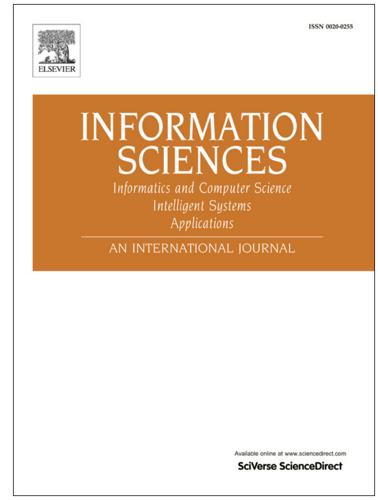


A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems

Cheng Weng Fong, Hishammuddin bin Asmuni, Barry McCollum, Paul McMullan, Sigeru Omatu

PII: S0020-0255(14)00599-4
DOI: http://dx.doi.org/10.1016/j.ins.2014.05.039
Reference: INS 10908

To appear in: *Information Sciences*

Received Date: 19 October 2013
Revised Date: 13 March 2014
Accepted Date: 20 May 2014

Please cite this article as: C.W. Fong, H.b. Asmuni, B. McCollum, P. McMullan, S. Omatu, A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems, *Information Sciences* (2014), doi: http://dx.doi.org/10.1016/j.ins.2014.05.039

A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems

Cheng Weng Fong[a], Hishammuddin bin Asmuni[a,*], Barry McCollum[b], Paul McMullan[b], Sigeru Omatu[c]

[a] *Software Engineering Research Group, Software Engineering Department, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia*
[b] *Department of Computer Science, Queen's University Belfast, Belfast BT7 1NN United Kingdom*
[c] *Department of Electronic, Information and Communication Engineering, Osaka Institute of Technology, Osaka 535-8585, Japan*

E-mail address: chengweng0410@hotmail.com (CW. Fong), hishamudin@utm.my (H. Asmuni), b.mccollum@qub.ac.uk (B. McCollum), p.p.mcmullan@qub.ac.uk (P. McMullan), omatu@rsh.oit.ac.jp (S. Omatu)

Highlights

- A new approach that improves exploration and exploitation abilities of basic Artificial Bee Colony (ABC) algorithm.
- Proposed approach is the combination of ABC with assimilation policy and hybridization of two local searches.
- Assimilation policy is to lead search process explores toward multiple promising search region.
- The hybrid local search (Great Deluge with Nelder-Mead Simplex Search) is to exploit the search region for each solution.

Abstract

Generating timetables for an institution is a challenging and time consuming task due to different demands on the overall structure of the timetable. In this paper, a new hybrid method which is a combination of a great deluge and artificial bee colony algorithm (INMGD-ABC) is proposed to address the university timetabling problem. Artificial bee colony algorithm (ABC) is a population based method that has been introduced in recent years and has proven successful in solving various optimization problems effectively. However, as with many search based approaches, there exist weaknesses in the exploration and exploitation abilities which tend to induce slow convergence of the overall search process. Therefore, hybridization is proposed to compensate for the identified weaknesses of the ABC. Also, inspired from imperialist competitive algorithms, an assimilation policy is implemented in order to improve the global exploration ability of the ABC algorithm. In addition, nelder-mead simplex search method is incorporated within the great deluge algorithm (NMGD) with the aim of enhancing the exploitation ability of the hybrid method in fine-tuning the problem search region. The proposed method is tested on two differing benchmark datasets i.e. examination and course timetabling datasets. A statistical analysis *t*-test has been conducted and shows the performance of the proposed approach as significantly better than basic ABC algorithm. Finally, the experimental results are compared against

[*] Corresponding author. Tel: +607-5538803; fax: +607-5538822

state-of-the art methods in the literature, with results obtained that are competitive and in certain cases achieving some of the current best results to those in the literature.

Keywords: Artificial bee colony, Great deluge algorithm, University timetabling, Imperialist competitive algorithm

1. Introduction

University timetable construction has long been a challenging administrative task for educational institutions worldwide. This, in part, is due to the large numbers of subjects offered by an institution, increasing number of students and the requirement to satisfy side constraints which differ across educational institutions. Generally, university timetabling can be defined as assigning a set of events into a number of permitted timeslots and given resources subject to a set of constraints [50]. Constraints within university timetabling can be divided into two types, namely hard and soft constraints [22]. Hard constraints must be satisfied under all circumstances in order to preserve feasibility of the timetable solution while violation of defined soft constraints should be minimized. It is typical that the quality of solution is assessed based on the degree of soft constraints satisfaction. A timetable solution that satisfies hard constraints is known as feasible solution (i.e. clash free timetable). Overviews and information on research directions regarding the university timetabling problems can be seen at [26, 27, 38, 49, 50, 52].

By referring to the survey papers [26, 27, 49, 50], it can be observed that educational timetabling problems have attracted researchers from the operational research and artificial intelligence for a few decades. Various approaches have been introduced and applied in tackling education timetabling problems. At an early stage, sequential graph coloring heuristics [27, 28], based on graph theory, were implemented and proved capable of generating timetable solutions quickly. However, the performance of these approaches is relatively poor as compared with meta-heuristic approaches.

Several research works have demonstrated that meta-heuristic approaches have a better performance in addressing university timetabling problems [9, 11, 21]. Basically, meta-heuristic approaches can be categorized into two: single-solution based and population based approaches [18]. The single-solution based approaches manipulate one solution through the use of neighborhood structures during the search process in order to reach better solutions. The effectiveness of these approaches highly depends on the definition of the neighborhood structures (Note that the greater number of used neighborhood structures, the better the connection in the search region although the price to pay is they are computational expensive [55]. Even though these approaches possess the strength of fine-tuning the given solution, the main shortcoming is that they easily get trapped in local optima as compared with population-based approaches [18]. Examples of single-solution based approaches that have been applied in solving university timetabling problems include simulated annealing [20, 55, 56], great deluge algorithm [19, 20, 37, 44], tabu search [9, 62] and large neighborhood search [3-5].

Population-based approaches manipulate multiple solutions at any one time [18]. During the solution searching process, they usually generate new solutions by using information for the solutions in the population or by utilizing simple local search approaches (some approaches use both). Example of population based approaches that have been successfully applied in solving optimization problems are genetic algorithm [70], cuckoo search [30], honey-bee mating algorithm [51], fish swarm algorithm [58, 61] and harmony search [13]. However, the quality of the solutions generated is usually poorer than solutions generated using single-solution based approaches. This is due to population-based approaches are focused in exploring the search region rather than fine-tuning the promising search region in order to locate local optimal solution [13].

Hybridization methods have attracted the interest of many researchers in recent years due to its outstanding performance in solving optimization problems. The main idea of hybrid methods is to combine constructive elements of different methods into one. With this combination, the searching abilities of the hybrid method will be increased. Blum and Roli [18] summarized that "In summary, population-based methods are better in identifying promising areas in the search space, whereas trajectory methods are better in some way manage to combine the advantage of population-based methods with the strength of trajectory methods are often very successful". For instance, Turabieh and Abdullah [59] developed a hybrid approach for university examination timetabling problems that combined electromagnetic-like algorithm with the great deluge algorithm. Turabieh *et al*. [61] presents a hybrid method which covered fish swarm optimization, hill climbing and great deluge algorithm in solving university course timetabling problem. Other hybridization methods that are presented in solving university timetabling domain can be seen at [8, 9, 12, 57, 58, 60]. Other than university timetabling, the effectiveness of hybrid methods can also seen in solving other optimization problems in the literature [64-69].

In this paper, inspired from imperialist competitive algorithm (ICA), assimilation policy is hybridized with an artificial bee colony algorithm (ABC) in order to enhance the exploration power so that the search process explores regions of multiple best solution points. Besides that, in order to avoid candidate solutions converging toward the respective best solution points, an assimilation policy probability (*AP*) is introduced for the sake of balancing the exploration by scout bee and the assimilation policy exploration. Furthermore, in order to improve exploitation, an integration of nelder-mead simplex search (NMSS) [46] within the great deluge (GD) [29] algorithm (NMGD [31] for short) is hybridized with ABC algorithm. The modification assimilation policy with the NMGD constitutes the variant of ABC (INMGD-ABC), which eliminated the probabilistic selection mechanism in onlooker bees.

In order to evaluate the performance of ABC and the proposed approach, two benchmark datasets were selected for experiment: Carter un-capacitated examination timetabling dataset [28] and Socha course timetabling dataset [53]. Experimental results illustrate that the proposed approach is capable of producing good quality solutions as compared with ABC algorithm and other approaches that have been reported in the literature.

The rest of the paper is organized as follows. Firstly, description of the educational timetabling problem (examination and course) is presented in Section 2. The basic idea of the ABC algorithm and proposed algorithm are presented in Section 3 and Section 4, respectively. Section 5 demonstrates and discusses the experimental results and a brief conclusion regarding this paper is presented in Section 6.

2. Problem Description

Generally, university timetabling can be divided into two problem areas: course and examination timetabling. These two problems are similar to each other in some regard, but there are still some significant differences between them. The major difference is, with examination timetabling, a number of conflict free exams can be allocated into the same room within the same timeslot provided there is enough seating capacity, whilst in course timetabling, only one course is allocated to a room within any time slot [38]. The second difference is the examination timetabling is usually carried out within a set of predefined time slots over the course of a limited number of weeks, whereas course timetabling requires allocation over an entire semester or term [38].

In light of above, the performance of the proposed approach is tested against both examination and course timetabling problems in order to illustrate the degree of generality as well as the robustness of the proposed approach. There are several timetabling benchmarks available in the literature such as the timetabling problems which were introduced in the International Timetabling Competition 2007 (ITC 2007). There are three tracks of problems in ITC 2007 [42, 43] which cover exam and course (postenrollment and curriculum based) timetabling problems. In this paper, two university timetabling problems are investigated. The selected university course and examination timetabling problems are Socha university course timetabling [53] and Carter un-capacitated examination timetabling datasets [28], respectively. One reason for using these two benchmarks is to provide the concept of using these with an eye to expanding to other benchmarks to explore the generality of the approach. Detailed explanations and specifications for both timetabling problems (course and exam) are presented in Section 2.1 and 2.2, respectively.

2.1. Course timetabling problems

By referring to Socha et al. [53], the definition of university course timetabling problem is defined as assigning a set of courses into a set of permitted time slots and subject to a set of predetermined constraints. The characteristics of the Socha dataset are shown in Table 1 and can be modeled as:

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

Table 1
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

The goal of the course timetabling problem is to schedule all the courses *C* into the permitted time slots *T*, such that rooms *R* satisfy all the hard constraints. The hard constraints that are considered in Socha course timetabling dataset are:

- No student is required to attend more than one course at the same time;
- A course can only allocate to a room that satisfies features required by the course;
- A course can only allocate to a room with sufficient capacity for students attending the course;
- Only one course can be allocated into one room at any time slot.

Solutions generated that satisfy all the hard constraints stated above are known as feasible timetables. After a feasible timetable is obtained, the overall quality (penalty cost) is further improved based on the satisfaction of soft constraints and therefore any associated violations should be reduced. Soft constraints which could incur penalties in the Socha course timetabling dataset are where:

- a student is required to attend more than two consecutive courses in a day;

- a student is scheduled to attend one class on a day;
- a student is required to attend a course that scheduled in last time slot of a day.

