A Hybrid Swarm-Based Approach to University Timetabling

Cheng Weng Fong, Hishammuddin Asmuni, and Barry McCollum

***Abstract*—This paper is concerned with the application of an automated hybrid approach in addressing the university timetabling problem. The approach described is based on the nature-inspired artificial bee colony (ABC) algorithm. An ABC algorithm is a biologically-inspired optimization approach, which has been widely implemented in solving a range of optimization problems in recent years such as job shop scheduling and machine timetabling problems. Although the approach has proven to be robust across a range of problems, it is acknowledged within the literature that there currently exist a number of inefficiencies regarding the exploration and exploitation abilities. These inefficiencies can often lead to a slow convergence speed within the search process. Hence, this paper introduces a variant of the algorithm which utilizes a global best model inspired from particle swarm optimization to enhance the global exploration ability while hybridizing with the great deluge (GD) algorithm in order to improve the local exploitation ability. Using this approach, an effective balance between exploration and exploitation is attained. In addition, a traditional local search approach is incorporated within the GD algorithm with the aim of further enhancing the performance of the overall hybrid method. To evaluate the performance of the proposed approach, two diverse university timetabling datasets are investigated, i.e., Carter's examination timetabling and Socha course timetabling datasets. It should be noted that both problems have differing complexity and different solution landscapes. Experimental results demonstrate that the proposed method is capable of producing high quality solutions across both these benchmark problems, showing a good degree of generality in the approach. Moreover, the proposed method produces best results on some instances as compared with other approaches presented in the literature.**

***Index Terms*—Artificial bee colony (ABC), evolutionary algorithm, great deluge (GD) algorithm, metaheuristics, university timetabling.**

I. Introduction

THIS paper focuses on two areas within the complex area of university timetabling, i.e., course and examination timetabling. On a global scale, timetabling within universities is an important administrative task involving a combination of planning and management. In general, university timetabling can be defined as assigning a set of events or exams within a set of permitted timeslots and rooms while satisfying a number of predefined constraints [1]. A clash-free timetable, e.g., no student has to either sit two examinations or attend two events associated with a course at the same time, is known as a feasible timetable. Generally, two distinct types of constraints, known as hard and soft, must be addressed when generating a timetable solution. The satisfaction of hard constraints is mandatory in order to produce a feasible timetable while the satisfaction of soft constraints should be maximized as these directly relate to the overall quality of the resultant solution. An example of a soft constraint is to allow students to have as much time as possible between successive events. An overview of university timetabling problems can be seen in [1]–[4].

By referring to the comprehensive survey papers within the area of educational timetabling [1]–[3], [5] it can be seen that various approaches have been proposed and applied in addressing university timetabling problems. Initial research in this area involved implementing sequential graph coloring [6], [7] heuristics in addressing the problem, but in comparison to more recent research, it can be observed that the performance of those techniques were relatively poor. More sophisticated search-based approaches were introduced into this domain over these last two decades, i.e., metaheuristic-based approaches. These approaches can be divided into two types, i.e., single solution and population-based approach. Examples of single solution-based metaheuristic are simulated annealing [8]–[11], tabu search [12]–[18], variable neighborhood search [19], large neighborhood search [20], local search [21], and the great deluge (GD) algorithm [22]–[32]. Population-based metaheuristics include genetic algorithm [33], [34], ant colony algorithm [35]–[37], honey bee mating algorithm [38], harmony search [39], [40], and artificial bee colony (ABC) algorithm [41], [42].

In recent years, hybridization of metaheuristic approaches has proven to be effective in solving university timetabling problems. According to the comprehensive survey on timetabling problems, Qu *et al.* [3] concluded that: "there are many research directions generated by considering the hybridization of metaheuristic methods particularly between population-based methods and other approaches." In addition, the efficiency of hybridizing population-based methods with single solution-based methods has been highlighted in

Manuscript received May 7, 2013; revised October 9, 2013, February 19, 2014, May 4, 2014, and February 5, 2015; accepted March 4, 2015. Date of publication March 10, 2015; date of current version November 25, 2015.

C. W. Fong is with the Faculty of Applied Sciences and Computing, Department of Computer Sciences and Mathematics, Tunku Abdul Rahman University College, Segamat 85000, Malaysia.

H. Asmuni is with the Software Engineering Research Group, Software Engineering Department, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia (e-mail: hishamudin@utm.my).

B. McCollum is with the Department of Computer Science, Queen's University Belfast, Belfast BT7 1NN, U.K.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2015.2411741

many recent research trends. Blum and Roli [43] summarized that "population-based methods are better in identifying promising areas in the search space, whereas trajectory methods are better in exploring promising areas in the search space. Thus, metaheuristic hybrids in some way manage to combine the advantages of population-based methods with the strength of trajectory methods." As an example, Eley [35] proposed two different versions of ant algorithms (max-mix ant system for examination timetabling (MMAS-ET) and ant colony algorithm for examination timetabling (ANTCOL-ET)) in solving examination timetabling problems. Both approaches involved the hybridization of an ant algorithm and a hill-climbing approach. Experimental results demonstrated that ANTCOL-ET outperformed MMAS-ET. This approach is better at exploring the promising search regions and is able to escape from local optima, but is poor in fine-tuning the final solution search region. Turabieh and Abdullah [26] proposed a hybrid fish swarm algorithm to address examination timetabling. In this paper, the fish swarm algorithm was hybridized with two local search approaches, i.e., steepest descent algorithm and GD. In addition, the authors modified the algorithm to exploit selected solutions (by using a roulette wheel selection scheme) rather than using all solutions in the population. The algorithm is effective in exploration, but weak in exploitation. Pillay and Banzhaf [34] proposed a two-stage informed genetic algorithm for the examination timetabling problem. In the first phase of the approach, feasible solutions that satisfy hard constraints are produced, whilst the second phase tries to maximize the satisfaction of the soft constraints. The experimental results show that the approach is good in exploring the problem search region, but poor in exploitation. In addition, the authors did not highlight the motive of using a genetic algorithm in both initialization and improvement phases, which has the tendency to significantly increase the computational time.

Turabieh and Abdullah [44] proposed a tabu-based memetic algorithm for the course timetabling problem. In this paper, a tabu search was applied to enhance the quality of a solution after the crossover and mutation operations of the genetic algorithm. It was demonstrated that this approach was good in exploration, but weak in exploitation. Turabieh and Abdullah [27] presented a hybridization of an electromagnetic-like approach with a GD algorithm for the examination timetabling problem. The method, tested on the Carter and the first track of ITC2007 [45] benchmark datasets, generated good quality results in both cases. However, the authors did not discuss what happens when the GD is unable to improve the quality of the solutions (local optima). Abdullah *et al.* [46] proposed a hybrid evolutionary algorithm for the course timetabling problem. The proposed approach constituted hybridization between a memetic algorithm and a randomized iterative improvement local search. However, the authors eliminated the crossover process in the memetic algorithm and therefore decreased the exploration ability of the search process. Burke *et al.* [47] proposed a hybridization of variable neighborhood search with a genetic algorithm for the examination timetabling problem. In the proposed approach, the chromosome is used to represent a set of neighborhood structures rather than a timetable solution. The neighborhood structures in the chromosome are subsequently used to produce potential solutions. Experimental results show that the approach is capable of producing best known results on certain instances. Other hybridization approaches that have been proposed in recent years in tackling university timetabling problems can be found in [15], [28]–[30], [33], [40], [44], and [47]–[55].

Motivated by the above, this paper proposes a hybrid ABC algorithm for the university timetabling problem. The ABC, developed by Karaboga [56] mimics the foraging behavior of the honey bee. It should be noted that this differs from the related honey bee algorithm which mimics the mating behavior of bees. The ABC approach is classified as an evolutionary algorithm and has been widely and successfully applied in solving a range of optimization problems, including timetabling problems [41], [42]. There are three types of bees in ABC which are utilized within the search process, i.e., employed, onlooker, and scout bees. The power of ABC is the ability to exploit (by employing neighborhood moves in employed and onlooker bees) and explore (by "scout bees" randomly generating new solutions to replace inactive solutions, i.e., solutions that have become stuck in local optima) within the problem search region. Nevertheless, as is common with many metaheuristic approaches, there are still some identified weaknesses regarding the exploration and exploitation processes [57]–[59]. Both exploration and exploitation are important abilities for any metaheuristic approach in searching for optimal solutions in the problem search region. In practice, these two abilities are very much in contradiction to each other, therefore a balance is required in being able to effectively search the solution space. To effectively solve the optimization problem using metaheuristic approaches, it has been acknowledged that both exploration and exploitation abilities must therefore be well balanced [40]. In the ABC algorithm, the scout bee randomly generates a new solution to replace an abandoned solution which has induced slow convergence speed (poor exploitation). In addition, the slow neighborhood search and greedy selection scheme that is normally employed by both employed and onlooker bees in accepting the newly generated candidate solutions is unable to fine-tune (poor exploitation) within the search region of each solution in the population. Greedy selection schemes accept only those candidate solutions with better or equivalent quality than that of the current solution, which often causes the search process to become trapped in local optima. In many cases this issue results in poor quality resultant solutions [25]. Hence, the method proposed here is introduced with the aim of attaining a balance between the exploration and exploitation abilities of the basic ABC algorithm.

In this paper, a "global best" model inspired from particle swarm optimization (PSO) [60] is implemented within the basic ABC algorithm. In PSO, the global best model is used to encourage the search toward promising search regions by sharing the information on the global best solution found so far. Therefore, the global best model is implemented in ABC to guide the search process, with the aim of directing the search toward promising regions within the solution space. In terms of exploitation, an integration of Nelder–Mead simplex search (NMSS) [61] within the GD algorithm [62] (NMGD) is hybridized with the basic ABC algorithm (named NMGD-ABC) with the aim of fine-tuning the problem search region. This hybridized approach attempts to improve the exploration and exploitation abilities of the ABC algorithm to enhance the convergence speed and search capabilities of the basic ABC.

In this paper, the basic ABC and proposed approach are tested on two benchmark datasets, i.e., the Carter un-capacitated examination timetabling dataset and the Socha course timetabling dataset. The purpose of using these two datasets is to demonstrate the degree of generality of the proposed approach in solving two timetabling problems with differing complexity. Experimental results illustrate that the proposed approach is capable of producing good quality solutions in comparison with the basic ABC algorithm and other metaheuristic approaches described in the literature.

The remainder of this paper is organized as follows. First, a description of the university timetabling problem is presented in Section II. The various algorithmic components used within the hybridization and general proposed algorithm are presented in Sections III and IV, respectively. Experimental results are discussed in Section V. And finally, the conclusion is given in Section VI.

