J Sched (2014) 17:249–262
DOI 10.1007/s10951-013-0352-y

An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling

M. Alzaqebah · S. Abdullah

Received: 30 December 2011 / Accepted: 9 October 2013 / Published online: 3 December 2013

Abstract The artificial bee colony (ABC) is a population-based metaheuristic that mimics the foraging behaviour of honeybees in order to produce high-quality solutions for optimisation problems. The ABC algorithm combines both exploration and exploitation processes. In the exploration process, the worker bees are responsible for selecting a random solution and applying it to a random neighbourhood structure, while the onlooker bees are responsible for choosing a food source based on a selection strategy. In this paper, a disruptive selection strategy is applied within the ABC algorithm in order to improve the diversity of the population and prevent premature convergence in the evolutionary process. A self-adaptive strategy for selecting neighbourhood structures is added to further enhance the local intensification capability (adaptively choosing the neighbourhood structure helps the algorithm to escape local optima). Finally, a modified ABC algorithm is hybridised with a local search algorithm, i.e. the late-acceptance hill-climbing algorithm, to quickly descend to a good-quality solution. The experiments show that the ABC algorithm with the disruptive selection strategy outperforms the original ABC algorithm. The hybridised ABC algorithm also outperforms the lone ABC algorithm when tested on examination timetabling problems.

Keywords Artificial bee colony · Late-acceptance hill climbing · Examination timetabling problems · Selection strategy · Self-adaptive strategy

M. Alzaqebah (✉) · S. Abdullah
Universiti Kebangsaan Malaysia,
Bangi, Selangor, Malaysia
e-mail: malek.zaqeba@gmail.com

1 Introduction

The examination timetabling problem (ETTP) is one of the most difficult combinatorial optimisation problems and is considered to be NP-hard (Cooper and Kingston 1995; Schaerf 1999), being prevalent in many academic institutions (Carter et al. 1996). The ETTP can be considered as the process of allocating a set of examinations to a limited number of time slots while satisfying a predetermined set of constraints (Burke et al. 1996; Qu et al. 2009). A large number of approaches have been described and discussed for solving such examination timetabling problems, which can be classified into two main types: local-search-based and population-based approaches (Qu et al. 2009). Interested readers can find more details about examination timetabling research in Abdullah et al. (2009), Burke and Newall (2004), Burke et al. (1996, 2010), Burke et al. (2004), Carter (1986), Lewis (2008) and Turabieh and Abdullah (2011a,b).

Observation of group behaviours in natural self-organised systems motivated researchers to develop population-based approaches, which are known as swarm intelligence. Examples of population-based approaches for examination timetabling problems include the following: ant colony (Dowsland and Thompson 2005), particle swarm optimisation (Chu et al. 2006), fish swarm optimisation algorithm (Turabieh and Abdullah 2011a,b) and honeybee mating optimisation (Sabar et al. 2009). Swarm intelligence aims to simulate the behaviour of such self-organised systems. In particular, the intelligent behaviour of honeybees motivated the development of metaheuristic algorithms that model such behaviour in order to find better solutions to optimisation problems. Such honeybee algorithms can be classified based on three different groups of behaviour (Baykasoglu et al. 2007), i.e. foraging, marriage and queen bee behaviours. Algorithms that are based on the foraging behaviour of

honeybees are applied to solve optimisation problems, e.g. the artificial bee colony (ABC), bee algorithm (BA) and bee colony optimisation (BCO). These three algorithms have different behaviour models for the drone bees.

In this paper, we study the ABC algorithm that was first proposed by Karaboga (2005) for solving numerical optimisation problems. A further version of the ABC algorithm was later updated and proposed by Karaboga and Basturk (2007) for solving constrained optimisation problems. This algorithm has drawn great attention from researchers (Bao and Zeng 2009; Kang et al. 2009; Karaboga and Akay 2009; Pham et al. 2006; Singh 2009). The ABC algorithm works based on local communication between three groups of bees (i.e. scouts, workers and onlookers) and with their environment, contributing to the collective intelligence of the bee colony (Karaboga 2005).

This paper focusses on hybridising the ABC and the late-acceptance hill-climbing (LAHC) algorithm. The LAHC algorithm is a local search method that was introduced by Burke and Bykov (2008). The algorithm has the capability to quickly explore the search space and accept a worse solution based on adaptive criteria using fewer parameters, in contrast to the ABC algorithm, which only accepts an improved solution. This capability is believed to be able to prevent the algorithm from becoming stuck in local optima. Thus, better solutions can be obtained.

In addition, the performance of the ABC algorithm can be enhanced by using a selection strategy and a self-adaptive mechanism as previously presented by Alzaqebah and Abdullah (2011a,b).

This paper is organised as follows: Sect. 2 presents the examination timetabling problem and its formulation, while the artificial bee colony algorithm is presented in Sect. 3. The proposed approach is discussed in Sect. 4. The experimental results are presented in Sect. 5, and conclusions are given in Sect. 6.