The quality of a timetable is measured based on the number of violations of the soft constraints described above. A penalty cost of 1 is applied for each violation of the soft constraints per student.

2.2. Exam timetabling problems

The timetabling problem investigated in this study was introduced by Carter et al. [28] which is an un-capacitated examination timetabling problem where the room capacity constraint is overlooked during generating a timetable solution. The dataset consists of real-world instances from different educational institutions and is considered as a well established and widely accepted benchmark dataset.

Table 2
Carter examination timetabling benchmark datasets.

Datasets	Number of time slots	Number of exams	Number of students	Conflict density
car-f-92	32	543	18,419	0.14
car-s-91	35	682	16,925	0.13
ear-f-83	24	190	1125	0.27
hec-s-92	18	81	2823	0.42
kfu-s-93	20	461	5349	0.06
lse-f-91	18	381	2726	0.06
sta-f-83	13	139	611	0.14
tre-s-92	23	261	4360	0.18
uta-s-92	35	622	21,267	0.13
ute-s-92	10	184	2750	0.08
yor-f-83	21	181	941	0.29

Based on Qu et al. [50], examination timetabling problems can be defined as the allocation of exams into a set of permitted time slots and satisfying a set of constraints (hard and soft) at the same time. The hard and soft constraints considered in Carter un-capacitated examination dataset are:

- Hard constraint: No students are required to sit two or more exams simultaneously;
- Soft constraint: To evenly spread the conflicting exams so that students can have more time for revision.

In Carter un-capacitated exam datasets [28], the quality of a feasible timetable is calculated based on the degree of satisfaction of soft constraints. Eq. (1) is formulated to calculate the penalty for two consecutive exams to which a student is enrolled, where:

- N, number of exams;
- M, number of students;
- P, a predefined timeslots;
- Conflict matrix, $c=(c_{ij})_{N\times N}$ where each element in the matrix is the students that attend exams i and j, where i, $j \in 1,\ldots,N\}$;
- t_i is the timeslots that scheduled exam i ($j \in 1,\ldots,N$) within the set of predefined timeslots ($1 \leq t_i \leq P$).

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

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

The penalty value assigned is based on the time gap(s) between the allocated time slots of two conflicting exams, where *prox*(t_i, t_j)=16, 8, 4, 2, 1 if the time gaps are 0, 1, 2, 3 and 4, respectively. For example, if a student has two consecutive exams (time gap equals to 0), then a proximity value of 16 will be assigned. If a student has one free time slot in between two exams, a proximity value of 8 will be assigned. A proximity value of 4 is applied if there are only two free time slots between exams, and so on.

3. Artificial Bee Colony Algorithm

A large number of meta-heuristic approaches that are based on natural phenomena and imitate real-life conditions have been proposed and applied in solving real world problems. An example of approaches that mimic natural phenomena are genetic algorithms inspired from gene evolution, ant colony algorithm imitating ant search behavior, fish swarm optimization based on fish food-search behavior, etc. In this section, conventional ABC algorithm introduced by Karoboga [36] is described.

Generally, the ABC algorithm inspired from the foraging behavior of the honey bee. There are three types of bee to cooperate in the food searching process: employed, onlooker and scout bees. Onlooker and scout bees can also be thought of as unemployed. During the foraging process, the employed bee will search for a new food source and information of a newly found source will be shared with the onlooker bee when they return to the hive. The Onlooker bee will observe and select food sources with an improved quality search based on information advertised by the employed bee via a "waggle dance" and further search for new food sources near to the selected food sources. Scout bees will abandon old food sources and randomly search for new food sources.

In swarm based approaches, there is a group of agents that work collaboratively in solving problems. Hence, all the artificial bees (employed, onlooker and scout) in the ABC algorithm represent an agent which they communicates and cooperates in searching for food sources (potential solutions for optimization problems) with good quality of nectar (cost or objective function values for optimization problems).

To start the ABC algorithm, an initial population which includes employed, onlooker and scout bees (each bee is attached with one food source) is initialized. One half of the population consists of employed bees whereas the second half constitutes the onlooker bees. Employed bees will turn into scouts if they unable to explore for better food sources after a certain period of time. In the employed bee phase, all employed bees will explore for new food sources. The new food sources are accepted if their quality is equal or better than current food sources (greedy selection scheme). When it comes to the onlooker bee phase, onlooker bees will select food sources advertised by employed bees. The selection is based on the roulette wheel selection scheme which will base selection on the quality of each food source. As with the employed bee, the onlooker bee accepts the new food source using greedy selection scheme. Lastly, if an employed bee is unable to improve the quality of the food sources in a predefined number of iterations, then the bee will turn into a scout bee. Scout bees will abandon old food sources and randomly look for new food sources in the search region. This food searching process will repeat until a termination criterion is met. The food sources abandon process is controlled by a parameter *limit*, which is the number of iterations for a particular food source in which improvement has not been achieved. For the food sources improvement process, both employed and onlooker bees explore new food sources by performing a neighborhood search on current food sources.

Research on the ABC algorithm has been widely studied and applied in tackling different types of real world optimization problems such as flow shop scheduling [54] and university examination timetabling [14, 15]. However, there are still some identified weaknesses that degrade its overall performance. The greedy selection scheme utilized in neighborhood search of employed and onlooker bees are unable to effectively fine tune the problem search region. This is due to the fact that the greedy selection scheme accepts only those solutions with better or equivalent quality, and can become trapped in local optima [44]. In addition, random solution generation within the scout bee phase can mean the search blindly explores the problem search region. Furthermore, the roulette wheel selection scheme that is employed in the onlooker bee phase makes the search process focus on exploiting promising solutions. This is based on the fact that in roulette wheel selection scheme, solutions with better quality have a higher probability of selection as compared with poor quality solutions. This may miss out some promising search regions that can be reached via those solutions. Hence, a modified ABC algorithm is proposed to address the weaknesses of each of the phases in the basic ABC. Even though ABC suffers from several inefficiencies, it is still an effective approach as it can manipulate several solutions during the search, increasing the chances of reaching good solution. Additionally, the effectiveness of exploration of ABC allows the search process to more easily escape from local optima. This is a very important issue since single solution based approaches usually suffer from getting tapped in local optima, and the algorithm will stop progressing.

3.1. Great Deluge Algorithm

The Great deluge (GD) algorithm is a single-solution based meta-heuristic proposed by Dueck [29]. GD is similar to simulated annealing (SA) to some extent in that both of the approaches accept inferior solutions based on an acceptance rule during the solution search process in order to escape from local optima. The inferior solution acceptance rule is controlled by a variable called *level*, where any inferior solution with a penalty cost value that is lower than *level* will be accepted. Hence, GD is able to explore a wider solution search region during the solution searching process. Generally, the value of *level* is initially assigned based on the cost value of initial solution and the value will be decreased by a calculated decay rate during the search progress. Another advantage of GD is only one parameter is required which is the estimated quality of the desire solution. GD will try to explore the solution search region in which the solution quality is close to the estimated quality. With the effective performance of GD, it have been widely studied and applied in solving timetabling problems [6, 7, 10, 11, 19, 20, 24, 44, 58-61].

3.2. Nelder-Mead Simplex Search

Nelder and Mead [46] introduced the Nelder-Mead simplex search (NMSS) in addressing unconstrained optimization problems [33, 39]. In this approach, a simplex (which is made up by $N + 1$ vertices in N dimension) is used in finding solutions for the problem under investigation. A simplex can take various forms depending on the number of vertices used. For instance, a triangle simplex is constituted by two-dimensional vertices, while a tetrahedron simplex is constituted by three-dimensional vertices. There are four operations used in NMSS: reflection, expansion, contraction and shrinkage. Each of the operations is used to rescale the simplex with respect to the objective function. In addition, a centre point needs to be calculated for those operations. A Detailed explanation on NMSS can be seen in [46] and it has been successfully hybridized with other approaches in addressing university timetabling problems [58, 61] and inverse analysis problems [34]. In previous research [58, 61], the authors had proposed a hybrid method which constituted by NMSS, GD and fish swarm algorithm [40]. The aim of the used of NMSS is to determine the estimated quality of GD based on selected fish (solution) and two nearest fish during the search process. However, the equations proposed might produce negative values for the estimated quality of GD (note that there should not have negative values in the problems studied in this paper).

In this paper, the conventional NMSS algorithm is modified with the aim of assisting the GD approach in calculating estimated qualities so that GD explores toward varied search regions with respect to the penalty value of the solutions in the population. A triangle simplex is used and three operations are considered; external contraction (*EC*), reflection (*R*) and expansion (*E*) operators. The operations for those operators are discussed below:

1. Identify three vertices x_b, x_s and x_w from current empire (a group of solution), which correspond to the vertices that have best, second worst and worst objective cost values (f_b, f_s and f_w) [35], respectively. Then the center of simplex (C_{cent}) is calculated using Eq. (3). Three coefficients are initialized, which are reflection coefficient α, expansion coefficient γ and contraction coefficient β.
$$C_{cent} = \frac{f(x_b)+f(x_s)+f(x_w)}{3} \tag{2}$$

2. External Contraction (*EC*),
$$EC = [C_{cent} - (\beta \times C_{cent})] \tag{3}$$

3. Reflection (*R*), where $R < EC$
$$R = [EC - (\alpha \times C_{cent})] \tag{4}$$

4. Expansion (*E*), where $E < R$
$$E = [R - (\gamma \times C_{cent})] \tag{5}$$

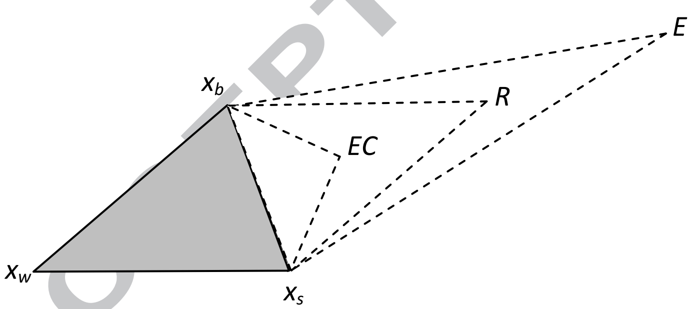


Fig. 1. NMSS operators in two dimensional simplex: external contraction (*EC*); reflection (*R*); expansion (*E*).

After the values of *EC*, *R* and *E* are calculated based on Eq. (3), Eq. (4) and Eq. (5), respectively, the values served as an estimated quality measure for GD in improving solutions with different qualities in the population. Note that the estimated quality is the quality of the desired solution that is declared by the user in basic GD. In ABC algorithm, there are a number of solutions in the population. A single estimated quality measure cannot be used to better exploit entire solutions in the population. Therefore, the NMSS is used to calculate a set of estimated qualities that can assist GD in exploiting solutions with different qualities in the population. The Triangle simplex approach used in this paper is demonstrated in Fig. 1. From Eq. (3 – 5), the value of *EC* is greater than *R*, and *R* greater than *E*. Note that *EC*, *R* and *E* also represent different search regions in the search space.

3.3. Assimilation Policy

The Imperialist competitive algorithm (ICA) is a recently invented approach introduced by Atashapz-Gargari and Lucas [17]. The approach is based on a socio-politically motivated strategy. Similar to other swarm based approaches, there are a number of agents in ICA that cooperate together in the provision of solutions. Agents (solutions) in ICA are colonies and imperialists (the words “solution” and “colony” will be used interchangeably throughout this paper). During the search process, the colonies form several ‘empires’ in the search region. The fittest colony in an empire will be selected as imperialist in the particular empire. Two movements are used in ICA: movement of colonies toward relevant imperialists (assimilation policy) and imperialist competition.