II. Problem Description

University timetabling can be categorized into two main problem areas: 1) course and 2) examination timetabling. These two types of timetabling problems are similar to some extent, but there exist some important differences. Both areas contain entities which require timetabling within a set of timeslots based on a number of hard and soft constraints. An important difference between examination timetabling and course timetabling concerns room allocation. Several exams can be assigned to the same room within the same time slot, whilst normally there is a one-to-one mapping of a course to a room [63]. A second difference is the number of time slots for course allocation in course timetabling are fixed (normally representing a standard teaching week), whereas in examination timetabling, the exams will be assigned to a set of permitted time slots whose number and structure vary across different institutions (normally spanning several weeks) [63]. Thirdly, in practice, there are usually many more constraints to consider within course timetabling in achieving a workable solution.

In this paper, these two university timetabling problem areas are investigated through the use of standard associated benchmark datasets (Carters un-capacitated examination datasets [7] and the Socha university course timetabling datasets [37], respectively). The performance of the proposed method is tested using these two problems in order to demonstrate the generality and robustness of the proposed method. The characteristics of exam and course timetabling datasets are illustrated in Tables I and II while the specification is presented in Sections II-A and II-B, respectively.

TABLE I
Carter's Examination Timetabling Benchmark Datasets

Datasets	Number of time slots	Number of exams	Number of students	Conflict density
car92	32	543	18,419	0.14
car91	35	682	16,925	0.13
ear83	24	190	1125	0.27
hec92	18	81	2823	0.42
kfu93	20	461	5349	0.06
lse91	18	381	2726	0.06
sta83	13	139	611	0.14
tre92	23	261	4360	0.18
uta92	35	622	21,267	0.13
ute92	10	184	2750	0.08
yor83	21	181	941	0.29

TABLE II
Socha Course Timetabling Benchmark Datasets

	Small	Medium	Large
Number of courses	100	400	400
Number of rooms	5	10	10
Number of timeslots	45	45	45
Number of features	5	10	10
Approx features per room	3	3	3
Percent feature use	70	80	90
Number of students	80	200	400
Max events per student	20	20	20
Max students per event	20	50	100

A. Examination Timetabling Problem

According to [3], examination timetabling problems can be defined as assigning a set of exams within a number of given timeslots while satisfying a set of predefined hard and soft constraints. The model of examination timetabling studied in this paper consists of:

N	number of exams;
M	number of students;
P	set of predefined timeslots;
Conflict matrix $c = (c_{ij})_{N\times N}$	where each element in the symmetrical matrix is the number of students that sit for both exams i and j, where i, $j \in 1, \ldots, N$;
t_i	timeslot scheduled to exam $i (i \in 1, \ldots, N)$ within the set of predefined timeslots $(1 \le t_i \le P)$.

The goal of this problem is to schedule all the exams to the timeslots, subject to hard and soft constraints as presented below.

1) *Hard Constraint:* Students are not required to attend more than one exam at the same time.
2) *Soft Constraint:* There should be a large time gap between two conflicting exams so that the students will have sufficient time for revision for their enrolled exams.

The quality of a feasible timetable is assessed based on the degree of soft constraint violation. For the datasets under consideration (1) is used to calculate the penalty for two consecutive exams enrolled for by a student. The assignment of a penalty value is based on the time gap(s) between two conflicting exams. For instance, if a student has two consecutive exams, then the proximity value of 16 ($2^5/2^1$)

will be calculated and counted. If a student has an empty slot between two exams, then proximity value of 8 ($2^5/2^2$) is counted. A proximity value is equal to 4 if the student has exams with two empty time slots in between and so on as

$$\text{Minimize} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} \cdot \text{prox}(t_i, t_j) / M$$

where

$$\text{prox}(t_i, t_j) = \begin{cases} 2^{5-|t_i - t_j|} & \text{if } 1 \leq |t_i - t_j| \leq 5 \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

B. Course Timetabling Problem

According to Socha *et al.* [37], university course timetabling problems can be defined as allocating a set of courses to a set of timeslots and rooms while satisfying a set of predefined hard and soft constraints. The model of the dataset, including the hard and soft constraints, is presented:

M	number of students;
C	number of courses;
T	set of predefined timeslots ($t_n \dots T$, where $T = 45$, $n = 1 \dots T$);
R	number of rooms ($r \dots R$, $r = 1, 2, 3, \dots, R$);
F	set of room features.

The problem of course timetabling is to allocate all the courses C, within the predefined number of timeslots T and rooms R such that all the following hard constraints are satisfied.

1) No student is required to attend two or more courses at the same time.
2) A course must be allocated to a room that satisfies the feature required for that course.
3) A course must be allocated to a room that can serve the students attending that course.
4) Only one course can be allocated in a room at any time slot.

Solutions that satisfy the hard constraints stated above are known as feasible solutions and violation of the following soft constraints should be minimized.

1) No student is required to attend only one course in a day.
2) No student is required to attend a course in the last time slot for a day.
3) No student is required to attend more than two consecutive courses in a day.

The penalty cost is calculated based on the violation of each soft constraint per student. Penalty cost = 1 for violation of each constraint per student.

III. General Idea of Basic Algorithms

A. ABC Algorithm

This section describes the conventional ABC algorithm that was proposed by Karaboga [56], which mimic the intelligent behavior of honey bees. ABC is classified as a bee swarm approach [56]. Generally, there are three types of bees that cooperate together in searching for food, i.e., employed, onlooker, and scout bees. Both onlooker and scout bees can be thought as unemployed bees. The employed bees search for food sources based on their memory and the information gathered on food sources is shared with onlooker bees on returning to the hive. Onlooker bees tend to choose the good food sources based on the information advertised by employed bees via a "waggle dance" and also further explore new food sources around the selected food sources. Scout bees abandon old food sources and explore new food sources. By mimicking the foraging behaviors of a natural honey bee swarm, an ABC algorithm has been developed by Karaboga [56] in addressing optimization problems. With this inherent search ability, a large number of application areas have adopted an ABC algorithmic approach in [41], [42], [59], and [64].

Swarm-based algorithms utilize agents which work in a collaborative manner in addressing problems. Therefore, each of the artificial bees (employed, onlooker, and scout) in the ABC algorithm represents an approach where agents communicate and cooperate with one another in exploring food sources (possible solution to optimization problems) by assessing the quality of the nectar (cost or fitness value for optimization problems). In addition, the ABC is categorized into two groups. The first half of the colony consists of employed bees and the second half comprises of the onlooker bees. Employed bees that abandon food sources are turned into scout bees with the purpose of discovering new food sources. Furthermore, the number of employed bees and onlooker bees is equal to the number of food sources (solutions) within a particular population which means that all the bees are working on the food sources in the population.

During the first step, an initial population (food source positions) is generated randomly from the search region. After the initialization, the solutions of this population are subjected to repeated cycles of search carried out by the three phases, i.e., employed, onlooker, and scout bee. In the employed bee phase, employed bees generate modifications on current solutions based on neighborhood search and also evaluate the nectar (fitness) of the new food source. If the amount of nectar of the new source is better than the old source, it will memorize the new source and disregard the old one. Otherwise the bee will retain the old food source. This particular food source acceptance scheme is known as greedy selection scheme.

After all employed bees have performed the search process, they go back to the hive and share the food source information with the onlookers in the "dance area." The onlooker bee phase is started when onlooker bees evaluate all the food sources information advertised by employed bees and select the food source based on the nectar amount. The food source selection is based on the roulette wheel selection scheme (2).

In nature, onlooker bees tend to choose the food sources advertised by employed bees that have a higher amount of nectar. To mimic this phenomenon, a roulette wheel selection scheme is used in the onlooker bee phase of the ABC where good quality solutions will have a higher probability of being selected by onlooker bees

$$p_i = \frac{\text{fit}_i}{\sum_{j=1}^{N} \text{fit}_j}. \quad (2)$$

With reference to (2), p_i and fit_i are the probability value and fitness value associated with food source i, respectively.

Similar to the employed bee phase, the onlooker bee phase generates a modification on the selected food source and memorizes the new source based on a greedy selection scheme. This process repeats until all onlookers complete the search process.

In the scout bee phase, if the quality of a food source is unable to be improved (inactive solution) in a predefined number of iterations, the corresponding employed bee will abandon the current food source and turn into a scout. The scout will explore for a new food source in the search region without any information. After a new source is identified, the scout will subsequently turn into an employed bee. It should be noted that both employed and onlooker bees generate new food sources by performing neighborhood search. Besides that, all food sources are improved by employed bees while onlookers only enhance selected food sources.

The ABC algorithm has been widely studied in relation to address a variety of real-world optimization problems such as flow shop scheduling [64] and university examination timetabling [41], [42]. However, the performance and convergence speed of the basic ABC are relatively poor. It is proposed here that this is due to the weaknesses of each of the phases within the basic ABC algorithm.

In the employed bee phase, the neighborhood search carried out is relatively slow causing poor exploitation ability and accordingly leads to the slow convergence of the search process. Within the onlooker bee phase, onlooker bees face the same problem since they are using the same neighborhood search in exploiting food sources. In addition, the roulette wheel selection scheme employed within the onlooker phase in selecting food sources can cause imbalance between exploitation and exploration within the search process. This is because this scheme is highly concentrated in exploiting solutions with higher fitness and solutions with poor fitness are less likely to be exploited. This might mean missing promising solutions.

In the scout bee phase, inactive food sources are identified and then possibly abandoned. Then, scout bees will replace the abandoned food sources by producing new food sources randomly. This can help the search process escape from local optima and direct exploration toward an un-visited search region. However, there is no guarantee that a promising search region will be located. Hence, this further decreases the exploitation ability and convergence speed of the ABC algorithm.

B. GD Algorithm

The GD algorithm is a metaheuristic approach first introduced by Dueck [62]. It is also known as a degraded ceiling approach and works in a similar fashion to simulated annealing. One of the abilities of GD is to accept worse solutions in order to escape from local optima during the search process. This ability is controlled by a variable called level, where any solution with quality value that is lower than the level will be accepted. The value of level is initially assigned based on the quality value of the initial solution used. Besides that, only one parameter is required which is the estimated quality of the desired solution. During execution, GD tries to improve and search for solutions with equal or greater quality than the estimated quality. In addition, the value of level degrades in line with a decay rate (based on estimated quality) so that the value of level is decreased until it reaches the same value as the estimated quality. This means that the use of level not only controls the acceptance of worse solutions, but is also used to guide the search process toward search regions that provide solutions equivalent to the original estimated quality. With the promising performance of GD, it has been widely applied within the timetabling research domain [22]–[27], [29], [30]. In this paper, a modification of GD is introduced and implemented with the aim of improving the exploitation ability of the ABC algorithm.