2 Problem description and formulation

In this paper, the problem description is divided into two distinct parts: the Toronto datasets and the International Timetabling Competition (ITC2007) datasets, as discussed below:

2.1 The Toronto datasets

These datasets were introduced by Carter et al. (1996), and were considered as uncapacitated examination timetabling problems, i.e. where room capacity requirements are not taken into account. The description of the problem is adapted from Burke and Newall (2004). In these datasets the problem consists of the following inputs:

- N is the number of examinations.
- E_i is an examination, $i \in \{1, \ldots, N\}$.
- T is the given number of available time slots.
- M is the total number of students.
- $C = (c_{ij})_{N \times N}$ is the conflict matrix, where each element denoted by $c_{ij}, i, j \in \{1, \ldots, N\}$ is the number of students taking examination i and j.
- t_k $(1 \leq t_k \leq T)$ specifies the assigned time slot for examination k $(k \in \{1, \ldots, N\})$.

The minimisation of the fitness function in Eq. 1 is formulated to space out students' examinations throughout the examination period:

$$\text{Min} \frac{\sum_{i=1}^{N-1} F_1(i)}{M}, \tag{1}$$

where

$$F_1(i) = \sum_{\text{j=i+1}}^{\text{N}} c_{ij} \cdot \text{Proximity}(t_i, t_j) \tag{2}$$

with

$$\text{Proximity}(t_i, t_j) = \begin{cases} 2^5/2^{|t_i - t_j|} & \text{if } 1 \leq |t_i - t_j| \leq 5 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

subjected to

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} \cdot \gamma(t_i, t_j) = 0,$$

where

$$\gamma(t_i, t_j) = \begin{cases} 1 & \text{if } t_i = t_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Equation 2 represents the penalty for examination i, which is given by the proximity value multiplied by the number of students in conflict. Equation 3 represents a proximity value between two examinations (Carter et al. 1996). Equation 4 represents a *clash-free* requirement so that no student is allocated to sit more than one examination at the same time. The *clash-free* requirement is considered to be a hard constraint.

Table 1 presents the details of the Toronto datasets (Qu et al. 2009).

2.2 The International Timetabling Competition (ITC2007) datasets

ITC2007 introduced three tracks of problems: examination timetabling, curriculum-based course timetabling and post-enrolment course timetabling problems. In this paper, the focus is on the first track, i.e. the examination timetabling problems, which include a number of real-world constraints. The details of the datasets are given in Table 2 followed by

Table 1 Toronto datasets

Dataset	Number of time slots	Number of examinations	Number of students	Conflict density
car92	32	543	18,419	0.14
car91	35	682	16,925	0.13
ear83 I	24	190	1,125	0.27
hec92 I	18	81	2,823	0.42
kfu93	20	461	5,349	0.06
lse91	18	381	2,726	0.06
pur93 I	42	2,419	30,032	0.03
rye92	23	486	11,483	0.07
sta83 I	13	139	611	0.14
tre92	23	261	4,360	0.18
uta92 I	35	622	21,267	0.13
ute92	10	184	2,750	0.08
yor83 I	21	181	941	0.29

Table 2 ITC2007 examination datasets

Dataset	D1	D2	D3	D4	D5	D6	CD
Exam_1	7,833	607	54	7	12	0	5.05
Exam_2	12,484	870	40	49	12	2	1.17
Exam_3	16,365	934	36	48	170	15	2.62
Exam_4	4,421	273	21	1	40	0	15.0
Exam_5	8,719	1,018	42	3	27	0	0.87
Exam_6	7,909	242	16	8	23	0	6.16
Exam_7	13,795	1,096	80	15	28	0	1.93
Exam_8	7,718	598	80	8	20	1	4.55

(*D1* number of students, *D2* number of examinations, *D3* number of time slots, *D4* number of rooms, *D5* period hard constraints, *D6* room hard constraints, *CD* conflict density)

Table 3 Hard constraints

Hard constraint	Explanation
H1	No student should be sitting more than one examination at the same time
H2	The total number of students assigned to each room in each period cannot exceed the room capacity
H3	The length of examinations assigned to each time slot should not violate the time slot length
H4	Some sequences of examinations have to be respected, e.g. exam A must be scheduled before exam B
H5	Room-related hard constraints must be satisfied, e.g. exam A must be scheduled in room 80

a set of hard and soft constraints, listed in Tables 3 and 4, respectively (McCollum et al. 2010).

A feasible timetable is one where each exam is assigned to a period and room, and there is no violation of hard constraints. The objective is to minimise the violation of soft constraints as given in Eq. 5 (McCollum et al. 2010).