In this paper, an assimilation policy is implemented in ABC algorithm. During the movement of assimilation policy, all the colonies in an empire improve themselves by moving toward the relevant imperialist point. The reason for this is that the imperialist usually carries good information and it is worthwhile for the colonies to move toward and explore the promising search region near to the imperialist point. In addition, there are different groups of empire searching in the solution search region. This ensures ICA possesses good global exploration abilities capable of exploring multiple promising search regions simultaneously. However, the fact that each of the empires focus in exploring one solution search region can decrease the performance of the algorithm.

An example that demonstrates this weakness can be seen in Fig.2. This shows the solution search region with three empires. Each of the empires have their own imperialist and colonies. During the assimilation policy, all the colonies in each empire move toward their respective imperialist. This is the case that each of the colonies in the empires is restricted in exploring solution search regions near to imperialists and unable to escape from that search region (local optima). This can increase the choices of the search process becoming cyclic and therefore trapped in a particular search region, missing out other un-visited search regions [32].

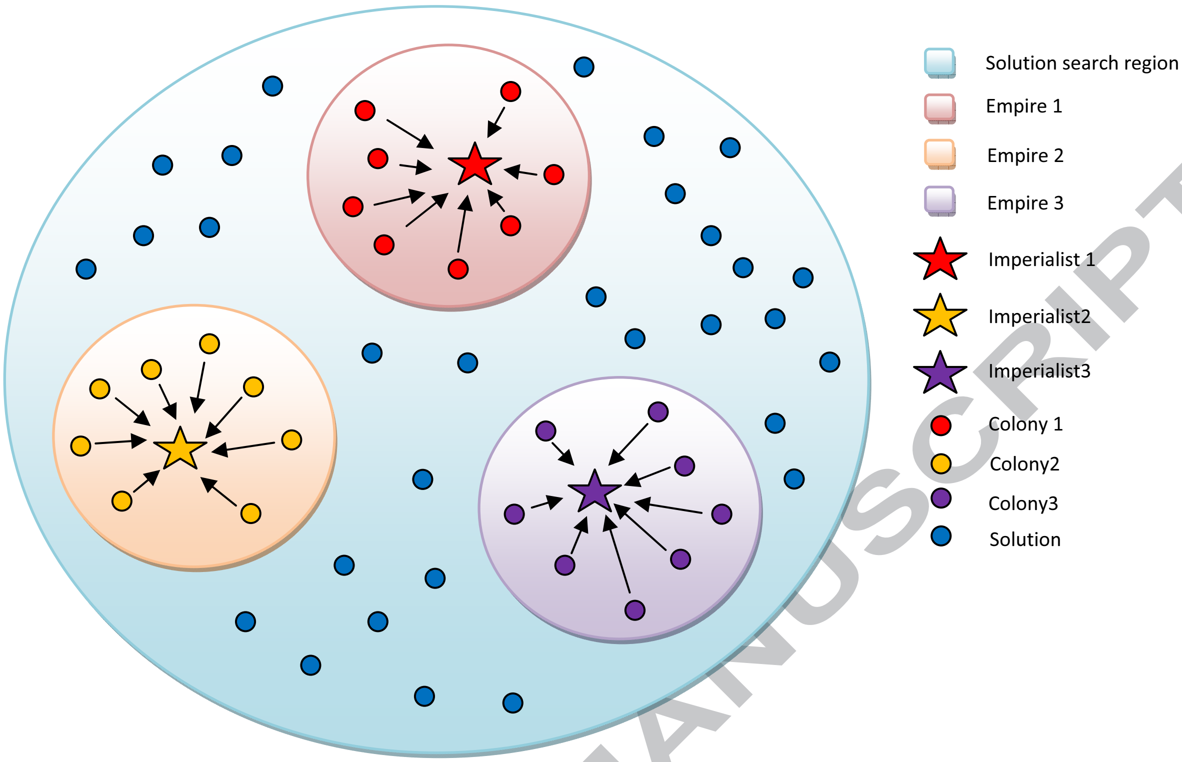


Fig. 2. Process of assimilation policy.

In addressing this, the conventional assimilation policy is modified by introducing a control probability within the assimilation policy. It is anticipated that the use of control probability is capable of controlling the degree of exploration for each colony on the search region near to the imperialists.

4. Imperialist Nelder-Mead great deluge artificial bee colony algorithm (INMGD-ABC)

In this section, the proposed approach (INMGD-ABC) is described. The goal of the proposed approach is to improve the global exploration power of the conventional ABC in identifying promising search regions. In addition, it also enhances the local exploitation ability of the conventional ABC so that it is able to fine-tune the search region of every solution. Furthermore, the roulette wheel selection scheme in basic ABC has been eliminated with the aim of overcoming the imbalance in exploitation.

The framework of the proposed approach is described Fig. 3. Generally, the proposed approach consists of two phases; initialization and improvement. There are three sub-phases in the improvement phase; employed bee, onlooker bee and scout bee. A detailed explanation of the proposed approach is demonstrated below.

```
1.  Imperialist Nelder-Mead Great Deluge Artificial Bee Colony Algorithm (INMGD-ABC)
2.
3.  Initialization:
4.  Initialize the population size, number of empire, empNumber;
5.  //population size = number of OnlookerBee = number of colony in all empires;
6.  Initialize the population;
7.  Calculate objective penalty cost value for each solution, f(sol);
8.  Set global best solution, sol_BS;
9.  Initialize number of iterations, ABCNumIteration;
10. Initialize value for parameter limit, limit;
11. Initialize BeeLimitCounter for each solution (sol) ← 0;
12. //parameter for ICA
13. Initialize assimilation policy probability, AP;
14. Number of colony in an empire, colInEMP ← population size/empNumber;
15. Assign colonies to empire based on colInEMP;
16. Identify imperialist of each empire (impSol_1 to impSol_e, where e = empNumber);
17.
18. Improvement:
19. For n = 1 to ABCNumIteration
20.    Explore solutions using Assimilation Policy (Fig. 4);  //Employed bee phase
21.
22.    Improve all solutions using NMGD (Fig. 7.);  //Onlooker bee phase
23.
24.    For i = 1 to EmployedBee do  //Scout bee phase
25.       If (BeeLimitCounter for sol_i ≥ limit)
26.          sol_s is abandoned and new solution sol_new, is generated randomly;
27.          Set limit counter for new solution sol_new to 0, BeeLimitCounter ← 0;
28.          sol_i ← sol_new;
29.       End If
30.    End For i
31. End For n
```

Fig. 3. Pseudo code for INMGD-ABC.

4.1. Initialization phase

In lines 4-8 (refer Fig. 3), 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 generating initial solutions in the population, a hybridization of graph coloring heuristics are used i.e. largest degree first, largest enrollment and saturation degree [16]. As can be seen from lines 9 and 10, two user defined parameters are initialized, which are the number of iteration (*ABCNumIteration*) for the proposed approach and the parameter *limit* (correspond to the numbers of iteration for a solution that stays inactive). In line 11, a parameter is defined to track the maximum number of iterations in which a solution stays inactive (*BeeLimitCounter*).

Lines 13 to 16 represent parameters required for ICA. In line 13 the probability for the execution of assimilation policy (*AP*) is initialized. Then the number of colonies in an empire (*colInEmp*) is calculated based on the population number and the number of empires (Line 14). The colonies are assigning to each empire based on the calculated *colInEmp* and the imperialist for each empire is identified in lines 15 and 16, respectively. Note that the number of colonies is equal to the number of solutions in the population.

4.2. Improvement phase

During the improvement phase (lines 19-31), the solution improvement process (exploration and exploitation on solution search region) is carried out by employed, onlooker and scout bees. Each of the process is outlined in the following section.

4.2.1. Employed bee phase

In this phase, the neighborhood search has been replaced by the assimilation policy search due to the fact that the exploitation ability of the neighborhood search in employed bee phase is relatively poor and slow. Hence, the assimilation policy from ICA is introduced with the aim of leading the search in exploring multiple promising search regions simultaneously. Line 20 in Fig. 3 represents the exploration on solutions using execution of the assimilation policy. Fig. 4 shows the details of the process for the assimilation policy. Firstly, the solution is checked to establish if it is a new solution generated in the scout bee phase (line 4). If so, a random number is drawn and compared against *AP* (lines 5-6). If the value of *r* is smaller than *AP*, the process of the assimilation policy will be invoked where the imperialist of the particular solution is identified and its information is inserted into the solution (lines 7-9).

```
1.  Assimilation Policy
2.
3.  For i = 1 to EmployedBee do
4.     If sol_i turn into scout bee in Scout bee phase
5.     Drawn a random number r, where (0 ≤ r ≤ 1);
6.        If (r ≤ AP)
7.           Determine imperialist of sol_i, impSol;
8.           Move sol_i(colony) towards it relevant imperialist by incorporate
9.           information of imperialist, impSol into sol_i;
10.       End If
11.    End If
12. End For i
```

Fig. 4. Pseudo code for assimilation policy.

To implement the concept of assimilation policy, haploid crossover introduced by [1, 2] is selected. The merit of haploid crossover is that the offspring generated contains information of best solution which enhances the search inefficiency (exploration). Therefore, the haploid crossover is used to incorporate information of an imperialist nature into the colony in an empire.

To carry out haploid crossover, there are two conditions needed to be satisfied in order to maintain the feasibility of the solutions:

1. There should not be a conflict between the moved events and scheduled events.
2. An event can only be moved if the corresponding time slot and room are free.

Fig. 5 demonstrates the process of crossover for examination timetabling problem in generating a new solution (Colony') from solution (Colony) by combining with the best solution (Imperialist). Firstly, two time slots are generated randomly, *t2* for Imperialist and *t3* for Colony. All the exams (i.e. *e5*, *e13*, *e15* and *e16*) in time slot *t3* (Colony) are moved to timeslot *t2* (Imperialist). During the movement of exams, any exams that clash or the exams already exist (i.e. *e15*) in the new time slot will not be moved. Lastly, a repair process is applied on the new solution (Colony') in order to maintain feasibility of the solution, where duplicate exams are removed (i.e. *e5* in time slot *t3* and *e13* and *e16* in time slot *t4*).

Fig. 6 presents the haploid crossover for course timetabling. The shaded time slots correspond to the selected time slot for the time slot exchange process. To begin this process, time slot *t1* and *t6* are chosen from Imperialist and Colony, respectively. Then the moveable courses (*c6* and *c23*) from *t6* are moved into *t1*. Coursesc*9* and *c12* cannot be moved as the room *r1* and *r3*are occupied by coursesc*21* and *c19*, respectively. For course *c28*, it cannot be inserted into *t1* because it clashes with course *c21*. Similar with haploid crossover for examination timetabling, a repair process is carried out to remove the duplicate courses (*c6* in time slot *t4* room *r2* and *c23* in time slot *t6* and room *r3*). This crossover process is controlled by a crossover point which indicates the number of timeslot for crossover process. For instance, the examples that illustrated at Figs. 6 and 7 are using crossover point of 1.

