one (i.e., SLST), which performs local refinement around potential solutions. The framework of the proposed HABC is shown as a flow chart in Figure 4. The implementation of the proposed HABC for uncapacitated examination timetabling is given in the next subsections.

### 4.2.1 Initialization of the ABC and Uncapacitated UETP Parameters

The parameters of uncapacitated UETP (U-UETP) are normally extracted from the problem instances. These parameters include the set of examinations, set of timeslots, set of rooms, etc. (see Table 1). The main decision variable of U-UETP is the examinations. Each examination can be assigned to a feasible timeslot in the timetable solution. A set of all feasible timeslots can be considered as the available range of such examinations. In fact, the feasible timeslot of each examination changes during the search of HABC. The proximity cost function described in eq. (1) is used to evaluate each solution.

At this stage, the parameters of the HABC used for U-UETP are initialized, i.e., the SN, which is similar to the population size in genetic algorithms; MCN, which is similar to the number of iterations; Limit, which

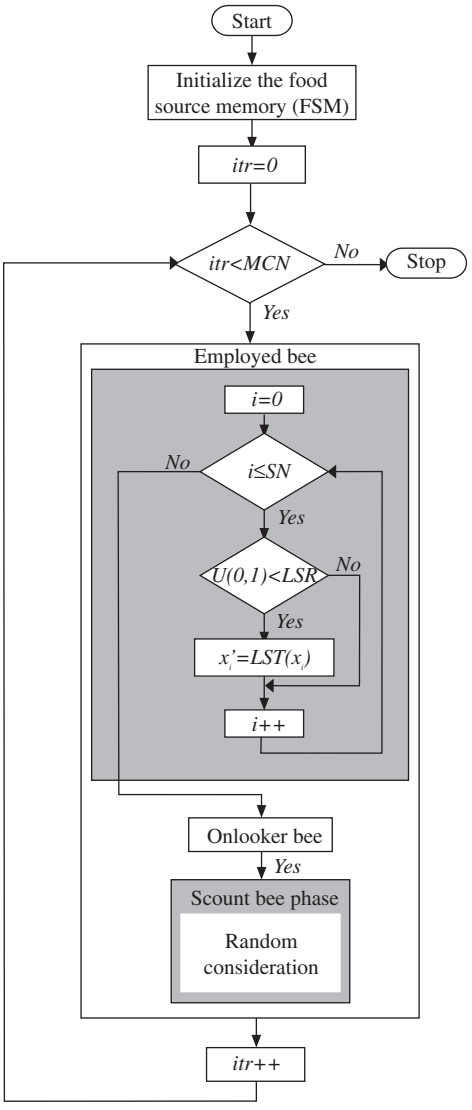


**Figure 4.** Flowchart of the HABC Framework.

functions as a mutation rate in GA; and Local Search Rate (LSR), which decides the rate of using the simple local search in employee bee. These parameters will be explained in more detail in the next steps.

### 4.2.2 Initialize the Food Source Memory

The Food Source Memory (FSM) is an augmented matrix of size SN comprising a vector in each row representing a timetable solution as in eq. (2). Note that the vectors in FSM are generated using a method that combines the saturation degree (SD) [15], and it was previously used by other techniques for U-UETP. The SD begins with an empty timetable, and the examination with the least number of valid timeslots is assigned first without consideration for the soft constraints violations and the process is repeated until all examinations are feasibly assigned to the timeslots. Once the feasibility of the timetabling solution is achieved, the process stops; otherwise, the whole process will be repeated until the hard constraints are satisfied. The SD is used to generate the initial set of feasible solutions because of its efficiency in terms of computational time [1]. In addition, these solutions are sorted in ascending order according to their fitness cost values (i.e., $f(\boldsymbol{x}^1) \leq f(\boldsymbol{x}^2) < \ldots < f(\boldsymbol{x}^{SN})$).

$$\mathbf{FSM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_M^1 \\ x_1^2 & x_2^2 & \cdots & x_M^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{SN} & x_2^{SN} & \cdots & x_M^{SN} \end{bmatrix} \begin{bmatrix} f(\boldsymbol{x}^1) \\ f(\boldsymbol{x}^2) \\ \vdots \\ f(\boldsymbol{x}^{SN}) \end{bmatrix} \tag{2}$$

The HABC initially begins with the population of provisional solutions at the initialized stage and keeps them in FSM. The richness (i.e., fitness) cost of each solution (food source) in FSM is evaluated using the proximity cost function $f(\boldsymbol{x})$ in eq. (1), and the best food is memorized (i.e., $f(\boldsymbol{x}^1)$).

### 4.2.3 Send the Employed Foragers to the Food Sources

This is the core idea of this research. The employed forager operator selects a timetabling solution from the FSM one by one and triggers the SLST to exploit the current solution (i.e., $\boldsymbol{x}^i$ where $i \in \{1, 2, \ldots, SN\}$) with the probability of LSR. The fitness of each new solution (i.e., $\boldsymbol{x}'^i$) is evaluated; if it is better than that of current solution, then the new one replaces the current in FSM. The SLST is triggered by the employed forager of ABC, as shown in Algorithm 3.