C. NMSS Method

The NMSS was proposed by Nelder and Mead [61] and is a local search approach designed for unconstrained optimization problems with N variables without the use of gradient information. A simplex consists of $N+1$ vertices in N dimensions. Different dimensions constitute different simplexes. For example, 2-D vertices constitute a triangle simplex, while it is a tetrahedron simplex in 3-D vertices. Four operations are used within NMSS (to rescale the simplex with respect to the current local behavior function). These are reflection, expansion, contraction [external contraction (EC) or internal contraction], and shrinkage. Note that a center point is needed for these operations. With these operations, the simplex can improve itself and eventually get nearer to the optimum. Improving itself means that improving the quality of solutions (the vertices that constitute the simplex) in searching for the local optimal solution. In the end, the vertex with the lowest cost value will be returned as the local optimal solution found. The details of the basic four operations for NMSS can be found in [61] and application of this method has been successfully hybridized with other methods in solving the inverse analysis problem [65] and university timetabling problems [26], [29]. In this paper, the purpose of hybridized NMSS into GD is to intelligently calculate the estimated quality for GD. It should be noted that, as described above, estimating quality is required prior to execution. Within the literature, NMSS has been applied in calculating the estimated quality for GD, but there remains some weaknesses where the calculated estimated quality might be a negative value based on the formula proposed in [26] and [29].

In this paper, the classic NMSS algorithm is modified for the purpose of assisting GD in calculating estimated quality so that GD searches toward promising regions in the problem search region. Triangle simplex is used in this paper and three operations are considered, which are EC, expansion (E), and reflection (R) operators. The operations for the modified NMSS are outlined as follows.

1) Identify three vertices x_b, x_s, and x_w from the current population, which correspond to the vertices that have best, second worst, and worst objective cost values (f_b, f_s, and f_w), respectively [65]. Through the use of x_b, x_s, and x_w, the centroid of simplex (C_{cent}) is

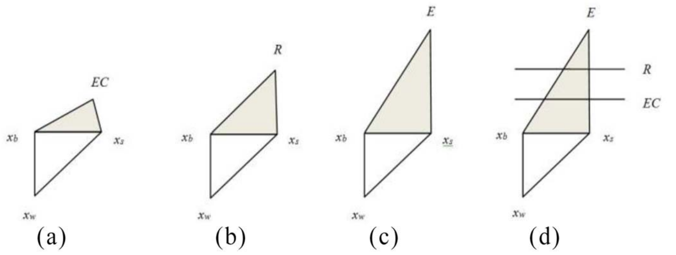


Fig. 1. NMSS operators in 2-D simplex. (a) EC. (b) Reflection (R). (c) Expansion (E). (d) Combination of EC, R, and E.

calculated using (3). Initialize three coefficients, which are reflection coefficient α, expansion coefficient γ, and contraction coefficient β

$$C_{\text{cent}} = \frac{f(x_b) + f(x_s) + f(x_w)}{3}. \tag{3}$$

2) EC

$$\text{EC} = [C_{\text{cent}} - (\beta \times C_{\text{cent}})]. \tag{4}$$

3) Reflection (R), where ($R <$ EC)

$$R = [\text{EC} - (\alpha \times C_{\text{cent}})]. \tag{5}$$

4) Expansion (E), where ($E < R$)

$$E = \left[R - (\gamma \times C_{\text{cent}})\right]. \tag{6}$$

The values of EC, R, and E are calculated based on (4)–(6), respectively. Since there are a number of solutions with different qualities in a population, all these values will be used as estimated qualities with the aim of assisting GD in better exploiting the search regions of different solutions. The triangle simplex used in this paper is illustrated in Fig. 1. The difference between the proposed NMSS in this paper with NMSS that have been applied to exam and course timetabling [26], [29] are listed below.

1) First, the equations for calculating EC, R, and E are different. Instead of selecting two adjacent solutions from the selected solutions in [26] and [29], the proposed NMSS uses best, second worst, and worst solutions in the population to calculate EC, R, and E.
2) Second, the values of EC, R, and E are used as estimated qualities for GD to improve all solutions in the population in this paper, but these values are only applied on a selected solution in [26] and [29] at each iteration.
3) Third, the calculated values of EC, R, and E in this paper always have a positive value. In previous studies, it is evident that negative values may arise [26], [29] though the authors did not articulate what will happen if these are used to estimate qualities for the GD.

IV. NMGD-ABC Algorithm

It is important to emphasize that both exploration and exploitation abilities are important elements for evolutionary algorithms. In order to maximize the performance, both abilities should be well balanced. In this paper, an NMGD-ABC algorithm is proposed with the aim of improving the performance (exploration and exploitation) of the basic ABC algorithm. To enhance exploration ability, the employed bee phase utilizes the global best model from PSO to direct exploration toward a promising search region. The explored regions are then exploited within the onlooker phase using a local search (an integration of NMSS within GD, NMGD) as proposed in this paper. Besides that, the selection scheme used in the basic ABC algorithm has also been eliminated to enhance the search process.

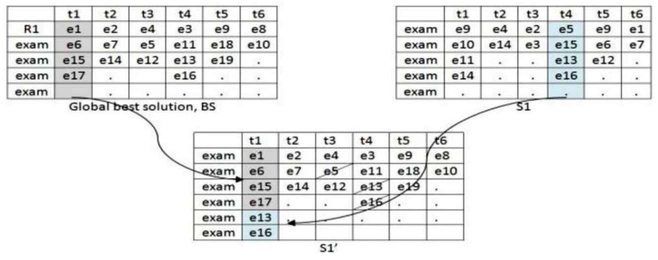
```
1.  Nelder-Mead Great Deluge Artificial Bee Colony Algorithm)
2.  (NMGD-ABC)
3.  Initialization:
4.  Set the population size, SN;
5.  Initialize the population;
6.  Calculate fitness value for each solution, f(sol);
7.  Identify global best solution, $sol_{BS}$;
8.  Set number of iterations, ABCNumIteration;
9.  Set value for parameter limit, limit;
10. Set, BeeLimitCounter for each solution (sol) ← 0;
11.
12. Improvement:
13. For n = 1 to ABCNumIteration do
14.     //Employed bee phase
15.     For i = 1 to SN do
16.         If ($sol_i$ is new solution generated by scout) do
17.             Incorporate information of best solution, $sol_{BS}$ into $sol_i$
18.             based on global best model from PSO;
19.         End If
20.     End For i
21.
22.     //Onlooker bee phase
23.     Improve all solutions using NMGD (Fig. 5.);
24.
25.     //Scout bee phase
26.     For i = 1 to SN do
27.         If (BeeLimitCounter for $sol_i \geq limit$)
28.             $sol_i$ is abandoned and new solution $sol_{new}$, is generated
29.             randomly;
30.             Set BeeLimitCounter for new solution $sol_{new}$ to 0,
31.             BeeLimitCounter ← 0;
32.             $sol_i \leftarrow sol_{new}$;
33.         End If
34.     End For i
35. End For n
```

Fig. 2. Framework for NMGD-ABC.

The framework of the proposed method can be seen in Fig. 2 and consists of two phases, i.e., the initialization and improvement phases. The improvement phase is further divided into three sub-phases which are employed bee, onlooker, and scout bee phases. A detailed presentation of the proposed approach is outlined below.

A. Initialization Phase

In lines 4–7 (refer to Fig. 2), the number of bees in the population is initialized, penalty values for all solutions are calculated, and the best solution in the population is identified. In constructing the initial solutions in the population, a hybridization of graph coloring heuristics is used where saturation degree, largest degree first, and largest enrolment heuristics



Global best solution, BS

	t1	t2	t3	t4	t5	t6
R1	e1	e2	e4	e3	e9	e8
exam	e6	e7	e5	e11	e18	e10
exam	e15	e14	e12	e13	e19	.
exam	e17	.		e16	.	.
exam		.	.	.	.	.

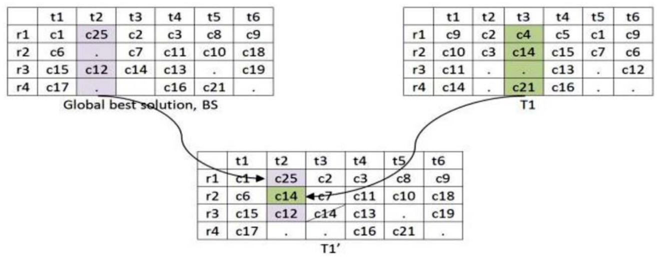
S1

	t1	t2	t3	t4	t5	t6
exam	e9	e4	e2	e5	e9	e1
exam	e10	e14	e3	e15	e6	e7
exam	e11	.	.	e13	e12	.
exam	e14	.	.	e16	.	.
exam	.	.	.	.	.	.

S1’

	t1	t2	t3	t4	t5	t6
exam	e1	e2	e4	e3	e9	e8
exam	e6	e7	~~e5~~	e11	e18	e10
exam	e15	e14	e12	~~e13~~	e19	.
exam	e17	.	.	~~e16~~	.	.
exam	e13	.	.	.	.	.
exam	e16					

Fig. 3. Haploid crossover for examination timetabling.



Global best solution, BS

	t1	t2	t3	t4	t5	t6
r1	c1	c25	c2	c3	c8	c9
r2	c6	.	c7	c11	c10	c18
r3	c15	c12	c14	c13	.	c19
r4	c17	.		c16	c21	.

T1

	t1	t2	t3	t4	t5	t6
r1	c9	c2	c4	c5	c1	c9
r2	c10	c3	c14	c15	c7	c6
r3	c11	.	.	c13	.	c12
r4	c14	.	c21	c16	.	.

T1’

	t1	t2	t3	t4	t5	t6
r1	c1	c25	c2	c3	c8	c9
r2	c6	c14	c7	c11	c10	c18
r3	c15	c12	~~c14~~	c13	.	c19
r4	c17	.	.	c16	c21	.

Fig. 4. Haploid crossover for course timetabling.

are considered simultaneously. Details on the implementation of this hybridization can be seen in [66]. In this solution construction phase, satisfaction of the hard constraint is considered solely. In lines 8 and 9, two user-defined parameters are initialized: 1) number of iterations for NMGD-ABC and 2) parameter limit, which corresponds to the maximum number of iterations for a solution that stays inactive.

B. Improvement Phase

In the improvement phase (lines 13–35), all bees (employed, onlooker, and scout) cooperate in exploring and exploiting the problem search region. Details of each function of the bees are as below.

1) Employed Bee Phase: The function (neighborhood search) utilized within the employed bee phase in the basic ABC has been changed based on the global best model inspired from PSO approach (Fig. 2, lines 15–20). By using this model, employed bees will explore for food sources that are located around the best solution region. Reasons for changing this basic employed bee function are as follows.

a) To overcome the slow convergence speed induced by the use of random exploration in the scout bee phase.
b) To increase the search efficiency since the best solution often carries better information as compared with others and search regions around it might be promising.