Table 4 Soft constraints

Soft constraint	Mathematical symbol	Explanation
S1	C_S^{2R}	Two examinations in a row: minimise the number of consecutive exams in a row for a student
S2	C_S^{2D}	Two examinations in a day: students should not be assigned to sit more than two examinations in a day. Of course, this constraint becomes important only when there are more than two examination periods on the same day
S3	C_S^{PS}	Periods spread: all students should have a fair distribution of exams over their timetable
S4	C_S^{2NMD}	Mixed durations: the numbers of examinations with different durations that are scheduled into the same room has to be minimised as much as possible
S5	C^{FL}	Larger examinations appearing later in the timetable: minimise the number of examinations of large class size that appear later in the examination timetable (to facilitate the assessment process) w^{PS}
S6	C^{P}	Period penalty: some periods have an associated penalty; minimise the number of examinations scheduled in penalised periods
S7	C^{R}	Room penalty: some rooms have an associated penalty; minimise the number of examinations scheduled in penalised rooms

Table 5 Associate weights of the ITC2007 collection of examination datasets

Dataset	w^{2D}	w^{2R}	w^{PS}	w^{NI}	w^{FL}	w^{P}	w^{R}
Exam_1	5	7	5	10	100	30	5
Exam_2	5	15	1	25	250	30	5
Exam_3	10	15	4	20	200	20	10
Exam_4	5	9	2	10	50	10	5
Exam_5	15	40	5	0	250	30	10
Exam_6	5	20	20	25	25	30	15
Exam_7	5	25	10	15	250	30	10
Exam_8	0	150	15	25	250	30	5

$$\min \sum_{s \in S} \left(W^{2R} C_S^{2R} + W^{2D} C_S^{2D} + W^{PS} C_S^{PS} \right) + \left(W^{NMD} C_S^{2NMD} + W^{FL} C^{FL} + W^{P} C^{P} + W^{R} C^{R} \right). \quad (5)$$

Each dataset has its own set of weights as presented in Table 5 (McCollum et al. 2010).

3 Artificial bee colony (ABC) algorithm

3.1 Basic artificial bee colony (ABC) algorithm

The ABC algorithm was originally developed based on the foraging behaviour of real honeybees in finding and sharing information on food sources in their hives (Kang et al. 2009; Karaboga and Basturk 2007, 2008; Karaboga 2009; Pham et al. 2006; Bao and Zeng 2009).

The ABC system consists of three groups of agents (scout, worker and onlooker bees). In the ABC algorithm, the position of a food source represents a possible solution, and the quality of nectar in the food source corresponds to the quality (fitness value) of this solution. The number of worker bees is equal to the number of solutions in the population. Table 6 illustrates the analogy between the natural and artificial bee colonies.

The algorithm begins with a population of randomly generated solutions (or food sources); then, steps 1–4 (in Fig. 1) are repeated until a termination criterion is met (Karaboga 2005, 2009).

The search process of the original ABC algorithm starts with initialisation of the population (food sources). After the initialisation, the worker bees adjust the food source position in their memories and start to discover the position of nearby food sources (step 1). If the quality of nectar in the new food source is higher than the previous food source, the bees memorise the position of the new source and forget the old one. After all the worker bees have completed the search process, they share the information on the sources with the onlooker bees by doing a *wiggle dance*. Each onlooker bee watches the dance, chooses one food source, and searches locally in the neighbourhood of the selected food source (step 2). In step 3, the scout bees find the abandoned sources and replace them by randomly produced new food sources. Finally, the algorithm memorises the best food source found so far (step 4).

Table 6 Analogy between natural and artificial bee colonies

Natural bees	ABC
Food source	Solution
Quality of nectar	Fitness function
Onlooker	Exploitation process
Worker	Exploration process
Scout	Scouting process (also can be exploration for new solutions)

```
Initial food sources are produced for all
employee bees.
REPEAT
1. Send the employee bees to explore the
   search space by visiting the neighbour
   of the food sources (the exploration
   process).
2. Onlooker bees select a food source
   based on the information given by
   employee bees and start to exploit
   them (the exploitation process)
3. Determine the scout bees and find new
   possible food sources (the scouting
   process).
4. Register the best food source found so
   far.
UNTIL (termination criterion is met)
```

Fig. 1 Original ABC search algorithm

3.2 The selection process

In the original ABC algorithm, the onlooker bees applied a stochastic selection strategy based on *roulette-wheel selection* to select a food source based on the information gathered from the worker bees, which can be summarised as follows:

(i) Calculate the value of fit_i as follows:

$$\text{fit}_i = \frac{1}{1 + f_i}, \tag{6}$$

where f_i is the fitness value of the ith food source. (f is calculated using Eq. 1 or 5 for the Toronto or ITC2007 datasets, respectively.)

(ii) Calculate the probability P_i as follows:

$$P_i = \frac{\text{fit}_i}{\sum_{i=1}^{\text{SN}} \text{fit}_i}, \tag{7}$$

where SN is the number of food sources and P_i is the probability of the ith food source.

(iii) Finally, from all the solutions (food sources), select a candidate solution with probability P_i using *roulette-wheel selection* as discussed in Holland (1975).