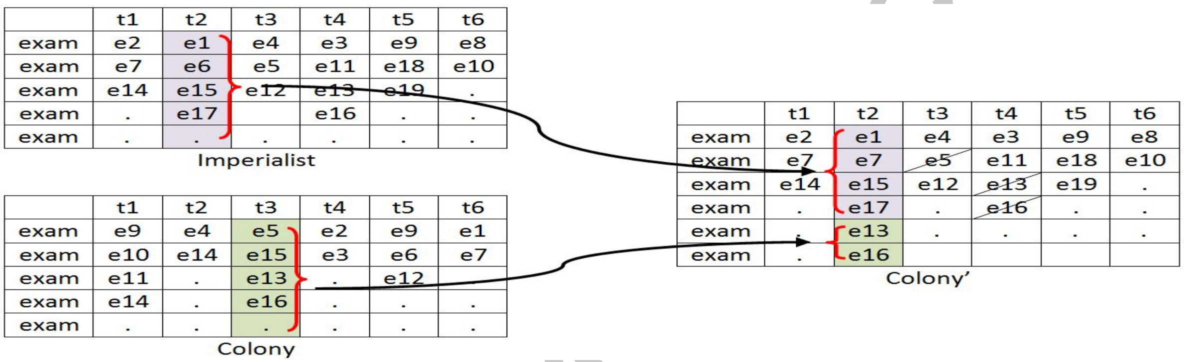


Fig. 5. Haploid crossover for examination timetabling.

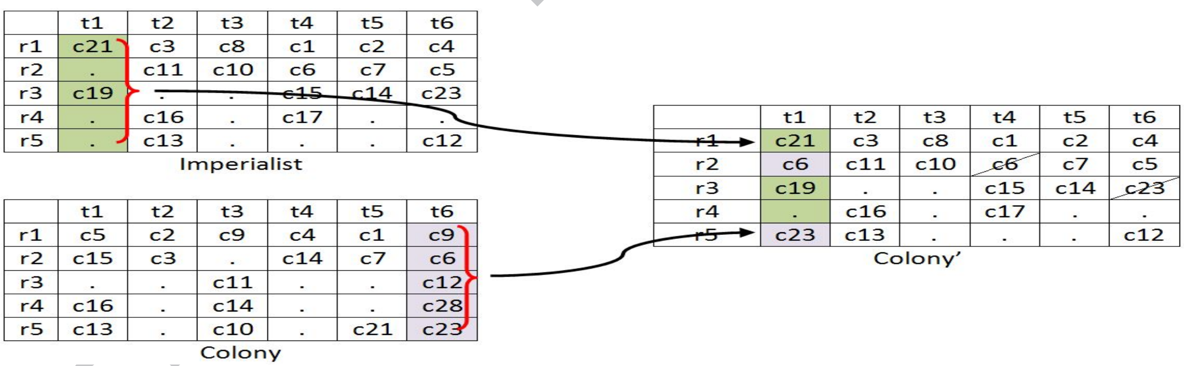


Fig. 6. Haploid crossover for course timetabling.

4.2.2. Onlooker bee phase

In the onlooker bee phase, entire solutions (bees in ABC = colonies in ICA) in the population are enhanced using NMGD and the roulette wheel selection scheme is eliminated. Detailed explanations of NMGD and the pseudo code for NMGD are presented in Section 4.3 and Fig. 7, respectively.

The Neighborhood moves used in NMGD to produce potential solutions are described as follows:

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

Nb 2: Selects two exams/courses randomly and swaps their time slot (and rooms for course timetabling). Feasibility of the solution is emphasized at the same time.

4.2.3. Scout bee phase

Lines 24-30 represent the solution abandon process carried out in the scout bee phase. Before a solution is abandoned, it's *BeeLimitCounter* (which tracks the number of iteration for which a solution stays inactive) will be checked against parameter *limit*. If the value of *BeeLimitCounter* is bigger than *limit*, the solution will be abandoned and replaced with a new randomly generated solution.

4.3. Nelder-mead great deluge algorithm (NMGD)

For conventional GD, only one parameter is required, i.e. the desired solution quality. In this research, NMSS is integrated within GD to calculate the estimated quality for GD (known as NMGD). Prior to the execution of GD, a set of estimated qualities will be calculated using NMSS. Then, GD will begin the exploitation process by selecting a suitable estimated quality for a solution. If the solution quality reaches the estimated quality before the stopping criteria is met, a new estimated quality will be selected and the exploitation process continues. Otherwise, GD will try to improve the quality of the solution towards the selected estimated quality. This exploitation process is repeated until all solutions are exploited and a set of estimated qualities will be recalculated in the next iteration. The merit of NMGD over basic GD is that NMGD manages to exploit multiple solutions with different estimated qualities. The pseudo code for NMGD is included in Fig. 7. There are several empires in the population and all of them will be improved by NMGD. The NMGD consists of two phases, initialization and improvement phases.

In the initialization phase, two parameters are initialized. These are: (i) estimated qualities (*EC*, *R* and *E*) which were calculated using NMSS (lines 5), and six more estimated qualities formed by dividing three extra regions between *EC*–*R* (known as *EC1*, *EC2* and *EC3*) and *R*–*E* (known as *R1*, *R2* and *R3*), (ii) iteration number for GD, *GD_Iteration*.

For the improvement phase, the quality of the entire solutions in an empire is enhanced based on the calculated estimated qualities. There are also two parameters needed to be assigned: (i) value of *level*, assigned based on the penalty cost value of the current solution, (ii) decay rates calculated based on the penalty cost value of the current solution and the estimated qualities (lines 11-14). After all the required parameters are initialized, the solution improvement process is started.

During execution, potential solution (*sol**) is generated using neighborhood structures (see Section 4.2.2) and the acceptance of it is based on two criteria. First, it is compared with the best solution ($bestSol_e$) found in the corresponding empire and second, compared against value of *level* (lines 20-30). If the first solution criteria are successful, the best solution is updated and the accepted solution will be compared against imperialist solution, $impSol_e$ (note that best solution is the current best solution in an empire and imperialist solution is the best solution found in an empire across the search process).

In contrast, if the first solution acceptance criteria failed, the second acceptance criteria will be considered. The tentative solution will be ignored if both acceptance criterions have failed followed by a decreasing of the value of *level* based on decay rate, used in generating a new solution in the next iteration. If the quality of solution is unable to be improved after the improvement process, the *BeeLimitCounter* will be increased by 1 (line 35).

```
NM-Great Deluge Algorithm (NMGD)

For e = 0 to empNumber do //loop all empires
   Initialization:
   Calculate estimated qualities;
   Set best solution in the empire in bestSol_e iteration for GD, GD_Iteration;

   Improvement:
   For i =1 to colInEMPin empire e do
      Set solution i as initial solution, sol_i;
      Set initial level, level ← f(sol_i);
      Calculate Nelder-Mead decay rates, β_EC, β_R, β_E, β_EC1, β_EC2, β_EC3, β_R1, β_R2, and
      β_R3 based on estimated qualities, (f(sol_i)- estimatedquality)/GD_Iteration;
      Decay rate β ← β_oc;

      For j = 1 to GD_Iteration do
         Determine estimated quality based on penalty cost of sol_i;
         β ← decay rate of estimated quality;
         Generate new solution sol* using Nb1 or Nb2;
         If f(sol*) < f(bestSol_e)
            sol_i ← sol*;
            bestSol_e ← sol*;
            If (f(impSol_e) ≤ f(sol*)) //update imperialist of the empire e
               impSol_e ← sol*;
            End If
         Else
            If ((f(sol*) ≤ level))
              Sol_i ← sol*;
            End If
         End If
         Level ← level - β;
     End for j

     If no improvement on quality of sol_i
        Add BeeLimitCounter for sol_i by 1;
     End If
   End For i
End For e
```

Fig. 7. Pseudo code for NMGD.

Table 3 demonstrates an example of a solution in selecting estimated quality during the execution of GD (Note that this example is taken from the experiment on car-s-91 Carter dataset). First, the estimated qualities *EC*, *R* and *E* are calculated based on x_b, x_s and x_w as shown in Fig. 8 (x_b, x_s and x_w correspond to best, second worst and worst solutions in an empire). Then, three more estimated qualities are formed between *EC* with *R* and *R* with *E*, respectively (Fig. 9). All these estimated qualities will be used by GD in improving the quality of the solution. The selection of the estimated quality depends on the penalty cost value of current solution where the estimated quality that is smaller and nearer to the penalty cost of current solution will be selected.

As referred to in Table 3, in the 1st GD iteration, the penalty value for sol_1 is 5.45828 (objective function of the problem in this paper), thus the estimated quality for GD to improve the solution (refer to Estimated Quality of *EC* in Table 3) is 6.41586 (i.e. value that is smaller or nearest to the current penalty value). By the 1000th iteration, the quality of sol_1 has been improved to 6.36542, therefore, *EC1*

(6.25723) will be selected as the estimated quality for GD. After whole solutions in an empire are enhanced by GD, the values of *EC*, *R* and *E* are recalculated based on best, second worst and worst solutions for the next empire.

Table 3
Estimated quality selection for NMGD.

GD_Iteration	Penalty Value	Estimated Quality								
		EC	*EC1*	*EC2*	*EC3*	*R*	*R1*	*R2*	*R3*	*E*
		6.41586	6.25723	6.09860	5.93996	5.78133	5.62269	5.46406	5.30543	5.14679
sol_1										
1	6.70369	√								
2	6.60123	√								
…										
1000	6.36542		√							
…										
2000	6.05441				√					

Fig. 8. Example of calculating estimated qualities for GD.

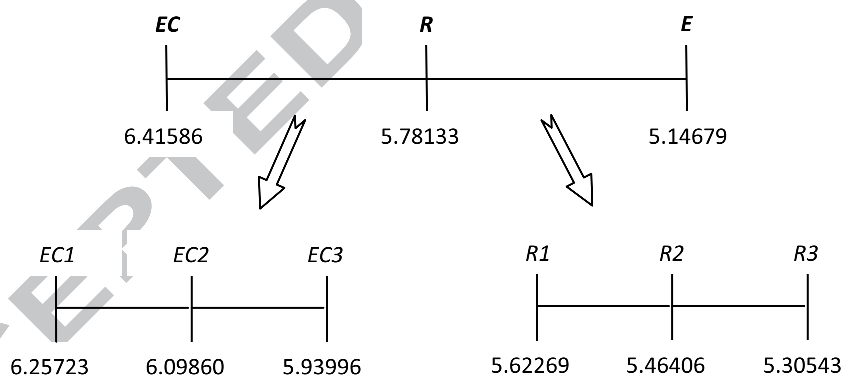


Fig. 9. Divides range of estimated qualities between *EC*-*R* and *R*-*E*.

5. Computational Results

In this section, a broad discussion on the computational experiments is presented. Generally, experiments conducted were tested on university examination and course timetabling problems that were described in Section 2. The experiments were conducted using a 2.2 GHz laptop and the proposed approach was coded using the C++ programming language. Parameters used in this study are shown in Table 4and were based on some preliminary experiments. It should be noted that these values should not be considered as optimal values but were the one that works fairly well across the two benchmarks. The experiments were conducted with 50 population sizes, 10000 and 2000 iterations for INMGD-ABC and NMGD, respectively. The parameter limit, on the other hand, equals to 4 (no improvement was observed after 4 attempts in the onlooker bee phase). From the preliminary experiments, it shows that the NMGD works well when the values of reflection coefficient α, expansion coefficient γ and contraction coefficient β are small [31]. This is the fact that a smaller gap in between the estimated qualities enable GD accepts worsening solution easily in fine-tuning the search region. In addition, it is found that the most important parameter is the probability for assimilation policy, *AP*. It plays an important role in determining the behavior of exploration and exploitation of the search process. The next section shall give some explanation regarding the effects of the probability for assimilation policy on the performance of the proposed approach.