**Algorithm 3:** Employed Forager-Triggered LST Procedure

```
1: for i = 1, ..., SN do
2:    if (U(0, 1) < LSR) then
3:       x'^i = SLST(x^i)
4:    end if
5: end for
```

Remarkably, the use of the LSR parameter is to examine the utilization of SLST. In other words, the higher the LSR, the higher the calling of SLST will be, and consequently the higher the exploitation provided. When the SLST is called, the current solution will be improved until the SLST cycle number (i.e., LSTCN) is reached. Thus, the output of the SLST is the refined solution of the current one.

### 4.2.4 Send the Onlooker Bees

The onlooker bee operates on the refined solutions in FSM. Initially, it selects the fittest solutions in FSM using the proportional selection method [31]. The process of selection in the onlooker phase thus works as follows:

- The proportional selection assigns the selection probability for each solution (food source) in FSM using eq. (3):

$$p_j = \frac{f(\boldsymbol{x}^j)}{\sum_{k=1}^{SN} f(\boldsymbol{x}^k)} \tag{3}$$

  Note that the $\sum_{i=1}^{SN} p_i$ is unity.
- Then, the fittest solutions are selected on the basis of their selection probabilities. The solution with a higher selection probability has a higher chance to be selected in the new population. The selected food source is then refined further, as shown in Algorithm 4. It is important here to note that the same neighborhood structures are used in the employed bee phase. However, the onlooker bees implement the neighborhood structure search for the fittest solution in FSM and it should be noted that the refinement process takes just one neighborhood at a time, as shown in Algorithm 4. The fitness of the new solution is calculated and, if it is better, then it replaces the current one.

**Algorithm 4:** Onlooker Bee Phase

```
1: for j = 1, …, SN do
2:   x^i = roulette wheel (FSM) where i∈{1, …, SN}
3:   i = RND(){RND generate a random integer number between 1 and 3}
4:   if (i = = 1) then
5:     x'^i = NL-Move (x^i)
6:   else
7:     if (i = = 2) then
8:       x'^i = NL-Swap(x^i)
9:     else
10:      if (i = = 3) then
11:        x'^i = NL-KempeChain(x^i)
12:      end if
13:    end if
14:  end if
15:  if f(x'^i) ≤ f(x^i) then
16:    x^i = x'^i
17:  end if
18:  next i
19: end for
```

### 4.2.5 Send the Scout Bee 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 a certain number of iterations as decided by the Limit. Here, the scout operator is replaced with random consideration idea of HSA, which is used to direct the search away from the local optimum toward new regions of the solution space. This is done by diversifying the timetabling solution (food source) through the alteration of the variable's information. The variable in the timetabling solution is considered one by one, and the value of the variable is altered at the rate that is determined by the random consideration rate (RCR), as shown in Algorithm 5. It is worthy of mention that the random consideration operator takes the abandoned food source (timetable solution) from FSM and works by selecting the number of examinations that meet certain probability as defined by the RCR at random. The selected examinations are removed from their present timeslots and then reassigned to the new ones (i.e., timeslots). For example, examination $i$ that meets the RCR criteria is removed from the timeslot $x_i$ and then the random consideration reassigned to another timeslot $x_i'$ within the possible range of timeslot as given by the set $\mathfrak{X}_i = \{b \mid t_{i,b} = 1 \wedge b \in 1, \ldots, P\}$. The function of random consideration is similar to the uniform mutation operator in GA, which is a good source of exploration in timetabling.[1]

**Algorithm 5:** Pseudocode of the Random Consideration

```
1: let x^i be a timetabling solution
2: for i = 1 to N do
3:   if (U(0, 1) < RCR) then
4:     diversify (x_i^j)
5:   end if
6: end for
```

where $U(0, 1)$ generates a random number between 0 and 1 and the diversify function $(\boldsymbol{x}_i^j)$ explores the solution search space by removing and reassigning the decision variables from the abandoned solution.

1 Uniform mutation is one of the example conventional mutation operators that simply replaces variables of the solution with a randomly selected real number within a specified range.

#### 4.2.6 Stopping Condition

Steps 3–5 are repeated until the MCN is reached.

# 5 Experimental Results and Analysis

In this section, the performance of the proposed HABC for uncapacitated examination timetabling is evaluated. It 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 proposed method required a maximum of 7 h to obtain the recorded result, although the computational time is not provided in the literature. Burke et al. [20] stated that the time taken is quite reasonable for examination timetabling (because, normally, examination timetables are produced months before they are required; thus, they do not require real-time algorithms to tackle them). Furthermore, it is unacceptable to reduce the computational time at the expense of the quality of solution. The proposed technique is tested using the Carter dataset established in Ref. [22] and published at a website.[2] The characteristics of this dataset are provided in Section 2.