There are two problems when using the basic ABC selection strategy as stated in Bao and Zeng (2009): (i) The super-individual is selected too often, which makes the whole population tend to converge towards its position. This will cause loss of diversity because the same solutions are selected too often; (ii) When all the solutions converge to the same area (because of the loss in diversity), the new generated solution will be from the same area as these solutions, so it is not guaranteed that the generated solution will be better than the existing one. Furthermore, this generated solution cannot be added to the current population. Consequently, the algorithm becomes stuck in a local optimum, thus preventing a better solution from being found. To alleviate these problems, a disruptive selection strategy, as introduced by Kuo and Huang (1997) and explained in Sect. 3.2, is therefore applied.

The findings from our previous work in Alzaqebah and Abdullah (2011a) showed that the ABC algorithm with the disruptive selection strategy is able to explore the search space better than either the original ABC algorithm or the ABC algorithm with tournament and rank selection strategies. The tournament selection strategy randomly selects a number of solutions and compares them based on a probability. The solution with the highest fitness value is chosen. In the rank selection strategy, the solutions are ranked based on the fitness values, so it is biased towards solutions with higher rank (i.e., better fitness). Meanwhile, the disruptive selection strategy gives preference to both low- and high-quality solutions, and tries to retain population diversity by improving the worse-fitness solutions concurrently with the high-fitness solutions. This motivated us to use the disruptive selection strategy within the ABC algorithm in this work.

Disruptive selection provides more chances for higher and lower individuals to be selected by changing the definition of the fitness function as shown in Eq. 8 (Kuo and Huang 1997).

$$\text{fit}_i = |f_i - \overline{f_i}|P_i = \frac{\text{fit}_i}{\sum_{i=1}^{\text{SN}} \text{fit}_i}, \tag{8}$$

where $\overline{f_i}$ is the average value of the fitness function f_i of the individuals in the population.

4 The proposed algorithm

4.1 Neighbourhood search operations

In this paper, the following neighbourhood operations are employed to enhance the search algorithm performance (adapted from Abdullah and Burke 2006; Abdullah et al. 2007a,b):

Nbs1 Select two examinations at random and swap their time slots (including rooms if rooms are available) feasibly.
Nbs2 Select a single examination at random and move to a new feasible time slot (including a new room if a room is available).
Nbs3 Select four examinations randomly and swap the time slots (including rooms if rooms are available) between them feasibly.
Nbs4 Select two examinations at random and move to new random feasible time slots (including random feasible new rooms if rooms are available).

The neighbourhood is selected by using a self-adaptive method as discussed in Sect. 4.2. Note that, for the ITC2007 datasets, the room is considered to be included in the above neighbourhood operators.

4.2 Self-adaptive method for neighbourhood search

During the search processes performed by the worker and onlooker bees, a self-adaptive strategy is applied in finding the neighbouring food sources, which is explained as follows (Pan et al. 2011):

(i) The self-adaptive method starts by filling a neighbour list (NL) with the four neighbourhood operations (as in Sect. 4.1) that are randomly selected. Note that the list NL with a specified length is generated previously.
(ii) During the search process, one neighbourhood operation is taken from NL and is used to generate a new food source for a worker or onlooker bee. If the new food source is better than the current one, this approach inserts the neighbourhood operation into a new list known as the winning neighbouring list (WNL). The process is repeated until NL becomes empty.
(iii) When NL becomes empty, 75 % of NL is refilled from the WNL list, the remaining 25 % is refilled randomly from the four neighbourhood search operations, and WNL is reset to zero. If WNL is empty, then the most recent NL is used again (Pan et al. 2011).

By using the self-adaptive method, suitable neighbourhood operations are learned and the winning ones are reused later. The chance of using the winning neighbourhood operations is higher than for others due to the fact that the NL list is 75 % filled from WNL. This method also has some randomness, i.e. 25 % of the NL. In this paper, the length of NL is set to 200 as stated in Pan et al. (2011). The details of the self-adaptive method can be found in Alzaqebah and Abdullah (2011b).

4.3 Late-acceptance hill climbing (LAHC)

The simple local search method LAHC is embedded into the basic ABC algorithm in order to quickly explore the search space in the ABC algorithm and prevent it from becoming stuck in local optima. This is due to the use of a greedy acceptance criterion in the basic ABC that only accepts an improved solution and eliminates the worst.

LAHC seems to be as fast as simple hill climbing but more powerful. This is due to the fact that LAHC is able to accept a worse solution based on intelligent use of an adaptive memory to keep information from previous iterations and reuse it later (Burke and Bykov 2012).