In order to exploit the global best model, haploid crossover [67] is used. The merit of haploid crossover is that the offspring generated contains information from the best solution, improving the search efficiency in exploring promising regions near to the best solution position. To perform the haploid crossover, two conditions must hold in order to preserve feasibility of the timetable solutions.

a) There should not exist conflicts between the moved and scheduled events.
b) An event can only be moved if the corresponding time slot and room are free and the room equipped with the features required for that event (for course timetabling since the room occupancy is considered during timetable construction).

Fig. 3 demonstrates the process of crossover in generating a new solution S1’ by combining S1 and best solution (BS), i.e., taking all exams of a selected timeslot from S1 and combining with another selected timeslot from BS. First, two random timeslots are selected which are t1 of BS and t4 of S1. Then, all the exams (i.e., e5, e13, e15, and e16) in time slot t4 are moved from S1 to timeslot t1 in BS. During this process, any exam conflicting with already scheduled exams in BS or exams that are already scheduled (i.e., e15) in the new time slot will not be moved. Then, a repair process is applied in order to maintain feasibility of the solution where duplicated exams in the new solution are removed (i.e., e5 in time slot t3 and e13 and e16 in time slot t4).

For course timetabling, the crossover operation is presented as in Fig. 4. As before, the shaded time slot represents the selected time slot for time slot exchange process. For example, t3 and t2 are selected from T1 and BS, respectively. Then, the entire moveable courses from t3 (c14) in this example are move into t2 which will then produce an offspring, T1’. The course c21 cannot move to t2 because it conflicts with course c12. For course c4, it cannot be moved since there is no free room (occupied by course c25 in BS). As with the examination timetabling example, a repair process is needed to maintain the feasibility of the generated offspring. In this example (refer to T1’), course c14 is scheduled twice in different time slots (t2 and t3) and rooms (r2 and r3). Hence, the old time slot (t3) and room (r3) for course c14 will be removed to avoid course duplication. For both timetabling problems, the crossover process is controlled by a crossover point which is the number of time slots for the crossover process. For instance, a crossover point of 1 is used in the examples demonstrated in Figs. 3 and 4, in which only one time slot is selected for the crossover process.

Therefore, by implementing the global best model within the employed bee phase, information relating to the best solution found so far can be incorporated within the new solutions generated in the scout bee phase. This can direct the search process toward promising areas that are near to the best solution position and improve the convergence speed. In addition, this can also provide a good environment for onlooker bees later in the searching process for better quality solutions to be found around the promising search regions.

2) Onlooker Bee Phase: With reference to Fig. 2, line 23 represents the beginning of the onlooker bee phase. In this phase, onlookers exploit the search regions using the described NMGD algorithm and the selection mechanism used within the basic ABC algorithm is eliminated. Hence, all food sources will be improved by onlookers. Details of NMGD are shown in Fig. 5 and a description of NMGD is demonstrated in Section IV.

Two neighborhood structures are used in NMGD to generate tentative solutions, which are as follows.

Nb 1: Randomly selects and moves an exam/course into a feasible time slot (and room for course timetabling).

Nb 2: Randomly selects two exams/courses and swap their time slots (and rooms for course timetabling). Feasibility of the solution is preserved at the same time.

As discussed earlier, the reasons for using the NMGD are as follows.

a) To overcome the imbalanced exploitation and weak neighborhood search by locally exploring entire solutions in the population in order to intensify the search for local minimum solutions.
b) To eliminate reliance on the use of the selection scheme (roulette wheel selection) in basic ABC where the selection scheme tends to focus more on exploiting the fittest solution.
c) To improve the convergence speed of the search process.

3) Scout Bee Phase: In the scout bee phase (lines 26–34), inactive solutions are discarded and new solutions generated randomly to replace the abandoned solutions (lines 28 and 29). An inactive solution is determined based on the predetermined number of iterations (parameter limit) in which the quality of a solution cannot be improved.

C. NMGD Algorithm

As discussed in Section I, the NMGD algorithm is a hybrid local search (GD with NMSS) that is introduced with the aim of improving the exploitation ability of basic ABC. It is implemented within the onlooker bee phase to further explore the promising search regions identified in the employed bee phase. During the exploitation process, NMGD will improve the quality of each solution based on a set of estimated qualities calculated using NMSS. After the estimated qualities are calculated, NMGD will select the estimated quality with respect to the quality of the current solution and the exploitation process will begin. If the quality of the solution reaches the estimated quality before the termination criterion is met, another estimated quality will be selected from the set of estimated qualities and the exploitation process continues. Otherwise NMGD tries to improve the quality of the solution with respect to the selected estimated quality. This exploitation process is repeated until all the solutions are improved and a new set of estimated qualities will be calculated in the next cycle of the process. There are two advantages to use multiple estimated qualities within NMGD. First, it helps NMGD exploit multiple search regions of a single solution. Second, it also enables NMGD to exploit solutions with different qualities in the population. This is very important because when only using one estimated quality as with the basic GD, it is not possible to fine-tune solutions with different search regions in a population.

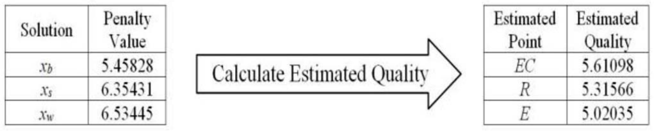
Therefore, the main purpose of hybridizing the NMSS method is to intelligently calculate the estimated quality value of the GD (pseudo code of NMGD can be seen at Fig. 5). Prior to the execution of GD, NMSS is used to compute the values of estimated quality (EC, *R*, and *E*) and the decay rates for the entire solutions in the population. After the values of EC, *R*, and *E* are calculated, three more regions are formed and divided equally between the EC–*R* and *R*–*E* (lines 4–8). The regions between EC–*R* are known as EC1, EC2, and EC3 whereas the regions between *R*–*E* are known

```
1.  NM-Great Deluge Algorithm (NMGD)
2.
3.  Initialization:
4.  Calculate estimated qualities based on Nelder-Mead Simplex
5.  method (external contraction EC, Reflection R and Expansion
6.  E based on Eq. (3-6));
7.  Calculate estimated qualities of EC1, EC2, EC3, R1, R2 and R3
8.  from EC, R and E;
9.  Set best solution in the population in $sol_{best}$;
10. Set total number of iterations for Great Deluge algorithm,
11. GD_Iteration;
12.
13. Improvement:
14. For i = 1 to SN do
15.     Set solution i as initial solution, $sol_i$;
16.     Set initial level, level $\leftarrow$ f($sol_i$);
17.     Set GDIterCounter $\leftarrow$ 0;
18.     Calculate Nelder-Mead decay rate, $\beta_{EC}$, $\beta_R$, $\beta_E$, $\beta_{EC1}$, $\beta_{EC2}$, $\beta_{EC3}$,
19.     $\beta_{R1}$, $\beta_{R2}$, and $\beta_{R3}$ based on estimated qualities,
20.     (f($sol_i$)- estimated quality)/GD_Iteration;
21.     Decay rate $\beta \leftarrow \beta_{EC}$;
22.
23.     For j = 1 to GD_Iteration do
24.         Determine estimated quality based on penalty cost of $sol_i$;
25.         $\beta \leftarrow$ decay rate of estimated quality that is nearest and smaller
26.         than $sol_i$ ($\beta_{EC}$, $\beta_R$, $\beta_E$, $\beta_{EC1}$, $\beta_{EC2}$, $\beta_{EC3}$, $\beta_{R1}$, $\beta_{R2}$ or $\beta_{R3}$);
27.         Apply a random neighborhood structure (Nb1 or Nb2) on $sol_i$
28.         to generate new solution sol*;
29.         Calculate penalty value for Sol*, f(sol*);
30.         If(f(sol*) < f($sol_{best}$))
31.             $sol_i \leftarrow$ sol*;
32.             $sol_{best} \leftarrow$ sol*;
33.         Else
34.             If(f(sol*) $\leq$ level)
35.                 $Sol_i \leftarrow$ sol*;
36.             End if
37.         End If
38.         If f($sol_{BS}$) $\leq$ f(sol*) //update global best solution
39.             $sol_{BS} \leftarrow$ sol*;
40.         End If
41.
42.         Level $\leftarrow$ level – $\beta$;
43.
44.     End For j
45.
46.     If no improvement on quality of $sol_i$
47.         Increase BeeLimitCounter for $sol_i$ by 1;
48.     End If
49. End For i
```

Fig. 5. Hybridization of NMSS with GD (NMGD).

as *R*1, *R*2, and *R*3. These ranges of values are used as the estimated qualities and also to calculate the decay rates (lines 18–20) in GD when enhancing the quality of solutions in the population (lines 23–44).

An example (taken from execution on car92 Carter dataset) is shown in Fig. 6 where three solutions (x_b, x_s, and x_w correspond to the best, second worst, and worst solutions) are selected to calculate the estimated qualities for GD (EC, *R*, and *E*). Table III demonstrates how NMGD selects the estimated quality based on the penalty value



Solution	Penalty Value
x_b	5.45828
x_s	6.35431
x_w	6.53445

Estimated Point	Estimated Quality
EC	5.61098
R	5.31566
E	5.02035

Fig. 6. Example of calculating estimated qualities for GD using best, second worst, and worst solutions.

TABLE III
ESTIMATED QUALITY DETERMINATION FOR NMGD

GD_Iteration	Penalty Value	Estimated Quality for GD
sol₁		
1	5.45828	*EC3* - 5.38949
2	5.44183	*EC3* - 5.38949
…		
1000	5.16537	*R3* - 5.09418
…		
2000	5.14165	*R3* - 5.09418

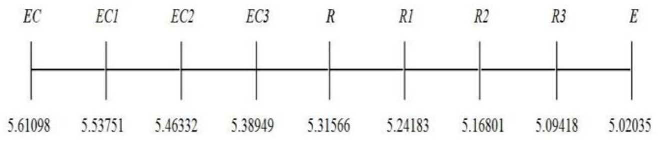


Fig. 7. Ranges of estimated qualities for GD.

of a solution. After the ranges of estimated qualities are calculated (refer to Fig. 6), these values are used in GD with the aim of enhancing the quality of the solution. For instance, during the first GD iteration, the sol_1 possesses a penalty value of 5.45828 (objective function of the problem in this paper), thus the estimated quality of the new solution (refer to estimated quality of EC3 in Fig. 7) is 5.38949 (i.e., value that is nearest and smaller than current penalty value). The GD will attempt to enhance the quality of the solution to the value of EC3. On reaching the 1000th iteration, the penalty value of sol_1 has been reduced to 5.16537, therefore the estimated quality of the solution will be changed to 5.09418 (estimated quality of R3). After the whole population is enhanced by GD, the values of EC, R, and E will be recalculated based on the best, second worst, and worst solutions in the current population. With reference to lines 46–48, if a solution is unable to be improved by GD, the value of bee limit counter for the solution will be increased by 1.