## 5.1 Experimental Design

This section presents the experimental designs showing the performance of the proposed HABC where the influence of using the SLST is studied. It is important to stress here that the SLST is used in the fine tuning of the solution search space toward the global optimum by improving the proposed HABC's local exploitation capability. Table 3 lists the parameter settings of HABC for the U-UETP, which were chosen on the basis of our preliminary experiments. These parameters provide a good balance between the quality of solution and the running

**Table 3.** Settings of Important HABC Parameters.

| Case | Solution number (SN) | Limit | Random consideration rate (RCR) | Simple local search rate (SLSR) | Maximum cycle number (MCN) |
|---|---|---|---|---|---|
| Case 1 | 10 | 100 | 10% | 10% | 10,000 |
| Case 2 | 10 | 100 | 10% | 25% | 10,000 |
| Case 3 | 10 | 100 | 10% | 50% | 10,000 |

**Table 4.** Average Runtime of the HABC on Each Problem Instance of the U-UETP.

| Problem instance | Case 1 | Case 2 | Case 3 |
|---|---|---|---|
| CAR-S-91-I | 13,351 | 18,928 | 24,098 |
| CAR-F-92-I | 5183 | 9431 | 11,029 |
| EAR-F-83-I | 543 | 797 | 1045 |
| HEC-S-92-I | 346 | 531 | 681 |
| KFU-S-93 | 2213 | 6141 | 9172 |
| LSE-F-91 | 3049 | 5618 | 7739 |
| RYE-S-93 | 5491 | 8801 | 10,111 |
| STA-F-83-I | 559 | 719 | 1097 |
| TRE-S-92 | 397 | 7179 | 11,901 |
| UTA-S-92-I | 15,811 | 19,007 | 23,938 |
| UTE-S-92 | 688 | 861 | 1011 |
| YOR-F-83-I | 5583 | 8231 | 10,326 |

2 http://www.cs.nott.ac.uk/ rxq/data.htm.

**Table 5.** Experimental Results of HABC on U-UETP.

| Dataset | Case 1 | Case 2 | Case 3 |
|---|---|---|---|
| CAR-S-91-I | | | |
| Best | **5.00** | 5.05 | 5.06 |
| Mean | 5.05 | 5.16 | 5.20 |
| Worst | 5.09 | 5.24 | 5.28 |
| Stdv | 0.03 | 0.07 | 0.06 |
| CAR-F-92-I | | | |
| Best | **4.22** | 4.25 | 4.22 |
| Mean | 4.29 | 4.30 | 4.25 |
| Worst | 4.33 | 4.33 | 4.27 |
| Stdv | 0.03 | 0.03 | 0.02 |
| EAR-F-83-I | | | |
| Best | 34.52 | 34.55 | **34.07** |
| Mean | 34.86 | 35.22 | 34.70 |
| Worst | 35.11 | 35.66 | 35.06 |
| Stdv | 0.20 | 0.37 | 0.28 |
| HEC-S-92-I | | | |
| Best | 10.68 | 10.50 | **10.36** |
| Mean | 10.78 | 10.74 | 10.64 |
| Worst | 10.87 | 11.22 | 10.83 |
| Stdv | 0.06 | 0.23 | 0.17 |
| KFU-S-93 | | | |
| Best | 14.02 | **14.01** | 14.07 |
| Mean | 14.17 | 14.11 | 14.23 |
| Worst | 14.27 | 14.32 | 14.39 |
| Stdv | 0.08 | 0.09 | 0.09 |
| LSE-F-91 | | | |
| Best | 11.04 | 11.08 | **11.01** |
| Mean | 11.18 | 11.40 | 11.17 |
| Worst | 11.26 | 11.65 | 11.33 |
| Stdv | 0.09 | 0.19 | 0.11 |
| RYE-S-93 | | | |
| Best | **9.28** | 9.30 | 9.31 |
| Mean | 9.49 | 9.40 | 9.41 |
| Worst | 9.64 | 9.57 | 9.70 |
| Stdv | 0.12 | 0.11 | 0.16 |
| STA-F-83-I | | | |
| Best | **157.04** | 157.07 | 157.06 |
| Mean | 157.13 | 157.18 | 157.12 |
| Worst | 157.17 | 157.22 | 157.15 |
| Stdv | 0.04 | 0.05 | 0.03 |
| TRE-S-92 | | | |
| Best | **8.38** | 8.42 | 8.51 |
| Mean | 8.47 | 8.50 | 8.64 |
| Worst | 8.55 | 8.55 | 8.72 |
| Stdv | 0.06 | 0.04 | 0.07 |
| UTA-S-92-I | | | |
| Best | **3.40** | 3.44 | 3.45 |
| Mean | 3.45 | 3.48 | 3.48 |
| Worst | 3.49 | 3.51 | 3.51 |
| Stdv | 0.03 | 0.02 | 0.02 |
| UTE-S-92 | | | |
| Best | **25.80** | 25.87 | 25.91 |
| Mean | 26.17 | 26.12 | 26.17 |
| Worst | 26.35 | 26.3 | 26.35 |
| Stdv | 0.19 | 0.15 | 0.15 |
| YOR-F-83-I | | | |
| Best | 37.53 | 37.27 | **36.95** |
| Mean | 37.69 | 37.75 | 37.51 |
| Worst | 37.84 | 38.18 | 37.97 |
| Stdv | 0.09 | 0.25 | 0.36 |