The main idea behind the LAHC method is based on a memory $\hat{C}$ (LAHC list) of size L that is used to memorise the fitness values of the previous solutions. During the search

process, the acceptance decision is made according to a comparison between the candidate solution and the previous solution obtained at the Lth step, as follows: the candidate solution is accepted if its fitness value is better than or equal to the fitness value in the list $\hat{C}$ with index v (the virtual beginning of the list). The virtual beginning (v) is calculated dynamically as the remainder of the integer division of the current iteration number I by the length (L) (see Eq. 9).

$$v = I \mod L. \tag{9}$$

Figure 2 (lines 27–49) shows the pseudo-code for the LAHC algorithm (Burke and Bykov 2012). The stopping condition used in this algorithm is described in Sect. 5.

4.4 Adaptive artificial bee colony with LAHC algorithm

Figure 2 illustrates the pseudo-code that represents the approach embedding the LAHC algorithm, as used in this paper.

As shown in Fig. 2, the algorithm starts with a feasible initial solution, as described in Sect. 4.5. The population is initialised as follows: (i) select a number of examinations at random and swap their time slots and/or rooms; (ii) select a number of examinations at random and move them to feasible time slots and/or rooms. There are three processes in each iteration. In the first process, the worker bees work on random solutions and apply neighbourhood operators based on the self-adaptive method (as explained in Sect. 4.2) to all solutions in the population in order to find more profitable ones.

In the second process, the solutions are arranged based on the profitability. The onlooker bees select a solution based on one of the three selection strategies (as explained in Sect. 4), and then a local search (LAHC) is utilised (as explained in Sect. 4.3). Finally, in the last process, the scout bees determine the abandoned food source (i.e. the food source that has been visited by all other bees without improvement) and replace it with a new food source (which is generated as the food source when the population is initialised).

```
1  Initialisation:
2  Initialise the initial population and evaluate
3  the fitness;
4  Calculate the initial fitness value, f(Sol);
5  Set best solution, Solbest← Sol;
6  Set maximum number of iteration, NumOfIte;
7  Set the population size;
8  //where population size = # of Onlooker Bees =
9  # of Employee Bees;
10 iteration ←   0;
11 Improvement:
12 do while (iteration<NumOfIte)
13   for i=1: # of Employee Bee
14      Selects a random solution from the
15        population, and applies random
16        neighbourhood structure (for Self-
17        Adaptive is based on Self-Adaptive
18        method as in section 4.2);
19   end for                                  Exploration process
20   for i=1: # of Onlooker Bees
21     Calculates the selection probability Pi,
22     based on basic ABC selection or
23     disruptive selection as in Equation (7)
24     or (8), respectively.
25     Sol* ← selects the solution depending
26     on Pi;
27     Start local search(LAHC) on Sol*;
28
29       s← Sol*; Ĉ: LAHC List;
30       I= number of iterations;
31       L=the length of the list;
32       for all k ∈{0...L-1} do Ĉk ← f(s)
33       I ← 0;
34       do until a chosen stopping condition
35       prevails:
36         shuffles a candidate solution s*;
37         (apply number of Nbs4 without
38         violating any hard constraint)
39         Calculates its fitness function
40         f(s*);
41         Calculate the virtual beginning (v)
42         v = I mod L;
43         if f(s*) ≤ Ĉv
44           accept candidate (s ← s*)
45         Inserts penalty value into the list
46         Ĉv←f(s)
47         I ← I+1;
48       end do                             Exploitation process
49     end local search
50     if (f(s)< f(Sol*))
51       Sol* ←s;
52     end if
53   end for
54   Solbest←best solution found so far;
55   Scout bees determine the abandoned food
56   source and replace it with the new food
57   source.                                Scouting process
58   iteration++;
59 end do
```

Fig. 2 Pseudo-code for ABC with the LAHC algorithm

4.5 Initial solution construction

The explanation of the application of two constructive heuristic algorithms in constructing the feasible initial solutions for the Toronto and ITC2007 datasets is given below:

- *Toronto datasets*: A largest degree graph colouring heuristic (Carter et al. 1996) is used to generate initial solutions.
- *ITC2007 datasets*: The examination with the largest number of hard constraints is scheduled first (step 1). Then, the largest degree heuristic (Carter et al. 1996) is applied by randomly selecting a time slot and a room that satisfy the hard constraints (step 2). If the examination cannot be scheduled to a specific room, then it is allocated to any randomly selected room. In some cases, a feasible timetable can be obtained after employing these two steps. However, if a feasible initial solution is not obtained, certain exams are moved (into different time slots and/or rooms) or swapped until feasibility is achieved (step 3).

5 Simulation results and comparison

In all our experiments, three different modifications of the ABC algorithm were investigated: (i) the ABC algorithm based on disruptive selection (denoted DABC), which emphasised the selection strategy chosen based on the experimental results presented in Sect. 5.1; (ii) the DABC algorithm with the self-adaptive method for neighbourhood search (denoted SA-DABC); and (iii) SA-DABC with a local search (LAHC) (denoted LAHC-SA-DABC).