V. COMPUTATIONAL EXPERIMENT

Experiments on the proposed method in addressing the university timetabling have been conducted. The datasets used were described in Section II and the proposed method was coded using C++. Table IV illustrates the parameters used in the proposed method which were established after some preliminary experiments. Even though all these parameter values cannot be regarded as optimal values, they represent general settings for ABC and NMGD-ABC to perform well over the two benchmarks. The experiments are carried out with a population size of 50, the number of iterations for both the basic ABC and the proposed approach is set to 10 000, and the number of iterations for NMGD is set to 2000. Note that the limit

TABLE IV
PARAMETER SETTING FOR NMGD-ABC

No.	Parameters	Values
1.	Iteration number for NMGD-ABC	10 000
2.	Iteration number for ABC	10 000
3.	Iteration number for NMGD	2000
4.	*limit* (scout bee) for NMGD-ABC	4
5.	*limit* (scout bee) for ABC	100
4.	reflection coefficient, α	0.05
5.	expansion coefficient, γ	0.05
6.	contraction coefficient, β	0.05
7.	Crossover (haploid) points	8

used in scout bee phase was 4 (no improvement gained after four attempts within the onlooker bee phase). From a preliminary experimentation, it was observed that the bigger the value of reflection coefficient α, expansion coefficient γ, and contraction coefficient β, the bigger the gaps between the estimated qualities (EC, R, and E). This leads to worse quality solutions being obtained as compared with the use of smaller values for α, γ, and β. Therefore, with the use of small values of the three coefficients, the range of EC, R, and E is small. This has induced a smaller decay rate (a smaller decay rate slows the rate at which level is decreased) and enabled GD to easily accept new solutions exhibiting better quality (better exploitation in the search region of each solution).

Furthermore, the preliminary experiments show that increasing the number of crossover points increases the exploitation toward the best solution search region, and accordingly decreases the diversity of the population, inducing premature convergence of the search process. The decreasing of diversity in the population can also lead to the search process cyclically exploring the same search region. In contrast, if the number of crossover points is too small, the search within the best solution region will decrease and exhibit random exploration characteristics.

A. *Experiments on Carter's Un-Capacitated University Examination Benchmark Dataset*

The application of NMGD-ABC on university examination benchmark dataset (Table I) has been carried out and is subsequently described. Experimental results for 30 runs on both algorithms with different random seeds are presented and comparison is made with best known results in the literature on 11 instances, as shown in Table V. The computational times are varied and depend on the size of each instance. For basic ABC, the computational times took 1–3 h, whereas, NMGD-ABC required 2–10 h (see Table VI). This computational time is acceptable for university timetabling problems because the timetables are usually generated several months before the actual timetable is used [4], [47]. As shown in Table V, the results obtained by NMGD-ABC are better than the basic ABC (best results are highlighted in bold font). Besides that, NMGD-ABC is able to obtain three of the best known results (equal with [38] on uta92 instance) in the literature.

In a second experiment, the results generated by NMGD-ABC are compared with results obtained by single, population-based, and hybrid approaches in the literature

TABLE V
RESULT COMPARISON WITH BEST KNOWN RESULTS

Datasets	ABC		NMGD-ABC		Best known results	
	Best	Avg.	Best	Avg.		
car92	4.80	4.82	**3.89**	4.27	3.90	Sabar *et al.* [38]
car91	5.60	5.76	4.79	4.85	**4.50**	Yang and Petrovic. [69]
ear83	37.47	38.88	33.43	34.48	**29.30**	Caramia *et al.* [21]
hec92	11.58	11.90	10.49	10.61	**9.20**	Caramia *et al.* [21]
kfu93	15.25	15.54	13.72	13.76	**13.00**	Burke *et al.* [47]
lse91	12.48	12.87	10.29	10.39	**9.60**	Caramia *et al.* [21]
sta83	158.27	159.01	157.07	157.37	**156.90**	Burke *et al.* [47]
tre92	8.85	8.97	**7.86**	8.04	7.87	Sabar *et al.* [38]
uta92	3.75	3.89	**3.10**	3.31	**3.10**	Sabar *et al.* [38]
ute92	27.65	28.41	25.33	26.04	**24.40**	Caramia *et al.* [21]
yor83	40.21	40.55	36.12	36.83	**34.90**	Burke *et al.* [47]

TABLE VI
COMPUTATIONAL TIME FOR CARTER DATASET

Datasets	ABC		NMGD-ABC	
	Best	Time (min)	Best	Time (min)
car92	4.80	121	3.89	521
car91	5.60	132	4.79	552
ear83	37.47	113	33.43	391
hec92	11.58	82	10.49	231
kfu93	15.25	92	13.72	365
lse91	12.48	72	10.29	254
sta83	158.27	61	157.07	178
tre92	8.85	109	7.86	414
uta92	3.75	161	3.10	588
ute92	27.65	79	25.33	210
yor83	40.21	88	36.12	420

and are shown in Table VII (best results are highlighted in bold font). The selected approaches under comparison are as follows.

1) *H1: Sabar et al. [38]:* Honey bee mating optimization algorithm.
2) *H2: Turabieh and Abdullah [27]:* Hybridization of electromagnetic-like mechanism with GD.
3) *H3: Burke et al. [47]:* Hybridization of variable neighborhood search with genetic algorithm.
4) *H4: Abdullah et al. [20]:* Large neighborhood search with local search.
5) *H5: Caramia et al. [21]:* Novel iterated local search algorithm.
6) *H6: Pillay and Banzhaf [34]:* Informed genetic algorithm.
7) *H7: Burke and Newall [22]:* Adaptive ordering initialization with GD algorithm.
8) *H8: Qu and Burke [54]:* Graph-based hyper-heuristic.
9) *H9: Merlot et al. [52]:* Hybridization of constraint programming, simulated annealing, and hill climbing.

From Table VII, it can be seen that the NMGD-ABC is capable of generating competitive results as compared with the approaches reported in the literature. In addition, NMGD-ABC outperforms other methods on car92 and tre92 instances and obtains the same result on uta92 with H1 [38]. Also, NMGD-ABC obtains the second best result on yor83 and the third best results on car91, ear83, sta83, and ute92 against other methods. In light of the above, it can be seen that NMGD-ABC is capable of generating competitive results when compared against other proposed approaches in the literature.

TABLE VII
RESULT COMPARISON WITH OTHER METHODS IN THE LITERATURE FOR CARTER'S EXAMINATION TIMETABLING DATASET

Datasets	NMGD-ABC	H1	H2	H3	H4
car92	**3.89**	3.90	4.10	3.90	4.40
car91	4.79	4.79	4.80	4.60	5.20
ear83	33.43	34.69	34.92	32.80	34.90
hec92	10.49	10.66	10.73	10.00	10.30
kfu93	13.72	**13.00**	**13.00**	**13.00**	13.50
lse91	10.29	10.00	10.01	10.00	10.20
sta83	157.07	157.04	158.26	**156.90**	159.20
tre92	**7.86**	7.87	7.88	7.90	8.40
uta92	**3.10**	**3.10**	3.20	3.20	3.60
ute92	25.33	25.94	27.00	24.80	26.00
yor83	36.12	36.15	36.22	**34.90**	36.20
Datasets	H5	H6	H7	H8	H9
car92	6.00	4.20	4.10	4.16	4.30
car91	6.66	4.90	4.65	5.16	5.10
ear83	**29.30**	35.90	37.05	35.86	35.10
hec92	**9.2**	11.50	11.54	11.94	10.60
kfu93	13.80	14.40	13.90	14.79	13.50
lse91	**9.60**	10.90	10.82	11.15	10.5
sta83	158.20	157.80	168.73	159.00	157.30
tre92	9.40	8.40	8.35	8.60	8.40
uta92	3.50	3.40	3.20	3.59	3.50
ute92	**24.40**	27.20	25.83	28.30	25.10
yor83	36.20	39.30	37.28	41.81	37.40

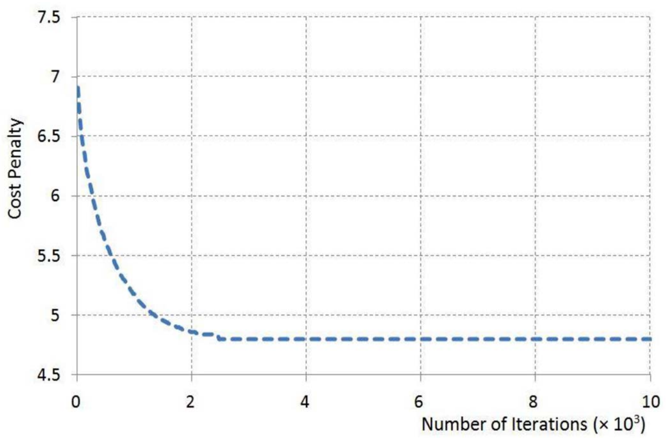


Fig. 8. Convergence graph for car91.

Figs. 8 and 9 illustrate the convergence graphs for instances car91 and sta83, respectively. The y-axis represents the penalty cost value while the x-axis represents the number of iterations. Each point on the graph indicates the current best penalty value obtained during the search process. From both Figs. 8 and 9, there is a steep descent curve at the beginning of the search process which indicates that there is a large improvement in terms of solution quality (penalty cost value). Nevertheless, the improvement rate decreases as the number of iterations increases and eventually the search converges, showing no further improvement.

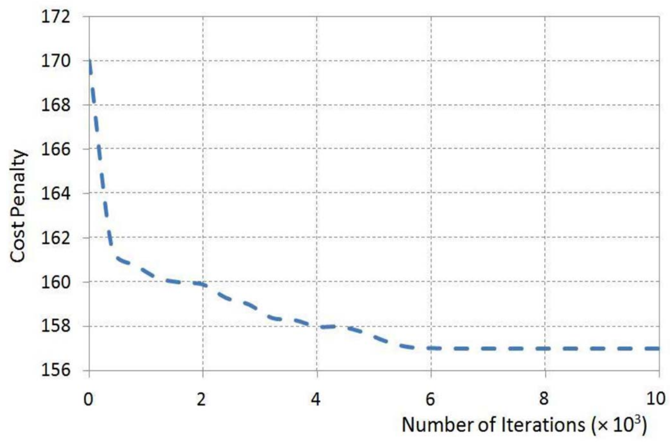


Fig. 9. Convergence graph for sta83.