time needed to achieve good solutions quality. The preliminary experiments show that increase in SN has no impact on the performance of the proposed technique, but the runtime is increased. It was found that the most sensitive parameter is the simple local search rate (SLSR), which determines the usage of an SLST. Note that when the value of SLSR increased, the exploitation rate of the HABC is increased and thus the runtime required by the proposed technique equally increases. The investigation of the effect of varying the SLSR parameter is conducted. The average runtime taken for each case on each problem instance is recorded in Table 4.

## 5.2 Experimental Results

Table 5 shows the experimental results of the proposed HABC with varying LSR values by showing the best, mean, worst, and standard deviation over 10 runs. The best solution for each Carter dataset is highlighted in bold, while Figure 5 shows the boxplots that illustrate the distribution of solution quality for all the datasets. The results show that the LSR with a lower value generally improves the solutions obtained. As shown in

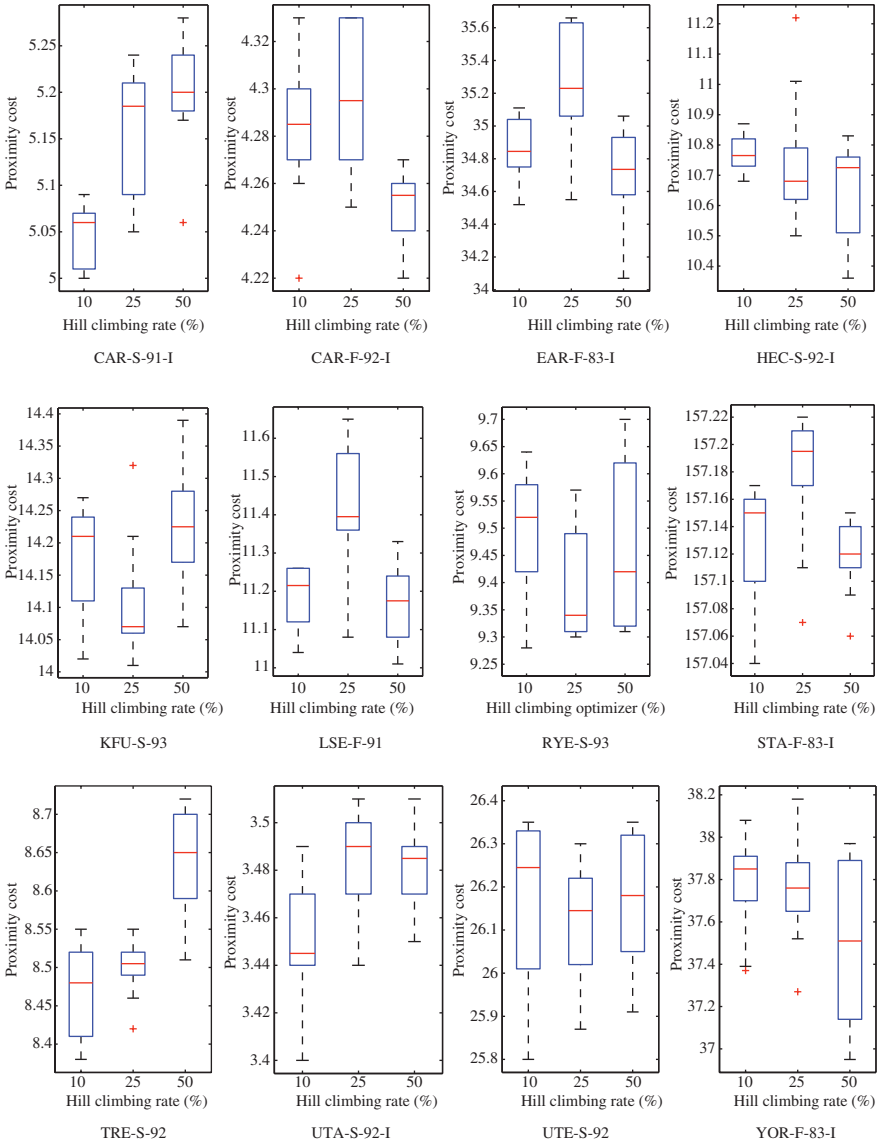


**Figure 5.** Boxplot Demonstrating the Effect of Varying HCR on HABC.

**Table 6.** Key to Hyperheuristic and Other Heuristic Comparative Techniques.

| No. | Key | Technique | References |
|---|---|---|---|
| 1 | HABC | Hybrid Artificial Bee Colony Algorithm | Proposed technique |
| 2 | MLNS-LS | A multistart large NS approach with LS methods | [1] |
| 3 | HPSO | Hybrid Particle Swarm Optimization | [3] |
| 4 | DABC | Disruptive Artificial Bee Colony | [6] |
| 5 | FLE | Fuzzy logic expert | [10] |
| 6 | NFA | Novel fuzzy approach | [11] |
| 7 | FHOM | Fuzzy heuristic ordering model | [9] |
| 8 | GBHH-ETP | Graph-based hyperheuristic for ETPs | [18] |
| 9 | ETS-LSM | Enhancing timetable solutions with local search methods | [16] |
| 10 | ASH-GRASP | Adaptive selection of heuristics within GRASP | [19] |
| 11 | LCA | Largest cliques as the initialization for graph heuristics with backtracking | [23] |
| 12 | TSA | Tabu Search algorithm | [26] |
| 13 | HMOEA | Hybrid Multiobjective Evolutionary | [24] |
| 14 | ANTCOL | Ant Colony | [28] |
| 15 | MMAC | Max–Min Ant Colony | [28] |
| 16 | HA | Hybrid algorithm | [35] |
| 17 | NGD | Evolving hyperheuristics | [38] |
| 18 | GPHH | Genetic programming hyperheuristic | [39] |
| 19 | SHCA | Study of heuristic combination approach | [40] |