The performance of these modifications was compared with the basic ABC algorithm to show the effects of employing the different modifications. Table 7 presents the final setting of the parameters, which were experimentally chosen from a total of ten runs to obtain average results (note that the experiments were carried out on Intel® Core™ i3 processors).

Note that the length of the LAHC list of 500 is adopted from Burke and Bykov (2008). The stopping condition for LAHC is 500 iterations, or until there is no more improvement after a number of non-improved moves (set to 50 in this work).

5.1 Toronto dataset experimental results

This section presents a comparison of the performance among the basic ABC, DABC, SA-DABC and LAHC-SA-DABC algorithms when tested on the Toronto datasets, as presented in Table 8. Note that all the algorithms in Table 8 use the same computational resources. The best results are highlighted in bold. Note that these results were obtained from 11 independent runs and the central processing unit (CPU) times are between 300 and 9,600 s based on Intel® Core™ i3 processors.

Table 7 Parameters: final setting

Parameter	Value
No. of iterations	500
Population size = no. of onlooker bees = no. of worker bees	50
No. of scout bees	1

The comparison in Table 8 shows that the three modified versions of the basic ABC algorithm perform better than the lone basic ABC algorithm. From Table 8, it can also be deduced that the disruptive selection strategy (DABC) outperforms the lone basic ABC algorithm. Use of the self-adaptive method to select the neighbourhood search operations further enhances the quality of the solutions. Applying the local search (LAHC algorithm) within the SA-DABC aids the algorithm to produce better solutions.

Figure 3 shows the resulting convergence graphs when applying the basic ABC, DABC and LAHC-SA-DABC algorithms on the Toronto datasets.

The lines in the graphs represent the trends between the number of iterations and the solution quality (penalty). These graphs show that the quality of the solution obtained by the basic ABC algorithm was greatly improved by the modifications, whereas the convergence of the lines in these graphs shows how the ABC, DABC and LAHC-SA-DABC algorithms explore the search space.

The behaviour of the algorithms is similar at the beginning of the search, where improvement of the solutions can be easily obtained. However, later, when the search becomes steady, it becomes harder to improve a solution. Regarding the effect of the LAHC algorithm, it can be easily seen that it is more flexible in accepting worse solutions, which later provides a better chance of finding a better solution compared with the ABC and DABC algorithms. It can be concluded that employing the local search (LAHC in

Table 8 Results of comparisons between the algorithms on the Toronto datasets

Instance	ABC		DABC		SA-DABC		LAHC-SA-DABC	
	Best	Average	Best	Average	Best	Average	Best	Average
car91	5.86	5.97	5.42	5.83	5.01	5.08	4.62	4.74
car92	4.92	5.12	4.84	4.9	4.31	4.36	4.00	4.08
ear83 I	38.34	38.63	37.54	37.73	35.08	35.92	33.14	33.72
hec92 I	11.51	11.85	11.21	11.52	11.13	11.31	10.43	10.59
kfu93	16.04	16.49	15.13	15.83	14.48	14.67	13.59	13.86
lse92	12.42	12.62	12.06	12.62	11.49	11.72	10.75	11.00
rye	11.37	11.74	10.48	10.52	10.27	10.36	9.17	9.54
sta83 I	158.12	158.47	157.52	157.76	157.43	157.48	157.06	157.16
tre92	9.58	9.81	9.23	9.79	8.68	8.76	8.00	8.14
uta92 I	3.99	4.28	3.94	4.02	3.47	3.53	3.27	3.33
ute92	27.8	27.98	27.57	27.9	26.17	26.44	25.16	25.37
yor83 I	41.44	41.42	40.94	41.23	38.48	39.30	35.58	36.32