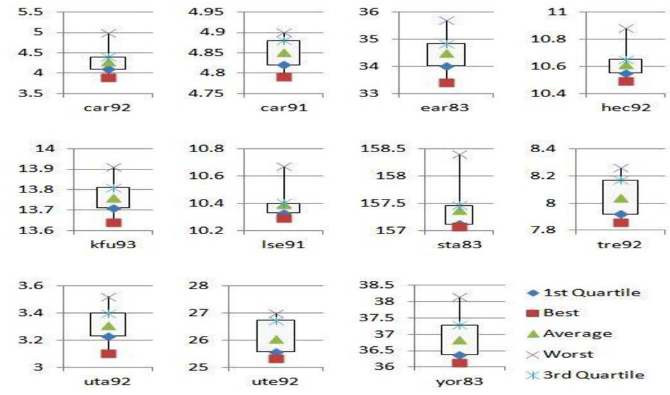


Fig. 10. Box plots for Carter's un-capacitated dataset.

Fig. 10 shows the box plots that represent the distribution of the best, first quartile, average, third quartile, and worst solution qualities generated from NMGD-ABC for the Carter's un-capacitated dataset. From Fig. 10, it can be observed that there is a close gap between the best, average, and worst solution qualities, illustrating the robustness of the proposed method (except for car92, ear83, ute92, sta83, and yor83 instances where the differences between best and worst cost values are between 1 and 2.29). It is thought that the use of the global best model might lead to cycling, i.e., the search process repeatedly explores the same solution search regions. An example can be seen from Fig. 9 where there is no improvement between iterations 3854–4956 and iterations 6483–10 000. Hence, the search process is unable to prevent cycling and gets trapped in a less promising search region and accordingly causes the difference between the best and worst cost values (for car92, ear83, ute92, sta83, and yor83 instances) to be slightly higher as compared with other instances.

B. Experiments on Socha Course Timetabling Benchmark Dataset

The characteristics of the Socha course timetabling dataset can be seen in Table II. The benchmark consists of 11 instances which have been categorized into five small, five medium, and one large problem instances. Thirty runs were carried out for each of the instances for NMGD-ABC and the parameter setting used is shown in Table IV. Table VIII represents the results (best and average penalty values) for the basic ABC and NMGD-ABC algorithms and shows comparison with the best known results in the literature. The computational time for each instance is dependent on the size of the instance. For the basic ABC, the computational time for each instance ranges from 15 min to 2 h. For NMGD-ABC, the time taken to complete each search is between 1 and 6 h (refer to Table IX).

TABLE VIII
RESULT COMPARISON FOR BASIC ABC AND NMGD-ABC WITH BEST KNOWN RESULTS

Datasets	ABC		NMGD-ABC		Best known results	
	Best	Avg.	Best	Avg.		
small 1	25	31.03	**0**	0	**0**	Many
small 2	32	36.70	**0**	0	**0**	Many
small 3	23	26.57	**0**	0	**0**	Many
small 4	15	20.83	**0**	0	**0**	Many
small 5	24	28.67	**0**	0	**0**	Many
medium 1	393	433.67	57	70.63	**50**	Turabieh and Abdullah [44]
medium 2	436	470.50	**54**	79.53	70	Turabieh and Abdullah [44]
medium 3	483	499.80	114	132.03	**102**	Turabieh and Abdullah [44]
medium 4	422	443.93	74	82.60	**32**	Turabieh and Abdullah [44]
medium 5	415	455.73	64	75.43	**61**	Turabieh and Abdullah [44]
large	908	991.67	**502**	549.90	523	Sabar *et al.* [38]

TABLE IX
COMPUTATIONAL TIME FOR SOCHA DATASET

Datasets	ABC		NMGD-ABC	
	Best	Time (min)	Best	Time (min)
small 1	25	17	0	45
small 2	32	19	0	51
small 3	23	19	0	55
small 4	15	15	0	56
small 5	24	17	0	53
medium 1	393	45	57	195
medium 2	436	57	54	203
medium 3	483	69	114	225
medium 4	422	58	74	218
medium 5	415	72	64	254
large	908	106	502	351

From Table VIII, it is clear that NMGD-ABC outperforms the basic ABC algorithm. Even though the best results for small instances are the same (zero penalty cost) for NMGD-ABC, it is relatively more stable as compared with basic ABC where the best and average results for small instances are the same. Importantly, NMGD-ABC is able to generate best results for medium 2 and large instances when compared against the best known results in the literature.

Table X illustrates the result comparison between NMGD-ABC with single solution-based, population-based, and hybridization approaches in the literature. Selected methods in comparison are as follows.

1) *M1: Turabieh and Abdullah [44]:* A tabu-based memetic approach.
2) *M2: Abdullah and Turabieh [33]:* Hybridization of genetic algorithm and local search.
3) *M3: Abdullah et al. [46]:* A hybrid evolutionary algorithm.

TABLE X
RESULT COMPARISON WITH OTHER METHODS IN THE LITERATURE FOR SOCHA COURSE TIMETABLING DATASET

Datasets	NMGD-ABC	M1	M2	M3	M4
small 1	**0**	**0**	2	**0**	3
small 2	**0**	**0**	4	**0**	4
small 3	**0**	**0**	2	**0**	6
small 4	**0**	**0**	**0**	**0**	6
small 5	**0**	**0**	4	**0**	**0**
medium 1	57	**50**	254	221	140
medium 2	**54**	70	258	147	130
medium 3	114	**102**	251	246	189
medium 4	74	**32**	321	165	112
medium 5	64	**61**	276	130	141
large	**502**	653	1027	529	876
Datasets	M5	M6	M7	M8	M9
small 1	**0**	6	**0**	**0**	**0**
small 2	**0**	7	**0**	**0**	**0**
small 3	**0**	3	**0**	**0**	**0**
small 4	**0**	3	**0**	**0**	**0**
small 5	**0**	4	**0**	**0**	**0**
medium 1	78	372	317	75	175
medium 2	92	419	313	88	197
medium 3	135	359	357	129	216
medium 4	75	348	245	74	149
medium 5	68	171	292	64	190
large	556	1068	-	523	912

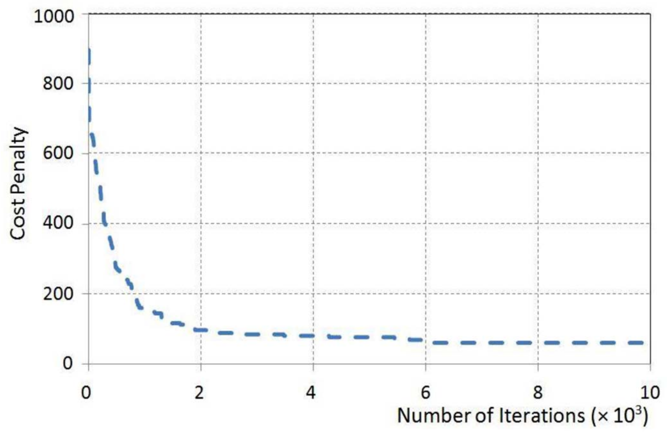


Fig. 11. Convergence graph for instance medium 4.

4) *M4: Landa-Silva and Obit [32]:* A nonlinear GD algorithm.
5) *M5: Abdullah et al. [53]:* GD and tabu search.
6) *M6: Burke et al. [55]:* A graph-based hyper-heuristic.
7) *M7: Abdullah et al. [19]:* Variable neighborhood search.
8) *M8: Sabar et al. [38]:* Honey bee mating optimization algorithm.
9) *M9: Turabieh et al. [28]:* A hybridization of electromagnetic-like mechanism with GD.

Referring to Table X, it can be seen that NMGD-ABC shares the same best results in all small instances with the best known results in the literature and also outperforms other approaches for medium 2 and large instances. Besides that, NMGD-ABC is able to obtain feasible solutions for all problem instances. A result denoted as "-" indicates that there is no feasible solution obtained for that particular instance.

Fig. 11 represents the convergence graph for instance medium 4. The y-axis represents the penalty cost value while x-axis represents the number of iterations. The trend of the graph for the best cost value at each iteration is almost the same as Figs. 8 and 9 where there is a large improvement (curve with a steep slope) at the beginning of the search with the cost value decreasing gradually until no improvement is shown by the end of the search process. It is clearly evident that NMGD-ABC manages to perform well on different problems (exam and course) which have different complexity and solution search regions.

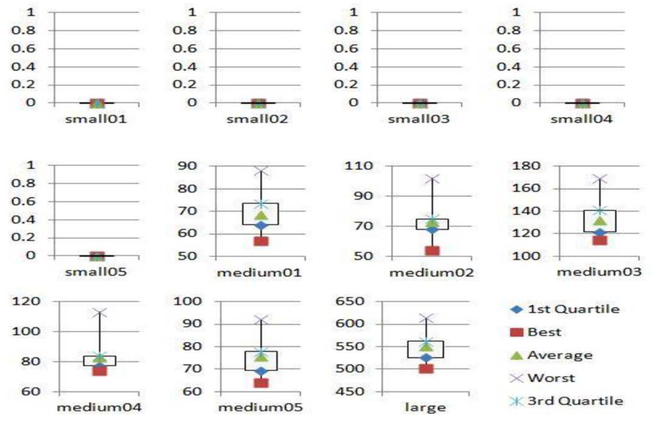


Fig. 12. Box plots for Socha course datasets.

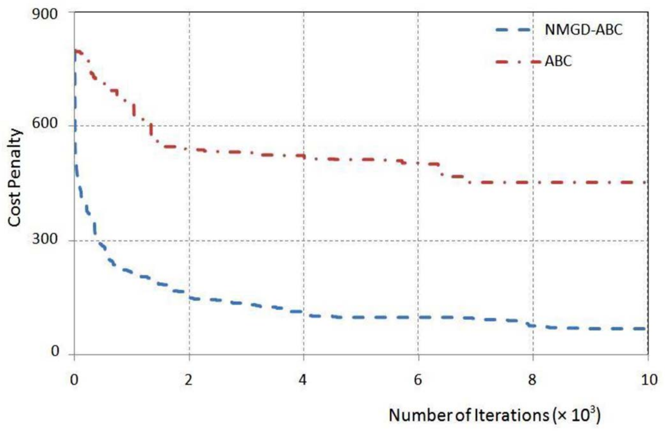


Fig. 13. Convergence graph for NMGD-ABC and basic ABC for instance medium 5.

Fig. 12 demonstrates the box plots for all instances of the Socha datasets. For small instances, the gaps between the best, averages, first and third quartiles, and worst cost values are zero. It can be observed that there is a relatively small gap between best and average cost values for medium and large instances. However, there are slightly bigger gaps between the best and the worst cost values for medium 02, 03, and large instances (31, 55, and 112, respectively). Again, it is believed that in some cases, the use of the global best model might lead the search to cycle in exploring the same problem search region.