| 20 | VNHH | Variable neighborhood hyperheuristic | [41] |
| 21 | GCHHF | Graph coloring hyperheuristic framework | [42] |
| 22 | AAT | Adaptive automated technique | [44] |
| 23 | GCCHH | Graph coloring constructive hyperheuristics | [45] |
| 24 | AIHA | An integrated hybrid approach | [47] |
| 25 | TSLTM | Tabu Search with longer-term memory | [48] |

Figure 5, it can be seen that the gaps between the best, average, and worst solution qualities are very close, which demonstrates that the proposed HABC is a robust technique.

The experimental results obtained by HABC are compared with those exiting techniques that are available to the authors. These are 24 hybrid SI-related techniques, heuristics, and other hyperheuristics techniques that worked on Carter problem instances. The abbreviations of the comparative techniques are given in Table 6.

Table 7 shows that the proposed HABC obtained very competitive results when compared with 24 other techniques (i.e., SI-, heuristic-, and hyperheuristic-based techniques). The HABC outperformed these techniques in three problem instances (i.e., EAR-F-83, HEC-S-92, and STA-F-83), came second in four instances (i.e., RYE-S-93, UTA-S-92-I, UTA-S-92, and YOR-F-83-I), and achieved third best on KFU-S-93 and TRE-S-92. Finally, it came fourth and fifth on the rest of the remaining problem instances. As shown in Table 7, the best proximity values (lowest is best) are highlighted in bold, while "-" indicates that the technique could not find a feasible timetable. Similarly, it can be seen that none of these techniques comprehensively outperformed the others.

It can be concluded that the HABC is generally able to produce high-quality results when compared against hyperheuristic-, other heuristic-, and SI-based techniques. This indicates that using ABC as a global improvement method hybridized with SLST as a local improvement method is a powerful technique for the uncapacitated examination timetabling problem where it is able to strike a right trade-off between global wide rage exploration of the timetabling problem search space and local nearby exploitation of the promising regions on the timetabling problem search space.

# 6 Conclusion

This article presents an HABC that hybridizes an SLST within the employee bee operator of the ABC algorithm for tackling the U-UETP. In HABC, the SLST is hybridized within the employed bee operator of the original