Table 4
Parameter setting for INMGD-ABC.

No.	Parameter	Values
1	Iteration number for INMGD-ABC	10,000
2	Iteration number for NMGD	2,000
3	*Limit* (scout bee)	4
4	Assimilation probability, *AP*	0.75
5	Reflection coefficient	0.05
6	Expansion coefficient	0.05
7	Contraction coefficient	0.05
8	Haploid points	8
9	Number of empire	5

5.1. Effects of the assimilation policy probability

Some preliminary experiments have been carried out in order to identify the most suitable value for assimilation policy probability, *AP* for the proposed approach. Five different values of *AP* have been tested (*AP* = 0.0 to 1.0). The proposed approach was tested on five instances from Socha benchmark which are small 05, medium 03, medium 04, medium 05 and large instances. Table 5 illustrates the best cost penalty values with different *AP* values and it demonstrates that the proposed approach obtains the 4 best results (except instance medium 03) when *AP* value equal to 0.75. Best cost values are highlighted in bold font.

Table 5
Effect of assimilation policy probability (*AP*) on the performance of INMGD-ABC.

Dataset	*AP*=0.0	*AP*=0.25	*AP*=0.5	*AP*=0.75	*AP*=1.0
small 05	**0**	**0**	**0**	**0**	**0**
medium 03	188	171	149	119	**110**
medium 04	129	103	96	**70**	83
medium 05	155	115	98	**68**	72
large	788	605	554	**514**	530

From Table 5**,** it can be seen that *AP* plays a vital role in determining the exploration and exploitation abilities of the search process. When *AP* equals to 0.0, it can be observed that the proposed approach works like a random search approach without the use of assimilation policy (in employed bee phase) where it explores the problem search region randomly (scout bee phase). This makes the search process difficult to identify promising search regions. In this situation, the proposed approach focuses in exploration (randomly searches the search region) rather than exploitation (searches on best solution areas) which causes the slow convergence of the search process, resulting in poor solution qualities.

For (*AP*=0.25) to (*AP*=0.75), the proposed approach works like an iterated local search with a certain degree of random search. Note that the proposed approach utilizes the strength of the assimilation policy (with certain probability) by incorporating the imperialist solution information into the new solutions generated in the scout bee phase. This shows that the proposed approach is capable of exploring areas of promising search regions (by the employed bee) and other un-visited search regions (by the scout bee) in the search space in order to maintain the diversity of solutions within each group (empires). Here, the exploitation power is increased (from *AP*=0.25 to *AP*=0.75) and the convergence speed of the search process is improved, resulting in improvement of the qualities of solutions.

Lastly, when *AP* equals to 1.0, it can be observed that the proposed approach behaves similarly to local search (with the use of assimilation policy in employed bee phase) where the search process is focused in exploring different promising search regions that are near to the varied imperialist solutions (global best solutions of different empires). Nevertheless, this has resulted in all the solutions in each empire converging toward their imperialist solution search region and the diversity within the solutions of each empire decreasing. As such, the proposed approach might result in a cycle where the same search region is explored, easily getting trapped in local optima, where all the solutions in different empires converge toward their imperialist points.

In short, the higher the value of *AP* allows the proposed approach to explore search regions near the best solution point and also enhance the convergence speed. In university timetabling, the value of *AP* should not be too large, in order to avoid large exploitation so that the proposed approach will not behave like pure local search and get stuck in local optima. In addition, the value of *AP* should not be too small to avoid large exploration which will result in similar behaviour to pure random search. This might decrease the convergence speed of the search process. Hence, the value of *AP* is set to 0.75 in this paper which can achieve a balance between exploration on promising search regions and exploration of un-visited search regions.

5.2. Experiments on Socha course timetabling benchmark dataset

Experiments on the Socha course timetabling have been conducted. Experimental results obtained for thirty runs (each taking 15 minutes to 2 hours for basic ABC and 1 to 4 hours for proposed approach depending on the size of the instances) on both algorithms (basic ABC algorithm and proposed approach with different random seeds) and the comparison with best known results are summarized in Table 6 (best results are highlighted in bold font). From Table 6, it can be seen that the proposed approach outperforms the basic ABC in all instances. As compared with best known results, on the other hand, the proposed approach obtained the same penalty values on small instances and produced three new best results on medium 02, 03 and 05 instances.

It should be noted that there would be limited research value to be gained in comparing the computational times for those of which times were published. In this work, the computational time taken is considered acceptable for university timetabling, as in real world university timetabling, timetables are usually generated several months before the real timetable is used [21, 41]. In addition, rendering a full comparison on computational time is impossible due to several reasons [21]. Firstly, it is often the case that the computational time is overlooked and not reported within the literature [21]. Secondly, the termination criteria used are different across the approaches published in the literature. Thirdly, the implementation of algorithms on different specifications of machines will significantly affect the computational time used [21]. Hence, this has increased the difficulty in conducting a full comparison on the computational time due to there being no standard platform (same termination criteria and machines) to allow a fair comparison.

Table 6
Results comparison on basic ABC and INMGD-ABC with best known results for Socha benchmark dataset.

Dataset	ABC		INMGD-ABC		Best known results	
	Best	Avg.	Best	Avg.		
small 01	25	31.03	**0**	0	**0**	many
small 02	32	36.70	**0**	0	**0**	many
small 03	23	26.57	**0**	0	**0**	many
small 04	15	20.83	**0**	0	**0**	many
small 05	24	28.67	**0**	0	**0**	many
medium 01	393	433.67	52	70.63	**50**	Turabieh and Abdullah [57]
medium 02	436	470.50	**45**	79.53	54	Fong et al. [31]
medium 03	483	499.80	**96**	132.03	102	Turabieh and Abdullah [57]
medium 04	422	443.93	52	82.60	**32**	Turabieh and Abdullah [57]
medium 05	415	455.73	**56**	75.43	61	Turabieh and Abdullah [57]
large	908	991.67	461	503.03	**417**	Al-Betar and Khader[13]

Table 7 demonstrates the results comparison of the approach proposed in this study with state-of-the-art and best results are highlighted in bold font. The approaches selected in comparison are:

M1: Landa-Silva and Obit [37] – a non-linear great deluge algorithm.
M2: Turabieh and Abdullah [57] – a tabu based memetic approach.
M3: Abdullah et al. [6] – great deluge and tabu search.
M4: Abdullah and Turabieh [8] – hybridization of genetic algorithm and local search.
M5: Sabar et al. [51] – honey bee mating optimization algorithm.
M6: Burke et al. [23] – a graph based hyper-heuristic.
M7: Al-Betar and Khader[13]– a hybrid harmony search algorithm.
M8: Abdullah et al. [5]– variable neighbourhood search.
M9: Turabieh et al. [60] – a hybridization of electromagnetic-like mechanism with great deluge.
M10: Fong et al. [31] – a hybrid artificial bee colony algorithm.

Table 7
Results comparison with state-of-the-art.

Datasets	NMGD-ABC	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
small 01	**0**	3	**0**	**0**	2	**0**	6	**0**	**0**	**0**	**0**
small 02	**0**	4	**0**	**0**	4	**0**	7	**0**	**0**	**0**	**0**
small 03	**0**	6	**0**	**0**	2	**0**	3	**0**	**0**	**0**	**0**
small 04	**0**	6	**0**	**0**	**0**	**0**	3	**0**	**0**	**0**	**0**
small 05	**0**	**0**	**0**	**0**	4	**0**	4	**0**	**0**	**0**	**0**
medium 01	52	140	**50**	78	254	75	372	168	317	175	57
medium 02	**45**	130	70	92	258	88	419	160	313	197	54
medium 03	**96**	189	102	135	251	129	359	176	357	216	114
medium 04	52	112	**32**	75	321	74	348	144	245	149	74
medium 05	**56**	141	61	68	276	64	171	71	292	190	64
large	461	876	653	556	1027	523	1068	**417**	-	912	502

By referring to Table7, it can be seen that the proposed approach has generated good quality solutions for all instances and obtained optimal solutions for all five small instances (best results are highlighted in bold font). In addition, the proposed approach also outperformed the state-of-the-art in medium 02, 03 and 05 instances and obtains second best result on medium 01, 04 and large instances. Furthermore, the proposed approach is outperformed M10 (author's pervious work) in all instances.

Fig. 10 represents the convergence graph for instance medium 05. The *y-axis* represents the penalty cost value while *x-axis* represents the number of iteration. By analyzing Fig. 10, it can be seen that there is a large improvement in term of solution quality (curve with steep slope) at the beginning of the search. Nevertheless, the increment of cost value decreases gradually as iteration number increases until no improvement at the end of the search process.

Fig. 11 represents the convergence graph for INMGD-ABC and basic ABC for large instance. From the observation on Fig. 11, the slope of the curve for INMGD-ABC is relatively steep as compared with the basic ABC which illustrates that there is great improvement in the quality of the solutions at the beginning of the search process. In addition, INMGD-ABC converges faster as compared with basic ABC and the qualities of solution found by INMGD-ABC are better than basic ABC across the search process.

Box plots of the solution penalty cost for all instances of the Socha dataset can be seen in Fig. 12. For small instances, the gaps between best, average and worst cost values are zero. In addition, there is a relatively small gap between the best, average and worst cost values for medium instances. For large instances, on the other hand, there is slightly bigger gap between the best and worst penalty value (different in penalty cost more than 30). However, it is acceptable where the worst cost value (545) is ranked in third place as compared with the state-of-the-art results (refer to Table 8).

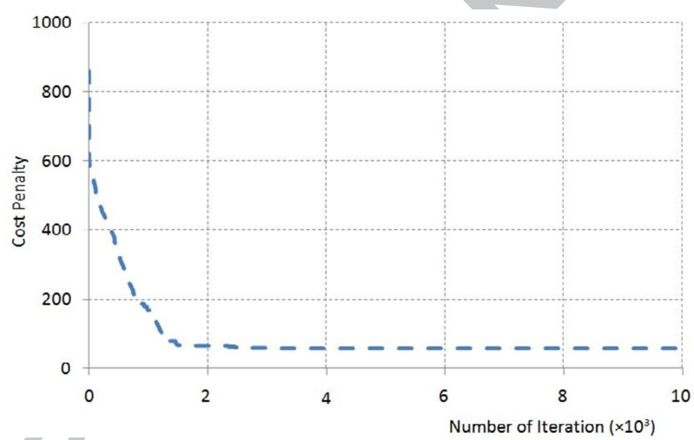


Fig. 10. Convergence graph for instance medium 05.