Fig. 13 (similarly to Figs. 8, 9, and 11, but with minor differences) represents the convergence graph for NMGD-ABC and basic ABC for instance medium 5. From Fig. 13, it can be observed that the slope of the curve for NMGD-ABC is relatively steep as compared with basic ABC which illustrates that there is great improvement in the quality of the solutions at

TABLE XI
STATISTICAL ANALYSIS FOR BASIC ABC AND NMGD-ABC ON CARTER'S UN-CAPACITATED EXAMINATION TIMETABLING PROBLEM

Datasets	ABC		NMGD-ABC		*t*-test
	Best	Avg.	Best	Avg.	*p*-value
car92	4.80	4.82	3.89	4.27	**4.3179 E–25**
car91	5.60	5.76	4.79	4.85	**1.6224 E–25**
ear83	37.47	38.88	33.43	34.48	**1.0741 E–33**
hec92	11.58	11.90	10.49	10.61	**5.0754 E–28**
kfu93	15.25	15.54	13.72	13.76	**4.1528 E–28**
lse91	12.48	12.87	10.29	10.39	**7.2240 E–40**
sta83	158.27	159.01	157.07	157.37	**5.4266 E–24**
tre92	8.85	8.97	7.86	8.04	**3.5143 E–36**
uta92	3.75	3.89	3.10	3.31	**6.2934 E–28**
ute92	27.65	28.41	25.33	26.04	**9.6786 E–22**
yor83	40.21	40.55	36.12	36.67	**6.8075 E–38**

TABLE XII
STATISTICAL ANALYSIS FOR BASIC ABC AND NMGD-ABC ON SOCHA COURSE TIMETABLING PROBLEM

Datasets	ABC		NMGD-ABC		*t*-test
	Best	Avg.	Best	Avg.	*p*-value
small 1	25	31.03	0	0	**7.8873E–28**
small 2	32	36.70	0	0	**3.4334 E–30**
small 3	23	26.57	0	0	**1.3339 E–28**
small 4	15	20.83	0	0	**6.1954 E–25**
small 5	24	28.67	0	0	**4.0640 E–27**
medium 1	393	433.67	57	70.63	**1.5173 E–52**
medium 2	436	470.50	54	79.53	**5.0787 E–59**
medium 3	483	499.80	114	132.03	**8.1385 E–65**
medium 4	422	443.93	74	82.60	**1.7776 E–73**
medium 5	415	455.73	64	75.43	**7.8315 E–50**
large	908	991.67	502	549.90	**5.2785 E–53**

the beginning of the search process. In addition, NMGD-ABC converges faster as compared with basic ABC and the quality of solutions found by NMGD-ABC is better than those found by the basic ABC across the search process.

From the analysis above, it can be seen that the proposed approach NMGD-ABC is able to produce high-quality solutions for both exam and course timetabling problems. It is believed that the use of global best model inspired from PSO in the employed bee phase leads the search process toward promising areas within the search region (global exploration ability). In addition, the hybridization of NMSS with GD (NMGD) means the onlooker bee phase is able to intelligently tune the estimated qualities during the execution of GD (local exploitation ability). Hence, NMGD is able to locally explore solution search regions that are near to the global best solution in order to seek better quality solutions. Note that the estimated quality for the standard GD is a predefined constant.

Tables XI and XII present an experimental comparison between the basic ABC and NMGD-ABC algorithms. A *t*-test has been carried out with a one-tail test to show that the performance (in terms of penalty value) of NMGD-ABC is better than the basic ABC at a level of confidence of 0.05. The null hypothesis (H_0) is defined as there is no difference between the performances of both approaches. For the alternative hypothesis (H_1), it has been defined as the performance of NMGD-ABC is better than that of the basic ABC. By referring to Tables XI and XII, there is significant evidence to support the claim on H_1 (H_0 is rejected) due to all the values of p-value are smaller than 0.05 (highlighted in bold font) which indicates there is a significant difference across all instances for both benchmarks. Hence, it can be concluded that the performance of NMGD-ABC is better than the basic ABC in solving both timetabling problems.

In general, NMGD-ABC performs well on both examination and course timetabling benchmark datasets, producing better and competitive results as compared with the existing approaches published in the literature. Even though the proposed approach requires long running times to complete the search process, it is clear that the final solutions obtained are much better than the solutions generated by basic ABC. It is noteworthy that in order to get better quality solutions, longer computational times are required. It should also be noted that it is often the case that computation time is not reported within the literature, rendering a full comparison based on time impossible. The fact that the approach is able to produce quality solutions for all instances of the examination and course timetabling benchmark datasets is testimony to the robustness of NMGD-ABC.

VI. CONCLUSION

In this paper, a hybrid NMGD-ABC has been proposed to tackle university exam and course timetabling problems. With the use of PSO global best model within the employed bee phase, all the solutions generated in the scout bee phase incorporate global best information, thus the search process is guided toward the most promising search region. NMGD (GD with intelligent estimated quality calculation using NMSS) is applied to locally explore and fine-tune promising solutions within the onlooker bee phase. In short, the proposed method is capable of improving both the exploration and exploitation abilities and the convergence speed of the basic ABC. It has been shown that this hybrid approach significantly improves the effectiveness of the basic ABC.

The basic ABC and proposed method (NMGD-ABC) were applied on two benchmark problems, which are Carter's un-capacitated examination timetabling datasets and the Socha course timetabling datasets. Datasets from two different areas of educational timetabling are deliberately chosen to illustrate the generality of the approach. Experimental results demonstrate that NMGD-ABC outperforms the basic ABC in producing good quality solutions. This is because NMGD-ABC is able to globally explore and locally exploit the solution search region using a PSO inspired global best model and NMGD, respectively. In addition, a statistical analysis is also carried out to prove that the performance of NMGD-ABC is significantly better than the basic ABC. The exploration ability of NMGD-ABC can be further improved by incorporating better exploration mechanisms and is the subject of future work.

ACKNOWLEDGMENT

The authors would like to thank Dr. P. McMullan for his kindness in reviewing and explaining important points for this paper.

REFERENCES

[1] 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.,* vol. 198, no. 2, pp. 392–404, 2009.

[2] M. W. Carter and G. Laporte, "Recent developments in practical examination timetabling," in *Practice and Theory Automated Timetabling* (LNCS 1153), E. K. Burke and P. Ross, Eds. Berlin, Germany: Springer, 1996, pp. 1–21.

[3] R. Qu, E. K. Burke, B. McCollum, L. T. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *J. Sched.,* vol. 12, no. 1, pp. 55–89, 2009.

[4] B. McCollum, "A perspective on bridging the gap between theory and practice in university timetabling," in *Practice and Theory of Automated Timetabling VI* (LNCS 3867), E. Burke and H. Rudová, Eds. Berlin, Germany: Springer, 2007, pp. 3–23.

[5] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *Eur. J. Oper. Res.,* vol. 140, no. 2, pp. 266–280, 2002.

[6] M. W. Carter and G. Laporte, "Recent developments in practical course timetabling," in *Practice and Theory of Automated Timetabling II* (LNCS 1408), E. K. Burke and M. Carter, Eds. Berlin, Germany: Springer, 1998, pp. 3–19.

[7] M. W. Carter, G. Laporte, and S. Y. Lee, "Examination timetabling: Algorithmic strategies and applications," *J. Oper. Res. Soc.,* vol. 47, no. 3, pp. 373–383, 1996.

[8] K. Dowsland, "Off-the-peg or made-to-measure? Timetabling and scheduling with SA and TS," in *Practice and Theory of Automated Timetabling II* (LNCS 1408), E. K. Burke and M. Carter, Eds. Berlin, Germany: Springer, 1998, pp. 37–52.

[9] J. Thompson and K. Dowsland, "Variants of simulated annealing for the examination timetabling problem," *Ann. Oper. Res.,* vol. 63, no. 1, pp. 105–128, 1996.

[10] J. Thompson and K. Dowsland, "General cooling schedules for a simulated annealing based timetabling system," in *Practice and Theory of Automated Timetabling* (LNCS 1153), E. K. Burke and P. Ross, Eds. Berlin, Germany: Springer, 1996, pp. 345–363.

[11] J. M. Thompson and K. A. Dowsland, "A robust simulated annealing based examination timetabling system," *Comput. Oper. Res.,* vol. 25, nos. 7–8, pp. 637–648, 1998.

[12] S. Abdullah, S. Ahmadi, E. K. Burke, M. Dror, and B. McCollum, "A tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem," *J. Oper. Res. Soc.,* vol. 58, no. 11, pp. 1494–1502, 2007.

[13] S. Abdullah and H. Turabieh, "On the use of multineighbourhood structures within a tabu-based memetic approach to university timetabling problems," *Inf. Sci.,* vol. 191, pp. 146–168, May 2012.

[14] G. Kendall and N. Hussin, "An investigation of a tabu-search-based hyper-heuristic for examination timetabling," in *Proc. Multidiscipl. Int. Sched. Conf. Theory Appl.*, New York, NY, USA, 2005, pp. 309–328.

[15] A. M. A. Malik, M. Ayob, and A. R. Hamdan, "Iterated two-stage multi-neighbourhood tabu search approach for examination timetabling problem," in *Proc. 2nd Conf. Data Min. Optim.*, Selangor, Malaysia, 2009, pp. 141–148.

[16] A. M. A. Malik, M. Ayob, and A. R. Hamdan, "Stratified random sampling technique for integrated two-stage multi-neighbourhood tabu search for examination timetabling problem," in *Proc. 10th Int. Conf. Intell. Syst. Design Appl.*, Cairo, Egypt, 2010, pp. 1326–1331.

[17] A. M. A. Malik, A. K. Othman, M. Ayob, and A. R. Hamdan, "Hybrid integrated two-stage multi-neighbourhood tabu search-EMCQ technique for examination timetabling problem," in *Proc. 3rd Conf. Data Min. Optim.*, Putrajaya, Malaysia, 2011, pp. 232–236.

[18] L. D. Gaspero and A. Schaerf, "Tabu search techniques for examination timetabling," in *Practice and Theory of Automated Timetabling III* (LNCS 2079), E. K. Burke and W. Erben, Eds. Berlin, Germany: Springer, 2001, pp. 104–117.

[19] S. Abdullah, E. K. Burke, and B. McCollum, "An investigation of variable neighbourhood search for university course timetabling," in *Proc. 2nd Multidiscipl. Int. Conf. Sched. Theory Appl.*, New York, NY, USA, 2005, pp. 413–427.

[20] S. Abdullah, S. Ahmadi, E. K. Burke, and M. Dror, "Investigating Ahuja–Orlin's large neighbourhood search approach for examination timetabling," *OR Spectr.,* vol. 29, no. 2, pp. 351–372, 2007.

[21] M. Caramia, P. Dell'Olmo, and G. F. Italiano, "New algorithms for examination timetabling," in *Algorithm Engineering* (LNCS 1982), S. Näher and D. Wagner, Eds. Berlin, Germany: Springer, 2001, pp. 230–242.