**Table 7.** Comparison of ABC Algorithms with Hyperheuristics and Other Heuristic Techniques.

| Method | CAR-F-92 I | CAR-S-91 I | EAR-F-83 I | HEC-S-92 I | KFU-S-93 | LSE-F-91 | RYE-S-93 | STA-F-83 I | TRE-S-92 | UTA-S-92 I | UTE-S-92 | YOR-F-83 I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HABC | 4.22 | 5.00 | **34.08** | **10.32** | 13.91 | 11.04 | 9.18 | **157.04** | 8.38 | 3.40 | 25.80 | 36.53 |
| MLNS-LS | **4.1** | 4.8 | 36 | 10.8 | 15.2 | 11.9 | – | 159 | 8.5 | 3.6 | 26 | 36.2 |
| HPSO | 4.67 | 5.22 | 35.74 | 10.74 | 14.47 | 10.76 | 9.95 | 157.1 | 8.47 | 3.52 | 25.86 | 38.72 |
| DABC | 4.84 | 5.42 | 37.54 | 11.21 | 15.13 | 12.06 | – | 157.52 | 9.23 | 3.94 | 27.57 | 40.94 |
| FLE | 4.56 | 5.29 | 37.02 | 11.78 | 15.81 | 12.09 | 10.35 | 160.42 | 8.67 | 3.57 | 27.78 | 40.66 |
| NFA | 4.51 | 5.19 | 36.64 | 11.6 | 15.34 | 11.35 | 10.05 | 160.79 | 8.47 | 3.52 | 27.55 | 39.79 |
| FHOM | 4.54 | 5.29 | 37.02 | 11.78 | 15.8 | 12.09 | 10.38 | 160.42 | 8.67 | 3.57 | 28.07 | 39.8 |
| GBHH-ETP | 4.53 | 5.36 | 37.92 | 12.25 | 15.20 | 11.33 | – | 158.19 | 8.92 | 3.88 | 28.01 | 41.37 |
| ETS-LSM | **4.1** | **4.65** | 37.05 | 11.54 | 13.9 | 10.82 | – | 168.73 | 8.35 | **3.2** | 25.83 | 37.38 |
| ASH-GRASP | 4.45 | 5.37 | 37.89 | 11.78 | 15.45 | 12.12 | – | 158.94 | 8.99 | 3.5 | 26.62 | 42.19 |
| LCA | 6.2 | 7.1 | 36.4 | 10.8 | 14 | 10.5 | **7.3** | 161.5 | 9.6 | 3.5 | 25.80 | 41.7 |
| TCA | 5.2 | 6.2 | 45.7 | 12.4 | 18 | 15.5 | – | 160.8 | 10 | 42 | 27.8 | 41 |
| HMOEA | 4.3 | 5.2 | 36.8 | 11.1 | 14.5 | 11.3 | 9.8 | 157.3 | 8.6 | 3.5 | 26.4 | 39.4 |
| ANTCOL | 4.2 | 5.4 | 34.2 | 10.4 | 14.3 | 11.3 | 8.8 | 158.03 | 8.6 | 3.5 | **25.30** | 36.4 |
| MMAC | 4.8 | 5.7 | 36.8 | 11.3 | 15 | 12.1 | 10.2 | 157.2 | 8.8 | 3.8 | 27.7 | 39.6 |
| HA | 4.3 | 5.1 | 35.4 | 10.6 | 13.5 | 10.5 | – | 157.3 | 8.4 | 3.5 | 25.1 | 37.4 |
| NGD | 4.18 | 4.93 | 36.64 | 11.26 | 14.21 | 10.81 | 9.25 | 157.39 | 8.48 | 3.32 | 27.16 | 39.84 |
| GPHH | – | – | 35.56 | 11.43 | – | – | – | 158.58 | – | – | 27.31 | 39.96 |
| SHCA | 4.28 | 4.97 | 35.86 | 11.85 | 14.62 | 11.14 | 9.65 | 158.33 | 8.48 | 3.4 | 28.88 | 40.74 |
| VNHH | 4.7 | 5.4 | 37.29 | 12.23 | 15.11 | 12.71 | – | 158.8 | 8.67 | 3.54 | 29.68 | 43 |
| GCHHF | 4.16 | 5.16 | 36.52 | 11.94 | 14.79 | 11.15 | – | 159 | 8.6 | 3.59 | 28.3 | 41.81 |
| AAT | 4.32 | 5.11 | 36.86 | 11.62 | 15.18 | 11.32 | – | 158.88 | 8.52 | 3.21 | 28 | 40.71 |
| GCCHH | 4.7 | 5.14 | 37.86 | 11.9 | 15.3 | 12.33 | 10.71 | 160.12 | 8.32 | 3.88 | 32.67 | 40.53 |
| AIHA | **4.1** | 4.8 | 34.92 | 10.73 | **13.00** | **10.01** | 9.65 | 158.26 | **7.88** | 3.2 | 26.11 | **36.11** |
| TSLTM | 4.63 | 5.73 | 45.8 | 12.9 | 17.1 | 14.7 | 11.6 | 158 | 8.94 | 4.44 | 29 | 42.3 |

ABC in order to improve the local exploitation capability of ABC in tackling the problems. The method is evaluated using a dataset produced by Carter. Three experimental cases have been designed to show the effect of using SLST within the employee bee operator. Their results show that using SLST with a lower rate is better than that with the higher rate.

Comparative evaluation with 25 comparative methods has been conducted. It shows that the proposed HABC is a powerful technique that is able to generate new results for some problem instances. Finally, the experimental results show that the HABC is competitive and works well across all tested Toronto instances in comparison with other approaches that have been studied in the literature.

As HABC-based U-UETP has been proved to be very robust and efficient, we believe future work can further improve the proposed HABC by

- Hybridizing crossover operator to diversify the solution search space;
- Further investigating the performance of HABC on other formulations of the university timetabling;
- Investigating other efficient local search-based techniques such as Great Deluge and Tabu Search;
- Combining different selection schemes in the onlooker bee phase, such as linear rank, exponential rank, tournament selection, and many others.