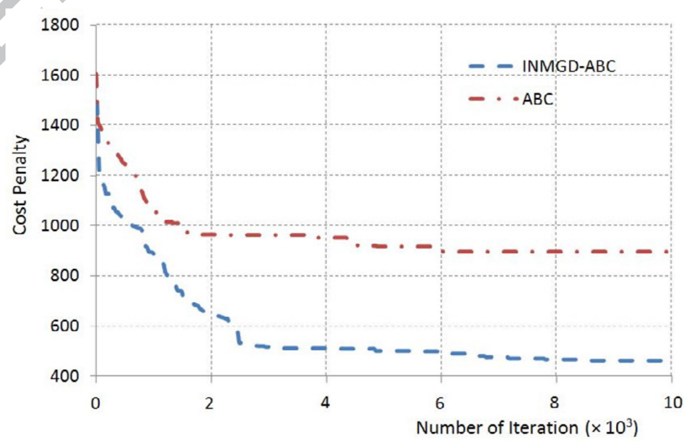


Fig. 11. Convergence graph for INMGD-ABC and basic ABC for large instance.

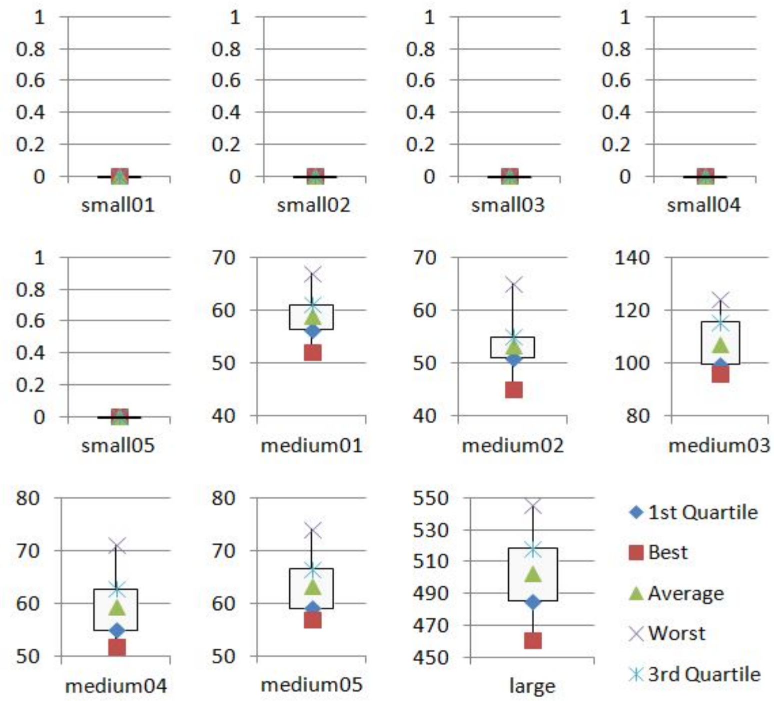


Fig. 12. Box plots for Socha course datasets.

Table 8 shows the penalty value for violations of each soft constraint based on the best solution generated by the proposed approach in relation to the Socha datasets. Soft constraint 1 states that a student should not be requested to attend more than two consecutive courses; soft constraint 2 states that students should not be required to attend a single class in a day; and soft constraint 3 states that a student should not need to attend a class that is scheduled in last time slot of a day.

Table 8
Soft constraints violation based on best solution obtained for Socha dataset.

Dataset	Penalty value for soft constraint 1	Penalty value for soft constraint 2	Penalty value for soft constraint 3	Total Penalty value
small 01	0	0	0	0
small 02	0	0	0	0
small 03	0	0	0	0
small 04	0	0	0	0
small 05	0	0	0	0
medium 01	49	3	0	52
medium 02	45	0	0	45
medium 03	96	0	0	96
medium 04	52	0	0	52
medium 05	55	1	0	56
large	294	3	164	461

From Table 8, it can be seen that the proposed approach is able to produce solutions that satisfies all the soft constraints for small instances and satisfies soft constraint 2 and 3 well for medium instances. It is believed that there is a close relationship among the soft constraints. Note that to satisfy soft constraint 2, there must be more than one class in a day. To satisfy soft constraint 3, the courses should not schedule in the last time slot of a day. As a result, this will increase the violation on soft constraint 1 where more classes will be scheduled in a day to avoid having a single class on a day and avoiding having courses on the last time slot on a day. This also indirectly reduces the solution points in the search region.

For large instance, the satisfaction of soft constraints 1 and 3 is poorer as compared with small and medium instances. It is believed that this is related to the nature of the dataset where there is a higher number of students which need to be considered and the maximum number of events can be enrolled per student is also higher as compared with small and medium instances.

5.3. Experiments on Carter examination un-capacitated timetabling benchmark dataset

In addition to applying the proposed approach to solving university course timetabling, its performance has also been tested on the university examination timetabling problem instances introduced by Carter [28] (see Section 2.2). The experiments were carried out by running 30 times (each taking 1 to 3 hours and 2 to 10 hours for basic ABC and proposed approach, respectively, depending on the size of the instances) for each instance with the parameters used in the experiment shown in Table 4. Two experimental comparisons have been carried out i.e. comparison with best known results (Table 9) and comparison with results produced by different approaches in the literature (Table 10). Table 9 shows the resultant comparison between the basic ABC, proposed approach and also best known results published in the literature (best results are highlighted in bold font). By observing Table 9, it is clear that the proposed approach outperforms basic ABC in all instances and produces two new best results (and equaling Sabar et al. [51] in uta-s-92 instance).

Table 9
Results comparison on basic ABC and INMGD-ABC with best known results for Carter un-capacitated examination benchmark dataset.

Datasets	ABC		INMGD-ABC		Best known results	
	Best	Avg.	Best	Avg.		
car-f-92	4.80	4.82	**3.77**	4.10	3.89	Fong et al. [31]
car-s-91	5.60	5.76	4.71	4.82	**4.50**	Yang and Petrovic[63]
ear-f-83	37.47	38.88	33.15	34.02	**29.30**	Caramia et al. [25]
hec-s-92	11.58	11.90	10.38	10.50	**9.20**	Caramia et al. [25]
kfu-s-93	15.25	15.54	13.69	13.78	**13.00**	Burke et al. [21]
lse-f-91	12.48	12.87	10.25	10.35	**9.60**	Caramia et al. [25]
sta-f-83	158.27	159.01	157.03	157.51	**156.90**	Burke et al. [21]
tre-s-92	8.85	8.97	**7.84**	7.98	7.86	Fong et al. [31]
uta-s-92	3.75	3.89	**3.10**	3.27	**3.10**	Sabar et al. [51]
ute-s-92	27.65	28.41	25.32	25.81	**24.40**	Caramia et al. [25]
yor-f-83	40.21	40.55	36.06	36.57	**34.90**	Burke et al. [21]

In addition, the results generated by the proposed approach are compared against results produced by other approaches in the literature. The comparison can be seen at Table 10 and best results are highlighted in bold font. Approaches selected for comparison are:

H1: Sabar et al. [51] – honey bee mating optimization algorithm.
H2: Turabieh and Abdullah [59] – hybridization of electromagnetic-like mechanism with great deluge.
H3: Burke et al. [21] – hybridization of variable neighborhood search with genetic algorithm.
H4: Abdullah et al. [3] – hybridization of large neighborhood search with local search.
H5: Caramia et al. [25] – novel iterated local search algorithm.
H6: Pillay and Banzhaf [47] – informed genetic algorithm.

H7: Burke and Newall[24] – adaptive ordering initialization with great deluge algorithm.
H8: Qu and Burke[48] – graph-based hyper-heuristic.
H9: Merlot et al. [45] – hybridization of constraint programming, simulated annealing and hill climbing.
H10: Fong et al. – [31] a hybrid artificial bee colony algorithm.

From Table 10, it can be observed that the proposed approach is capable of producing good quality solutions and generating three new best results on car-f-92, tre-s-92 and uta-s-92 (equaling (H1) and (H10) on uta-s-92 instance) instances. Besides that, the results generated are better than H10 (author's pervious work). In addition, the proposed approach obtains second best results for sta-f-83, car-s-91 and yor-f-83 instances and third best for the ear-f-83 instance.

Table 10
Results comparison with state-of-the-art.

Datasets	INMGD-ABC	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10
car-f-92	**3.77**	3.90	4.10	3.90	4.40	6.00	4.20	4.10	4.16	4.30	3.89
car-s-91	4.71	4.79	4.80	**4.60**	5.20	6.66	4.90	4.65	5.16	5.10	4.79
ear-f-83	33.15	34.69	34.92	32.80	34.90	**29.30**	35.90	37.05	35.86	35.10	33.43
hec-s-92	10.38	10.66	10.73	10.00	10.30	**9.2**	11.50	11.54	11.94	10.60	10.49
kfu-s-93	13.69	**13.00**	**13.00**	**13.00**	13.50	13.80	14.40	13.90	14.79	13.50	13.72
lse-f-91	10.25	10.00	10.01	10.00	10.20	**9.60**	10.90	10.82	11.15	10.5	10.29
sta-f-83	157.03	157.04	158.26	**156.90**	159.20	158.20	157.80	168.73	159.00	157.30	157.07
tre-s-92	**7.84**	7.87	7.88	7.90	8.40	9.40	8.40	8.35	8.60	8.40	7.86
uta-s-92	**3.10**	**3.10**	3.20	3.20	3.60	3.50	3.40	3.20	3.59	3.50	**3.10**
ute-s-92	25.32	25.94	27.00	24.80	26.00	**24.40**	27.20	25.83	28.30	25.10	25.33
yor-f-83	36.06	36.15	36.22	**34.90**	36.20	36.20	39.30	37.28	41.81	37.40	36.12

Fig. 13 shows the convergence graph for the proposed approach in improving the quality of solutions for car-f-92 instance. It can be seen that the trend of the graph is almost the same with Fig. 10. Hence, it can be argued that the proposed approach is capable of performing in a similar manner to different problems with varying complexity.

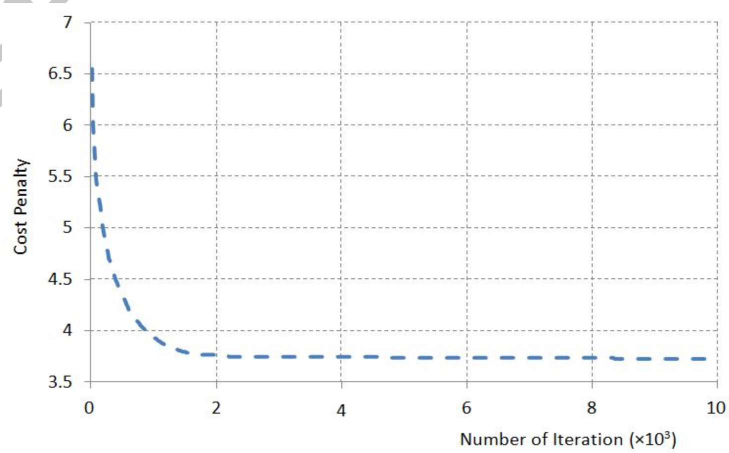


Fig. 13. Convergence graph for car-f-92.

Fig. 14 shows the box plots that demonstrate the distribution of the solutions quality (best, first quartile, average, third quartile and worst) for Carter un-capacitated examination dataset. It is clearly shown that the gap between the best, average and worst cost penalty values is relatively small which illustrate this is a stable approach, performing consistently across all instances.