[22] E. K. Burke and J. Newall, "Enhancing timetable solutions with local search methods," in *Practice and Theory of Automated Timetabling IV* (LNCS 2740), E. K. Burke and P. Causmaecker, Eds. Berlin, Germany: Springer, 2003, pp. 195–206.

[23] Y. Bykov, "Time-predefined and trajectory-based search: Single and multiobjective approaches to exam timetabling," Ph.D. dissertation, Dept. Comput. Sci., Univ. Nottingham, Nottingham, U.K., 2003.

[24] 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 *Proc. 4th Multidiscipl. Int. Sched. Conf. Theory Appl.*, Dublin, Ireland, 2009, pp. 424–434.

[25] P. McMullan, "An extended implementation of the great deluge algorithm for course timetabling," in *Proc. 7th Int. Conf. Comput. Sci.*, Beijing, China, 2007, pp. 538–545.

[26] H. Turabieh and S. Abdullah, "A hybrid fish swarm optimisation algorithm for solving examination timetabling problems," in *Learning and Intelligent Optimization* (LNCS 6683), C. C. Coello, Ed. Berlin, Germay: Springer, 2011, pp. 539–551.

[27] H. Turabieh and S. Abdullah, "An integrated hybrid approach to the examination timetabling problem," *Omega,* vol. 39, no. 6, pp. 598–607, 2011.

[28] H. Turabieh, S. Abdullah, and B. McCollum, "Electromagnetism-like mechanism with force decay rate great deluge for the course timetabling problem," in *Rough Sets and Knowledge Technology* (LNCS 5589), P. Wen *et al.,* Eds. Berlin, Germany: Springer, 2009, pp. 497–504.

[29] H. Turabieh, S. Abdullah, B. McCollum, and P. McMullan, "Fish swarm intelligent algorithm for the course timetabling problem," in *Rough Set and Knowledge Technology* (LNCS 6401), J. Yu, S. Greco, P. Lingras, G. Wang, and A. Skowron, Eds. Berlin, Germany: Springer, 2010, pp. 588–595.

[30] S. Abdullah, H. Turabieh, and B. McCollum, "A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems," in *Hybrid Metaheuristics* (LNCS 5818), M. Blesa *et al.,* Eds. Berlin, Germany: Springer, 2009, pp. 60–72.

[31] E. K. Burke, Y. Bykov, J. Newall, and S. Petrovic, "A time-predefined approach to course timetabling," *Yugoslav J. Oper. Res.,* vol. 13, no. 2, pp. 139–151, 2003.

[32] D. Landa-Silva and J. Obit, "Evolutionary non-linear great deluge for university course timetabling," in *Hybrid Artificial Intelligence Systems*. Berlin, Germany: Springer, 2009, pp. 269–276.

[33] S. Abdullah and H. Turabieh, "Generating university course timetable using genetic algorithms and local search," in *Proc. 3rd Int. Conf. Converg. Hybrid Inf. Technol.*, Busan, Korea, 2008, pp. 254–260.

[34] N. Pillay and W. Banzhaf, "An informed genetic algorithm for the examination timetabling problem," *Appl. Soft Comput.,* vol. 10, no. 2, pp. 457–467, 2010.

[35] M. Eley, "Ant algorithms for the exam timetabling problem," in *Practice and Theory of Automated Timetabling VI* (LNCS 3687), E. K. Burke and H. Rudová, Eds. Berlin, Germany: Springer, 2007, pp. 364–382.

[36] K. Socha, J. Knowles, and M. Sampels, "A MAX-MIN ant system for the university course timetabling problem," in *Proc. 3rd Int. Workshop Ant Algorithms*, Brussels, Belgium, 2002, pp. 1–13.

[37] K. Socha, M. Sampels, and M. Manfrin, "Ant algorithms for the university course timetabling problem with regard to the state-of-the-art," in *Applications of Evolutionary Computing* (LNCS 2611), S. Cagnoni *et al.*, Eds. Berlin, Germany: Springer, 2003, pp. 334–345.

[38] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A honey-bee mating optimization algorithm for educational timetabling problems," *Eur. J. Oper. Res.,* vol. 216, no. 3, pp. 533–543, 2012.

[39] M. A. Al-Betar and A. T. Khader, "A harmony search algorithm for university course timetabling," *Ann. Oper. Res.*, vol. 194, no. 1, pp. 3–31, 2012.

[40] M. A. Al-Betar, A. T. Khader, and M. Zaman, "University course timetabling using a hybrid harmony search metaheuristic algorithm," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.,* vol. 42, no. 5, pp. 664–681, Sep. 2012.

[41] 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.,* vol. 1, no. 5, pp. 158–163, 2011.

[42] M. Alzaqebah and S. Abdullah, "Hybrid artificial bee colony search algorithm based on disruptive selection for examination timetabling problems," in *Combinatorial Optimization and Applications* (LNCS 6831), W. Wang, X. Zhu, and D.-Z. Du, Eds. Berlin, Germany: Springer, 2011, pp. 31–45.

[43] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.,* vol. 35, no. 3, pp. 268–308, 2003.

[44] H. Turabieh and S. Abdullah, "Incorporating tabu search into memetic approach for enrolment-based course timetabling problems," in *Proc. 2nd Conf. Data Min. Optim.*, Selangor, Malaysia, 2009, pp. 115–119.

[45] B. McCollum *et al.*, "Setting the research agenda in automated timetabling: The second international timetabling competition," *INFORMS J. Comput.,* vol. 22, no. 1, pp. 120–130, 2010.

[46] S. Abdullah, E. K. Burke, and B. McCollum, "A hybrid evolutionary approach to the university course timetabling problem," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Singapore, 2007, pp. 1764–1768.

[47] 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.,* vol. 206, no. 1, pp. 46–53, 2010.

[48] S. Abdullah, K. Shaker, B. McCollum, and P. McMullan, "Dual sequence simulated annealing with round-robin approach for university course timetabling," in *Evolutionary Computation Combinatorial Optimization* (LNCS 6022), P. Cowling and P. Merz, Eds. Berlin, Germany: Springer, 2010, pp. 1–10.

[49] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan, "A hybrid metaheuristic approach to the university course timetabling problem," *J. Heuristics,* vol. 18, no. 1, pp. 1–23, 2012.

[50] P. Côté, T. Wong, and R. Sabourin, "A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem," in *Practice and Theory of Automated Timetabling V* (LNCS 3616), E. K. Burke and M. Trick, Eds. Berlin, Germany: Springer, 2005, pp. 294–312.

[51] H. S. F. Irene, S. Deris, M. Hashim, and S. Zaiton, "University course timetable planning using hybrid particle swarm optimization," in *Proc. 1st ACM/SIGEVO Genet. Evol. Comput.*, Shanghai, China, 2009, pp. 239–246.

[52] L. G. Merlot, N. Boland, B. Hughes, and P. Stuckey, "A hybrid algorithm for the examination timetabling problem," in *Practice and Theory of Automated Timetabling IV* (LNCS 2740), E. K. Burke and P. Causmaecker, Eds. Berlin, Germany: Springer, 2003, pp. 207–231.

[53] S. Abdullah, K. Shaker, B. McCollum, and P. McMullan, "Construction of course timetables based on great deluge and tabu search," in *Proc. 8th Int. Conf. Meteheuristic*, Hamburg, Germany, 2010, pp. 13–16.

[54] R. Qu and E. K. Burke, "Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems," *J. Oper. Res. Soc.,* vol. 60, no. 9, pp. 1273–1285, 2009.

[55] 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.,* vol. 176, no. 1, pp. 177–192, 2007.

[56] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Dept. Comput. Sci., Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, 2005.

[57] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," *Inf. Process. Lett.,* vol. 111, no. 17, pp. 871–882, 2011.

[58] W.-F. Gao and S.-Y. Liu, "A modified artificial bee colony algorithm," *Comput. Oper. Res.,* vol. 39, no. 3, pp. 687–697, 2012.

[59] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Appl. Math. Comput.,* vol. 217, no. 7, pp. 3166–3173, 2010.

[60] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, Perth, WA, Australia, 1995, pp. 1942–1948.

[61] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.,* vol. 7, no. 4, pp. 308–313, 1965.

[62] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *J. Comput. Phys.,* vol. 104, no. 1, pp. 86–92, 1993.

[63] R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems," *OR Spectr.,* vol. 30, no. 1, pp. 167–190, 2008.

[64] M. F. Tasgetiren, Q.-K. Pan, P. N. Suganthan, and A. H. L. Chen, "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops," *Inf. Sci.,* vol. 181, no. 16, pp. 3459–3475, 2011.

[65] F. Kang, J. Li, and Q. Xu, "Structural inverse analysis by hybrid simplex artificial bee colony algorithms," *Comput. Struct.,* vol. 87, no. 13, pp. 861–870, 2009.

[66] 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.,* vol. 36, no. 4, pp. 981–1001, 2009.

[67] H. A. Abbass, "A monogenous MBO approach to satisfiability," in *Proc. Int. Conf. Comput. Intell. Model. Control Autom. (CIMCA)*, Las Vegas, NV, USA, 2001.

[68] Y. Yang and S. Petrovic, "A novel similarity measure for heuristic selection in examination timetabling," in *Practice and Theory of Automated Timetabling V* (LNCS 3616), E. Burke and M. Trick, Eds. Berlin, Germany: Springer, 2005, pp. 247–269.

Cheng Weng Fong received the B.Sc. and Ph.D. degrees in computer science from Universiti Teknologi Malaysia, Johor Bahru, Malaysia, in 2012 and 2014, respectively.

He is a Senior Lecturer with the Faculty of Applied Sciences and Computing, Tunku Abdul Rahman University College, Johor Branch Campus, Segamat, Malaysia. His research interests include timetabling and scheduling.

Hishammuddin Asmuni received the B.Sc. degree from Universiti Malaya, Kuala Lumpur, Malaysia; the M.Sc. degree from Universiti Teknologi Malaysia, Johor Bahru, Malaysia; and the Ph.D. degree from University of Nottingham, Nottingham, U.K., in 2008, all in computer science.

He is a Senior Lecturer with the Faculty of Computing, Universiti Teknologi Malaysia. His research interests include timetabling, scheduling, bioinformatics, and iris recognition.

Barry McCollum received the B.Sc. (Hons.), M.Sc., and Ph.D. degrees from the School of Electronics, Electrical Engineering and Computer Science, Queen's University, Belfast, U.K., in 1992, 1994, and 2000, respectively.

He is a Senior Lecturer with the School of Electronics, Electrical, Engineering, and Computer Science, Queen's University Belfast, Belfast, U.K. His research interests include scheduling, timetabling, and complex fields of optimization.