**Acknowledgments:** The authors wish to thank the anonymous referees for their helpful and insightful comments, which have greatly improved the clarity of the paper. The first author would like to appreciate 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.

Received January 5, 2014; previously published online March 28, 2014.

# Bibliography

[1] S. Abdullah and E. K. Burke, A multi-start large neighbourhood search approach with local search methods for examination timetabling, in: *International Conference on Automated Planning and Scheduling (ICAPS 2006),* pp. 334–337, Cumbria, UK, 2006.

[2] S. Ahmadi, R. Barone, P. Cheng, P. Cowling and B. McCollum, Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem, in: *Proceedings of Multidisciplinary International Scheduling: Theory and Applications (MISTA 2003),* pp. 13–16, Nottingham, August, 2003.

[3] M. Alinia Ahandani, M. T. Vakil Baghmisheh, M. A. Badamchi Zadeh and S. Ghaemi, Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem, *Swarm Evol. Comput.* **2** (2012), 21–34.

[4] M. Alzaqebah and S. Abdullah, Artificial bee colony search algorithm for examination timetabling problems, *Int. J. Phys. Sci.* **6** (2011), 1452–1462.

[5] M. Alzaqebah and S. Abdullah, Comparison on the selection strategies in the artificial bee colony algorithm for examination timetabling problems, *Int. J. Soft Comput. Eng.* **1** (2011), 158–163.

[6] M. Alzaqebah and S. Abdullah, Hybrid artificial bee colony search algorithm based on disruptive selection for examination timetabling problems, *Combin. Optim. Appl.* **6831** (2011), 31–45.

[7] M. A. Al-Betar, A. T. Khader and F. Nadi, Selection mechanisms in memory consideration for examination timetabling with harmony search, in: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pp. 1203–1210, ACM, 2010.

[8] M. A. Al-Betar, A. T. Khader and I. A. Doush, Memetic techniques for examination timetabling, *Ann. Oper. Res.* (2013), 1–28, doi 10.1007/s10479-013-1500-7.

[9] H. Asmuni, E. K. Burke, J. M. Garibaldi, B. McCollum and A. J. Parkes, An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables, *Comput. Oper. Res.* **36** (2009), 981–1001.

[10] H. Asmuni, E. Burke, J. Garibaldi and B. McCollum, Fuzzy multiple heuristic orderings for examination timetabling, *Pract. Theory Autom. Timetabling* **V** (2005), 334–353.

[11] H. Asmuni, E. Burke, J. Garibaldi and B. McCollum, A novel fuzzy approach to evaluate the quality of examination timetabling, *Pract. Theory Autom. Timetabling* **VI** (2007), 327–346.

[12] P. Boizumault, Y. Delon and L. Péridy, Constraint logic programming for examination timetabling, *J. Logic Program.* **26** (1996), 217–233.