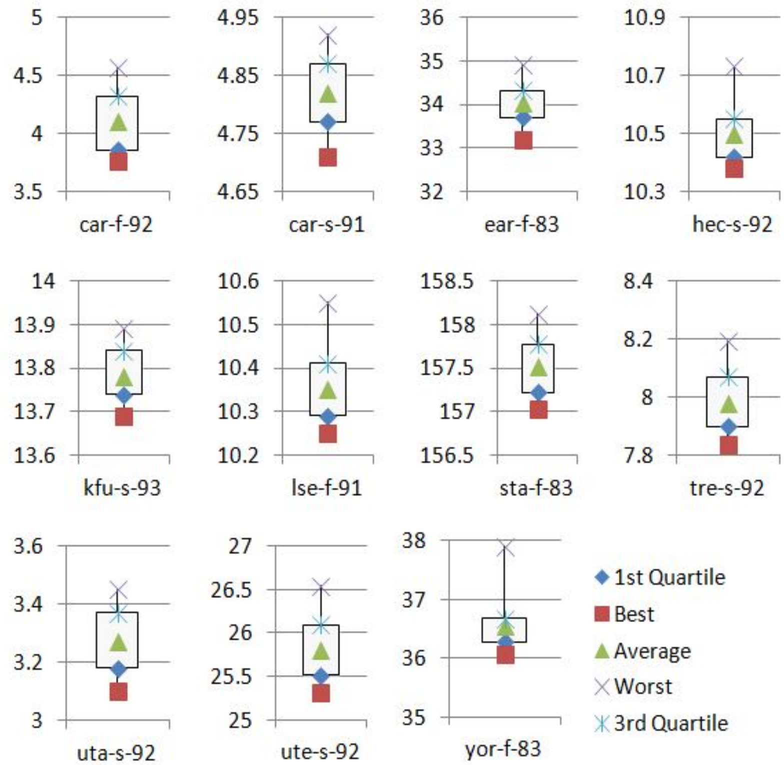


Fig. 14. Box plots for Carter un-capacitated dataset.

Tables 11 and 12 illustrate the statistical analysis between the ABC algorithm and the proposed approach in solving Socha course timetabling and Carter examination timetabling problems, respectively. Two *t*-tests have been performed to demonstrate that the proposed approach performs better than the basic ABC algorithm in solving both timetabling problems at the level of confidence of 0.05. The null hypothesis (H_0) has been defined as there is no difference between the performances of both approaches, whereas, the alternative hypothesis (H_1) defined as the performance of INMGD-ABC is better than the basic ABC algorithm. The goal of the testing is to bolster the H_1. By observing Tables 11 and 12, it can be seen that the *p*-value for all instances are smaller than 0.05 which indicates that there is a significant difference for all instances in both benchmarks. Hence, with the significant evidence to support the claim on H_1, the H_0 is rejected and it can be concluded that the performance of INMGD-ABC outperforms basic ABC algorithm in addressing both timetabling problems.

From the analysis above, it is clear that the proposed approach is capable of generating high quality solutions for both university course and examination timetabling problems. Indeed, it does produce some best results for both problems. It is believed that the use of the assimilation concept manages to improve the exploration ability of the proposed approach where multi promising search region are explored simultaneously. In addition, the control probability (*AP*) is capable of controlling the degree of exploration of promising search regions. This can prevent the search process focusing only in exploring promising search region which might trap the search in a particular search region (or cycling within the same search region). Furthermore, inactive solutions are abandoned and new solutions are generated to drive the search process away from potential local optima. Also, strength is the use of the exploitation ability local search (NMGD) for exploiting entire solutions in the population to optimal solutions.

Table 11
Statistical analysis for basic ABC and INMGD-ABC on Socha course timetabling problem.

Dataset	ABC		INMGD-ABC		*t*-test
	Best	Avg.	Best	Avg.	*p*-value
small 01	25	31.03	0	0	7.8873E-28
small 02	32	36.70	0	0	3.4334E-30
small 03	23	26.57	0	0	1.3339E-28
small 04	15	20.83	0	0	6.1954E-25
small 05	24	28.67	0	0	4.064E-27
medium 01	393	433.67	52	70.63	2.8542E-46
medium 02	436	470.50	45	79.53	4.4790E-49
medium 03	483	499.80	96	132.03	2.7277E-80
medium 04	422	443.93	52	82.60	7.5627E-63
medium 05	415	455.73	56	75.43	4.6689E-44
large	908	991.67	461	503.03	3.8242E-56

Table 12
Statistical analysis for basic ABC and INMGD-ABC on Carter examination timetabling problem.

Datasets	ABC		INMGD-ABC		t-test
	Best	Avg.	Best	Avg.	*p*-value
car-f-92	4.80	4.82	3.77	4.10	5.0601E-24
car-s-91	5.60	5.76	4.71	4.82	1.3291E-27
ear-f-83	37.47	38.88	33.15	34.02	1.7051E-40
hec-s-92	11.58	11.90	10.38	10.50	2.0016E-29
kfu-s-93	15.25	15.54	13.69	13.78	5.6964E-27
lse-f-91	12.48	12.87	10.25	10.35	2.9117E-38
sta-f-83	158.27	159.01	157.03	157.51	6.5564E-23
tre-s-92	8.85	8.97	7.84	7.98	1.1294E-40
uta-s-92	3.75	3.89	3.10	3.27	1.0151E-31
ute-s-92	27.65	28.41	25.32	25.81	8.1766E-36
yor-f-83	40.21	40.55	36.06	36.57	2.1943E-42

6. Conclusion

In this paper, an INMGD-ABC has been proposed to tackle the university exam and course timetabling problems. With the use of the assimilation policy concept in employed bee phase the search process is guided and explores promising search regions with certain probability controlled by the assimilation policy probability. NMGD is then applied to locally explore and fine-tune promising solutions within the onlooker bee phase. In short, the proposed method is capable of improving both exploration and exploitation abilities and the convergence speed of the basic ABC.

The basic ABC and proposed method (INMGD-ABC) were applied on two benchmark problems; Carter un-capacitated examination timetabling datasets and Socha course timetabling datasets. Datasets from two different areas of educational timetabling are chosen to show the generality and robustness of the proposed approach where it manages to solve two problems with different complexity and different solution landscapes. Experimental results demonstrate that INMGD-ABC outperforms basic ABC in producing good quality solutions. This is because of INMGD-ABC is able to globally explore and locally exploit the solution search region using the concept of assimilation policy from ICA and NMGD, respectively. Furthermore, statistical analysis by using *t*-test has been conducted and the results show that the performance of the proposed approach is significantly better than the basic ABC algorithm. Lastly, several very competitive literature-beating results have been obtained and prove the worth of continuing further work on this approach. In order to further establish proof of generality in the proposed approach, it will be further applied on areas of timetabling such as the benchmarks introduced in ITC2007.

References

[1] H.A. Abbass, MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach, in: Proceedings of the 2001 Congress on Evolutionary Computation, IEEE, Seoul, 2001, pp. 207-214.

[2] H.A. Abbass, A monogenous MBO approach to satisfiability, in: Proceeding of the International Conference on Computational Intelligence for Modeling, Control and Automation, CIMCA'2001, Las Vegas, NV, USA., 2001.

[3] S. Abdullah, S. Ahmadi, E.K. Burke, M. Dror, Investigating Ahuja–Orlin’s large neighbourhood search approach for examination timetabling, OR Spectrum, 29 (2007) 351-372.

[4] S. Abdullah, S. Ahmadi, E.K. Burke, M. Dror, B. McCollum, A tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem, Journal of the Operational Research Society, 58 (2007) 1494-1502.

[5] S. Abdullah, E.K. Burke, B. McCollum, An investigation of variable neighbourhood search for university course timetabling, in: The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications, 2005, pp. 413-427.

[6] S. Abdullah, K. Shaker, B. McCollum, P. McMullan, Construction of course timetables based on great deluge and tabu search, in: The 8th Metaheuristics International Conference, Citeseer, 2010, pp. 13-16.

[7] S. Abdullah, K. Shaker, B. McCollum, P. McMullan, Dual sequence simulated annealing with round-robin approach for university course timetabling in: P. Cowling, P. Merz (Eds.) Evolutionary Computation in Combinatorial Optimization 10th European Conference, Proceedings EvoCOP 2010, Lecture Notes in Computer Science, Springer, Istanbul, Turkey, 2010, pp. 1-10.

[8] S. Abdullah, H. Turabieh, Generating university course timetable using genetic algorithms and local search, in: Proceedings of the 2008 3rd International Conference on Convergence and Hybrid Information Technology, IEEE Computer Society, 2008, pp. 254-260.

[9] S. Abdullah, H. Turabieh, On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems, Information Sciences, 191 (2012) 146-168.

[10] S. Abdullah, H. Turabieh, B. McCollum, A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems, in: M. Blesa, C. Blum, L. Gaspero, A. Roli, M. Sampels, A. Schaerf (Eds.) Hybrid Metaheuristics 6th International Workshop, Proceedings HM 2009, Lecture Notes in Computer Science, Springer, Udine, Italy, 2009, pp. 60-72.

[11] S. Abdullah, H. Turabieh, B. McCollum, P. McMullan, A hybrid metaheuristic approach to the university course timetabling problem, Journal of Heuristics, 18 (2012) 1-23.

[12] M.A. Al-Betar, A.T. Khader, A hybrid harmony search for university course timetabling, in: Proceedings of the 4nd Multidisciplinary Conference on Scheduling: Theory and Applications (MISTA 2009), Dublin, Ireland, August, 2009, pp. 10-12.

[13] M.A. Al-Betar, A.T. Khader, A harmony search algorithm for university course timetabling, Ann Oper Res, (2012) 1-29.

[14] M. Alzaqebah, S. Abdullah, Comparison on the selection strategies in the artificial bee colony algorithm for examination timetabling problems, International Journal of Soft Computing and Engineering, 1 (2011).
[15] M. Alzaqebah, S. Abdullah, Hybrid artificial bee colony search algorithm based on disruptive selection for examination timetabling problems, in: W. Wang, X. Zhu, D.-Z. Du (Eds.) Combinatorial Optimization and Application 5th International Conference, Proceedings COCOA 2001, Lecture Notes in Computer Science, Springer, Zhangjiajie, China, 2011, pp. 31-45.
[16] H. Asmuni, E.K. Burke, J.M. Garibaldi, B. McCollum, A.J. Parkes, An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables, Computer & Operations Resarch, 36 (2009) 981-1001.
[17] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: IEEE Congress on Evolutionary Computation, IEEE, Singapore, 2007, pp. 4661-4667.
[18] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv., 35 (2003) 268-308.
[19] E.K. Burke, Y. Bykov, J. Newall, S. Petrovic, A time-predefined approach to course timetabling, Yugoslav Journal of Operations Research, 13 (2003).
[20] E.K. Burke, Y. Bykov, J. Newall, S. Petrovic, A time-predefined local search approach to exam timetabling problems, Yugoslav Journal of Operations Research, 13 (2003) 139-151.
[21] E.K. Burke, A.J. Eckersley, B. McCollum, S. Petrovic, R. Qu, Hybrid variable neighbourhood approaches to university exam timetabling, European Journal of Operational Research 206, (2010) 46-53.
[22] E.K. Burke, K. Jackson, J. Kingston, R. Weare, Automated university timetabling: The state of the art, The Computer Journal, 40 (1997) 565-571.
[23] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyper-heuristic for educational timetabling problems, European Journal of Operational Research, 176 (2007) 177-192.
[24] E.K. Burke, J.P. Newall, Enhancing timetable solutions with local Search methods, in: E. Burke, P. Causmaecker (Eds.) Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer, Gent, Belgium, 2003, pp. 195-206.
[25] M. Caramia, P. Dell'Olmo, G. Italiano, New algorithms for examination timetabling, in: S. Näher, D. Wagner (Eds.) Algorithm Engineering 4th International Workshop, Proceedings WAE 2001, Lecture Note in Computer Science, Springer, Saarbrücken, Germany, 2001, pp. 230-241.
[26] M.W. Carter, G. Laporte, Recent developments in practical examination timetabling, in: E.K. Burke, P. Ross (Eds.) Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer, Edinburgh, U.K., 1996, pp. 1-21.
[27] M.W. Carter, G. Laporte, Recent developments in practical course timetabling, in: E.K. Burke, M.W. Carter (Eds.) Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer, Toronto, Canada, 1998, pp. 3-19.
[28] M.W. Carter, G. Laporte, S.Y. Lee, Examination timetabling: algorithmic strategies and applications, The Journal of the Operational Research Society, 47 (1996) 373-383.
[29] G. Dueck, New optimization heuristics: The great deluge algorithm and the record-to-record travel, Journal of Computational Physics, 104 (1993) 86-92.
[30] I. Durgun, A.R. Yildiz, Structural design optimization of vehicle components using cuckoo search algorithm, MP Materials Testing, 54 (2012) 185.
[31] C.W. Fong, H. Asmuni, B. McCollum, A Hybrid Swarm Based Approach to University Timetabling, IEEE Transactions on Evolutionary Computation (Accepted, to be pusblished), (2014).
[32] W. Gao, S. Liu, A modified artificial bee colony algorithm, Computer & Operation Research, 39 (2012) 687-697.