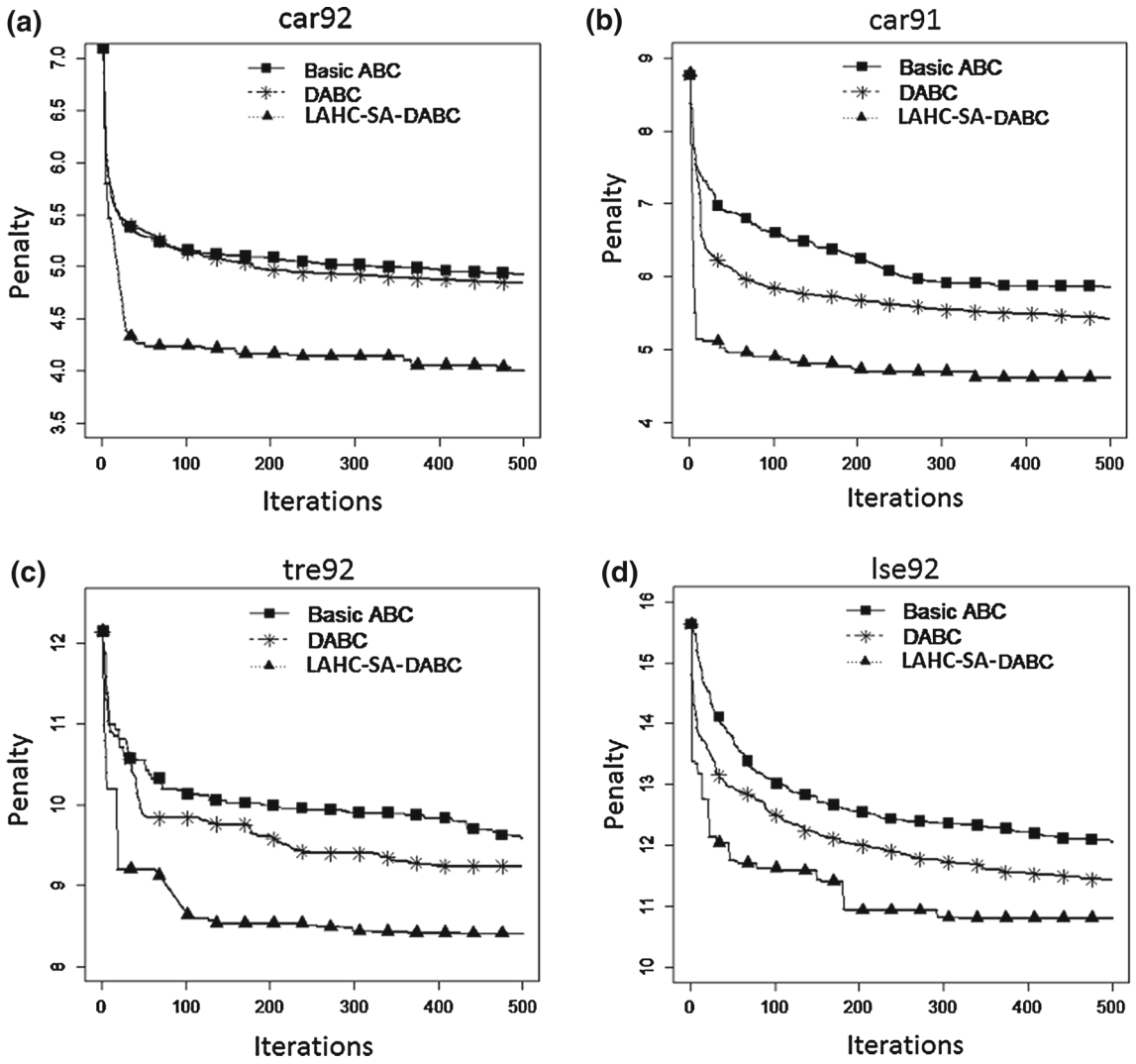

Fig. 3 Convergence graphs for the Toronto datasets

this case) within DABC helps to descend quickly to high-quality solutions. In addition, the self-adaptive method for selecting neighbourhood search operations also helps the algorithm to determine the best neighbourhood structure, unlike selecting a neighbourhood structure at random as applied in ABC and DABC. This is indicated by the greater improvement in the fitness value with the self-adaptive mechanism.

5.2 International Timetabling Competition (ITC2007) dataset experimental results

The performance of the basic ABC, DABC, SA-DABC and LAHC-SA-DABC algorithms was also tested on the International Timetabling Competition (ITC2007) datasets. Table 9 presents a comparison of the results produced by these algorithms. The best results are highlighted in bold. Note that these results were obtained from 11 independent runs with a stopping condition of 460 s (as a result of using the competition computation tools to determine the "time limit" based on Intel® Core™ i3 processors).

Table 9 shows that, once again, there is a significant improvement as ABC progresses to DABC, then to SA-DABC and finally to LAHC-SA-DABC.

The graphs in Fig. 4 show the status of the improvement of the population for the ABC, DABC and LAHC-SA-DABC algorithms. The diamond symbol in the graphs indicates how the initial population was spaced throughout the search process, and the triangle symbol represents the improved solutions in the population after 460 s. From this figure, it can be concluded that the DABC and LAHC-SA-DABC algorithms improved the solutions of the population. This can be observed in the spacing of the triangle symbols for the DABC and LAHC-SA-DABC algorithms, which are located much closer to each other. This indicates the closeness of the quality of the solutions in the population, unlike

Table 9 Results of the comparisons between the algorithms on the ITC2007 datasets

Dataset	ABC		DABC		SA-DABC		LAHC-SA-DABC	
	Best	Average	Best	Average	Best	Average	Best	Average
Exam_1	6,971	7,124.7	6,552	6,590.8	6,236	6,420.55	5,328	5,517.3
Exam_2	989	1,012.2	893	925.4	809	849.36	512	537.9
Exam_3	11,912	12,114.7	11,441	11,614.3	11,390	11,461.18	10,178	10,324.9
Exam_4	17,470	17,454.5	17,168	17,197.5	16,937	16,988.45	16,465	16,589.1
Exam_5	4,261	4,368.8	3,864	3,970.6	3,797	3,805.73	3624	3,631.9
Exam_6	26,905	27,217	26,845	26,867.5	26,755	26,779.09	26,240	26,275
Exam_7	6,247	6,572.1	5,480	5,841.2	4,847	4,945.09	4,562	4,592.4
Exam_8	10,653	10,913.2	9,888	9,954.4	8,501	8,764.36	8,043	8,328.8