[13] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, M. A Awadallah and J. J. Thomas, The effect of neighborhood structures on examination timetabling with artificial bee colony, in: *9th International Conference on the Practice and Theories of Automated Timetabling (PATAT 2012)*, pp. 131–144, Son, Norway, SINTEF, 2012.
[14] A. L. Bolaji, A. T. Khader, M. A. Al-Betar and M. A. Awadallah, Artificial bee colony algorithm, its variants and applications: a survey, *J. Theor. Appl. Inf. Technol.* **47** (2013), 434–459.
[15] D. Brélaz, New methods to color the vertices of a graph, *Commun. ACM* **22** (1979), 251–256.
[16] E. K. Burke and J. P. Newall, Enhancing timetable solutions with local search methods, in: *Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science*, vol. 2740, E. Burke and P. De Causmaecker, eds., pp. 195–206, Springer-Verlag, Berlin, 2003.
[17] E. K. Burke, A. J. Eckersley, B. McCollum, S. Petrovic and R. Qu, Hybrid variable neighbourhood approaches to university exam timetabling, *Eur. J. Oper. Res.* **206** (2010), 46–53.
[18] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic and R. Qu, A graph-based hyper-heuristic for educational timetabling problems, *Eur. J. Oper. Res.* **176** (2007), 177–192.
[19] E. K. Burke, R. Qu and A. Soghier, Adaptive selection of heuristics within a grasp for examination timetabling problems, in: *Proceedings of Multidisciplinary International Conference on Scheduling*, pp. 409–422, 2009.
[20] E. Burke, Y. Bykov, J. Newall and S. Petrovic, A time-predefined local search approach to exam timetabling problems, *IIE Trans.* **36** (2004), 509–528.
[21] E. Burke and J. Landa Silva, The design of memetic algorithms for scheduling and timetabling problems, *Recent Adv. Memetic Algorithms* **166** (2005), 289–311.
[22] M. W. Carter, G. Laporte and S. Y. Lee, Examination timetabling: algorithmic strategies and applications, *J. Oper. Res. Soc.* **47** (1996), 373–383.
[23] M. Carter and G. Laporte, Recent developments in practical examination timetabling, in: *Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science*, vol. 1153, E. Burke and P. De Causmaecker, eds., pp. 3–21, Springer-Verlag, Berlin, 1996.
[24] P. Côté, T. Wong and R. Sabourin, A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem, *Pract. Theory Autom. Timetabling* **V** (2005), 294–312.
[25] S. Daskalaki, T. Birbas and E. Housos, An integer programming formulation for a case study in university timetabling, *Eur. J. Oper. Res.* **153** (2004), 117–135.
[26] L. Di Gaspero and A. Schaerf, Tabu search techniques for examination timetabling, in: *Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science,* vol. 2079, E. Burke and P. De Causmaecker, eds., pp. 104–117, Springer-Verlag, Berlin, 2001.
[27] K. A. Dowsland and J. M. Thompson, Ant colony optimization for the examination scheduling problem, *J. Oper. Res. Soc.* **56** (2004), 426–438.
[28] M. Eley, Ant algorithms for the exam timetabling problem, in: *Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling VI*, pp. 364–382, Springer-Verlag, Berlin, 2006.
[29] D. R. Fealko and S. Adviser-Mukherjee, *Evaluating particle swarm intelligence techniques for solving university examination timetabling problems*, a dissertation for the degree of Doctor of Philosophy, Graduate School of Computer and Information Sciences, Nova Southeastern University, 2006.
[30] M. R. Garey and D. S. Johnson, *Computers and intractability. A guide to the theory of NP-completeness*, A Series of Books in the Mathematical Sciences, WH Freeman and Company, San Francisco, CA, 1979.
[31] D. Karaboga, An idea based on honey bee swarm for numerical optimization, *Techn. Rep. TR06*, Erciyes Univ. Press, Erciyes, 2005.
[32] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *J. Global Optim.* **39** (2007), 459–471.
[33] G. Kendall and N. Hussin, An investigation of a Tabu-search-based hyper-heuristic for examination timetabling, in: *Multidisciplinary Scheduling: Theory and Applications*, G. Kendall, E. K. Burke, S. Petrovic and M. Gendreau (Eds.), pp. 309–328, Springer US, 2005.
[34] B. McCollum, P. J. McMullan, A. J. Parkes, E. K. Burke and S. Abdullah, An extended great deluge approach to the examination timetabling problem, in: *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009)*, pp. 424–434, 2009.
[35] L. Merlot, N. Boland, B. Hughes and P. Stuckey, A hybrid algorithm for the examination timetabling problem, in: *The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science*, vol. 2740, E. Burke and P. De Causmaecker, eds., pp. 207–231, Springer-Verlag, Berlin, 2003.
[36] E. Özcan, A. J. Parkes and A. Alkan, The interleaved constructive memetic algorithm and its application to timetabling, *Comput. Oper. Res.* **39** (2012), 2310–2322.
[37] L. F. Paquete and C. M. Fonseca, A study of examination timetabling with multiobjective evolutionary algorithms, in: *4th Metaheuristics International Conference (MIC 2001)*, pp. 149–154, 2001.
[38] N. Pillay, Evolving hyper-heuristics for the uncapacitated examination timetabling problem, *J. Oper. Res. Soc.* **63** (2011), 47–58.

[39] N. Pillay and W. Banzhaf, A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem, in: *Progress in Artificial Intelligence*, J. Neves, M. F. Santos and J. M. Machado (Eds.), Lecture Notes in Computer Science 4874, pp. 223–234, Springer, Berlin Heidelberg, 2007.
[40] N. Pillay and W. Banzhaf, A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem, *Eur. J. Oper. Res.* **197** (2009), 482–491.
[41] R. Qu and E. Burke, Hybrid variable neighborhood hyperheuristics for exam timetabling problems, 2005.
[42] R. Qu and E.K. Burke, Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems, *J. Oper. Res. Soc.* **60** (2008), 1273–1285.
[43] R. Qu, 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, *J. Sched.* **12** (2009), 55–89.
[44] R. Qu, E. K. Burke and B. McCollum, Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems, *Eur. J. Oper. Res.* **198** (2009), 392–404.
[45] N. R. Sabar, M. Ayob, R. Qu and G. Kendall, A graph coloring constructive hyper-heuristic for examination timetabling problems, *Appl. Intell.* **37** (2012), 1–11.
[46] J. M. Thompson and K. A. Dowsland, A robust simulated annealing based examination timetabling system, *Comput. Oper. Res.* **25** (1998), 637–648.
[47] H. Turabieh and S. Abdullah, An integrated hybrid approach to the examination timetabling problem, *Omega* **39** (2011), 589–607.
[48] G. White and B. Xie, Examination timetables and Tabu search with longer-term memory, in: *The Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science*, vol. 2079, E. Burke and W. Erbens, eds., pp. 85–103, Springer-Verlag, Berlin, 2001.
[49] G. Zhu and Sam Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, *Appl. Math. Comput.* **217** (2010), 3166–3173.