[33] J. J. E. Dennis, R.B. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations (Classics in Applied Mathematics, 16), Soc for Industrial & Applied Math, 1996.
[34] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers & Structures, 87 (2009) 861-870.
[35] F. Kang, J. Li, Q. Xu, Structural inverse analysis by hybrid simplex artificial bee colony algorithms, Computers & Structures, 87 (2009) 861-870.
[36] D. Karaboga, An idea based on honey bee swarm for numerical optimization, in, Erciyes University, 2005.
[37] D. Landa-Silva, J. Obit, Evolutionary non-linear great deluge for university course timetabling, in: E. Corchado, X. Wu, E. Oja, Á. Herrero, B. Baruque (Eds.) Hybrid Artificial Intelligence Systems, 4th International Conference, Proceedings HAIS 2009, Lecture Notes in Computer Science, Springer, Salamanca, Spain, 2009, pp. 269-276.
[38] R. Lewis, A survey of metaheuristic-based techniques for university timetabling problems, OR Spectrum, 30 (2008) 167-190.
[39] D.-H. Li, M. Fukushima, On the global convergence of the BFGS method for nonconvex unconstrained optimization problems, SIAM Journal on Optimization, 11 (2001) 1054-1064.
[40] X.L. Li, Z.J. Shao, J.X. Qian, An optimizing method based on autonomous animate: fish swarm algorithm, System Engineering Theory and Practice, 11 (2002) 32-38.
[41] B. McCollum, A perspective on bridging the gap between theory and practice in university timetabling, in: E. Burke, H. Rudová (Eds.) Practice and Theory of Automated Timetabling VI, Springer Berlin Heidelberg, 2007, pp. 3-23.
[42] B. McCollum, P. McMullan, A.J. Parkes, E.K. Burke, R. Qu, A new model for automated examination timetabling, Ann Oper Res, 194 (2012) 291-315.
[43] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A.J. Parkes, L. Di Gaspero, R. Qu, E.K. Burke, Setting the research agenda in automated timetabling: The second international timetabling competition, INFORMS Journal on Computing, 22 (2010) 120-130.
[44] P. McMullan, An extended implementation of the great deluge algorithm for course timetabling, in: Y. Shi, G. Albada, J. Dongarra, P.A. Sloot (Eds.) Computer Science 7th International Conference Part I, Proceedings ICCS 2007, Lecture Notes in Computer Science, Springer, 2007, pp. Pages: 538 - 545
[45] L.G. Merlot, N. Boland, B. Hughes, P. Stuckey, A hybrid algorithm for the examination timetabling problem, in: E. Burke, P. Causmaecker (Eds.) Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer, Gent, Belgium, 2003, pp. 207-231.
[46] J.A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal, 7 (1965) 308-313.
[47] N. Pillay, W. Banzhaf, An informed genetic algorithm for the examination timetabling problem, Applied Soft Computing, 10 (2010) 457-467.
[48] R. Qu, E.K. Burke, Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems, Journal of the Operational Research Society, 60 (2008) 1273-1285.
[49] R. Qu, E.K. Burke, B. McCollum, Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems, European Journal of Operational Research, 198 (2009) 392-404.
[50] R. Qu, E.K. Burke, B. McCollum, L.T. Merlot, S.Y. Lee, A survey of search methodologies and automated system development for examination timetabling, Journal of Scheduling, 12 (2009) 55-89.
[51] N.R. Sabar, M. Ayob, G. Kendall, R. Qu, A honey-bee mating optimization algorithm for educational timetabling problems, European Journal of Operational Research, 216 (2012) 533-543.
[52] A. Schaerf, A survey of automated timetabling, Artificial Intelligence Review, 13 (1999) 87-127.

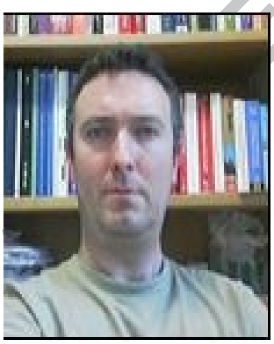
[53] K. Socha, J. Knowles, M. Sampels, A MAX-MIN ant system for the university course timetabling problem, in: M. Dorigo, G. Caro, M. Sampels (Eds.) Ant Algorithm 3rd International Workshop, ANTS 2002, Lecture Notes in Computer Science, Springer, Brussels, Belgium, 2002, pp. 1-13.
[54] M.F. Tasgetiren, Q.-K. Pan, P.N. Suganthan, A.H.L. Chen, A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops, Information Sciences, 181 (2011) 3459-3475.
[55] J. Thompson, K. Dowsland, Variants of simulated annealing for the examination timetabling problem, Ann Oper Res, 63 (1996) 105-128.
[56] J. Thompson, K. Dowsland, A robust simulated annealing based examination timetabling system, Computers & Operations Research, 25 (1998) 637-648.
[57] H. Turabieh, S. Abdullah, Incorporating tabu search into memetic approach for enrolment-based course timetabling problems, in: 2nd Conference on Data Mining and Optimization, (DMO) 2009, pp. 115-119.
[58] H. Turabieh, S. Abdullah, A hybrid fish swarm optimisation algorithm for solving examination timetabling problems, in: C.C. Coello (Ed.) Learning and Intelligent Optimization, 5th International Conference, Proceedings LION 5, Lecture Notes in Computer Science, Springer, Rome, Italy, 2011, pp. 539-551.
[59] H. Turabieh, S. Abdullah, An integrated hybrid approach to the examination timetabling problem, Omega - The International Journal of Management Science, 39 (2011) 598-607.
[60] H. Turabieh, S. Abdullah, B. McCollum, Electromagnetism-like mechanism with force decay rate great deluge for the course timetabling problem, in: P. Wen, Y. Li, L. Polkowski, Y. Yao, S. Tsumoto, G. Wang (Eds.) Rough Sets and Knowledge Technology, 4th International Conference, Proceedings RSKT 2009, Lecture Notes in Computer Science, Springer, Gold Coast, Australia, 2009, pp. 497-504.
[61] H. Turabieh, S. Abdullah, B. McCollum, P. McMullan, Fish swarm intelligent algorithm for the course timetabling problem, in: J. Yu, S. Greco, P. Lingras, G. Wang, A. Skowron (Eds.) Rough Set and Knowledge Technology, 5th International Conference, Proceedings RSKT 2010, Lecture Notes in Computer Science, Springer, Beijing, China, 2010, pp. 588-595.
[62] G. White, B.S. Xie, S. Zonjic, Using tabu search with longer-term memory and relaxation to create examination timetables, European Journal of Operational Research, 153 (2004) 80-91.
[63] Y. Yang, S. Petrovic, A novel similarity measure for heuristic selection in examination timetabling, in: E. Burke, M. Trick (Eds.) Selected Papers from 5th international conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, Springer, Pittsburgh, PA, USA, 2005, pp. 247-269.
[64] A.R. Yildiz, An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry, Journal of Materials Processing Technology, 209 (2009) 2773-2780.
[65] A.R. Yildiz, Hybrid immune-simulated annealing algorithm for optimal design and manufacturing, International Journal of Materials and Product Technology, 34 (2009) 217-226.
[66] A.R. Yildiz, A comparative study of population-based optimization algorithms for turning operations, Information Sciences, 210 (2012) 81-88.
[67] A.R. Yildiz, Hybrid Taguchi-differential evolution algorithm for optimization of multi-pass turning operations, Applied Soft Computing, 13 (2013) 1433-1439.
[68] A.R. Yildiz, A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing, Applied Soft Computing, 13 (2013) 2906-2912.
[69] A.R. Yildiz, Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach, Information Sciences, 220 (2013) 399-407.
[70] A.R. Yildiz, K. Saitou, Topology synthesis of multicomponent structural assemblies in continuum domains, Journal of Mechanical Design, 133 (2011).

Fong Cheng Weng is currently a researcher at Faculty of Computing, the Universiti Teknologi Malaysia. He received the B.Sc. degree in Computer Science from the Universiti Teknologi Malaysia, in 2012. His research interests focus on timetabling/scheduling.

Hishammudin bin Asmuni is a Senior Lecturer at the Faculty of Computing, Universiti Teknologi Malaysia. He received the Ph.D. degree in Computer Science from the University of Nottingham, UK in year 2008, M.Sc. degree in Computer Science and B.Sc. Computer Science from Universiti Teknologi Malaysia and Universiti Malaya, respectively. His research interests focus on timetabling, bioinformatics and iris recognition.

Barry McCollumis currently a Senior Lecturer at the School of Electronics, Electrical Engineering and Computer Science at The Queen's University Belfast, United Kingdom. His main professional academic interest is investigating how Intellectual Property developed through research can be applied within an industrial setting. His research interests cover complex fields of optimization, scheduling and timetabling.

Dr. Barry is a reviewer of several journals such as Journal of Operational research Society (JORS), European Journal of Operation Research, IEEE Transaction on Evolutionary Computing and Journal of Scheduling.



Paul McMullan is currently a Senior Lecturer in the School of Electronics, Electrical Engineering and Computer Science at The Queen's University of Belfast, United Kingdom. In addition, he also a director of two spin-out companies from Queen's University of Belfast, Realtime Solution Limited and EventMap Limited. His research interests include scheduling and optimization.

Sigeru Omatu is a Professor at the Department of Electronics, Information and Communication, Faculty of Engineering, Osaka Institute of Technology, Osaka, Japan since year 2010. He received the Ph.D. in Electronic Engineering from Osaka Prefecture University and joined the faculty at University of Tokushima in year 1969. He was Professor of University Tokushima and Osaka Prefecture University in years 1988 and 1995, respectively. His research interests include intelligent signal processing, pattern recognition, intelligent control, adaptive control and scheduling.

Prof. Omatu was an Associate Editor of IEEE Transaction on Neural Network and IMA Journal of Mathematic Control of Information (Oxford University Press).