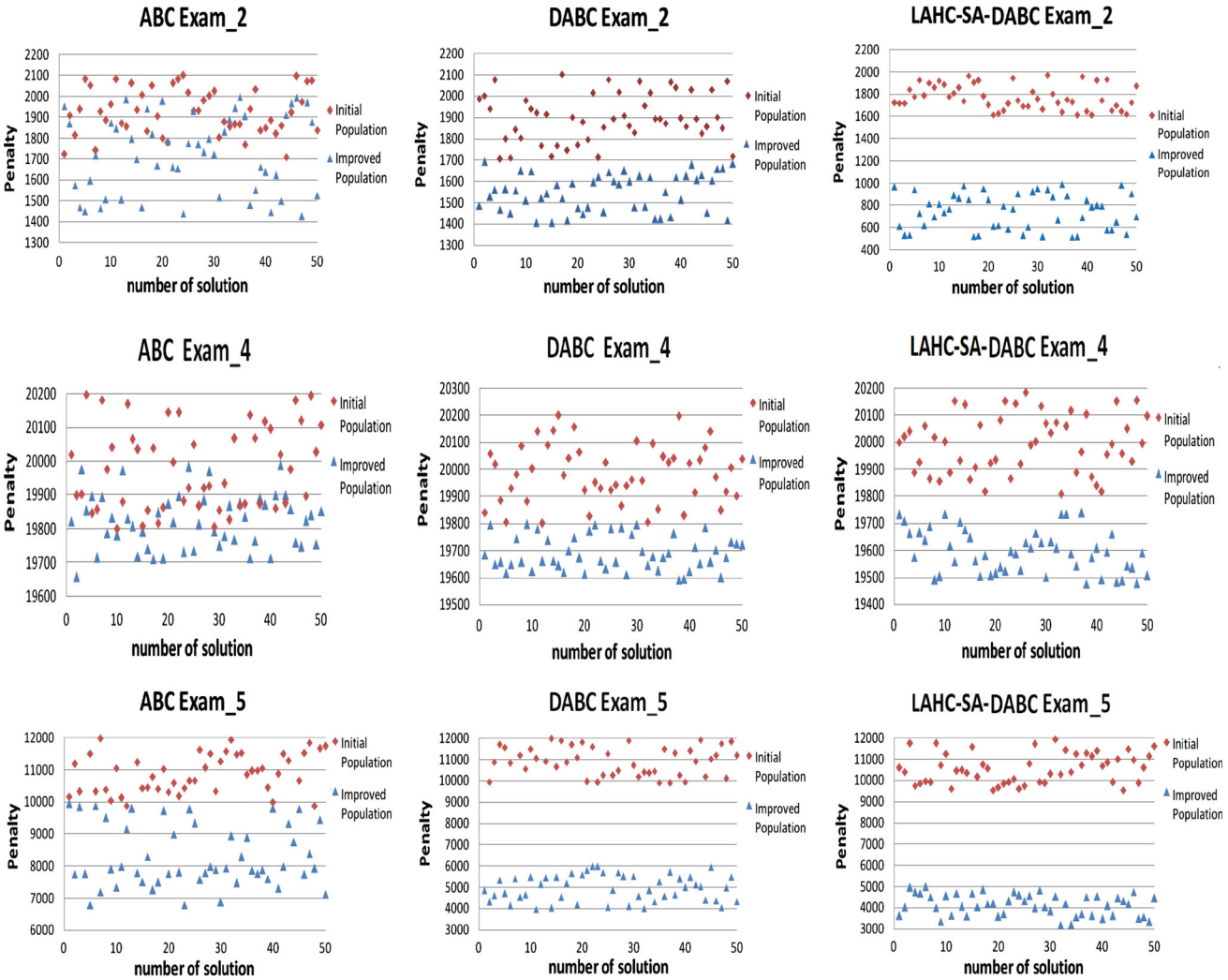

Fig. 4 Convergence graphs for three datasets (Exam_2, Exam_4 and Exam_5)

the spacing found in the lone basic ABC, which appears to be more scattered.

Figure 5 shows the behaviour of the proposed algorithms during the search process when tested on the ITC2007 datasets. Again from this figure, it can be easily seen that the use of the stochastic selection strategy in the basic ABC algorithm can lead to premature convergence of the algorithm and limits the search scope by repeatedly choosing only high-quality individuals. This can be noted by the curve representing DABC, which shows a downward slope that is steeper than for the basic ABC algorithm. Later, when both the local search and self-adaptive method are added to the algorithm, it

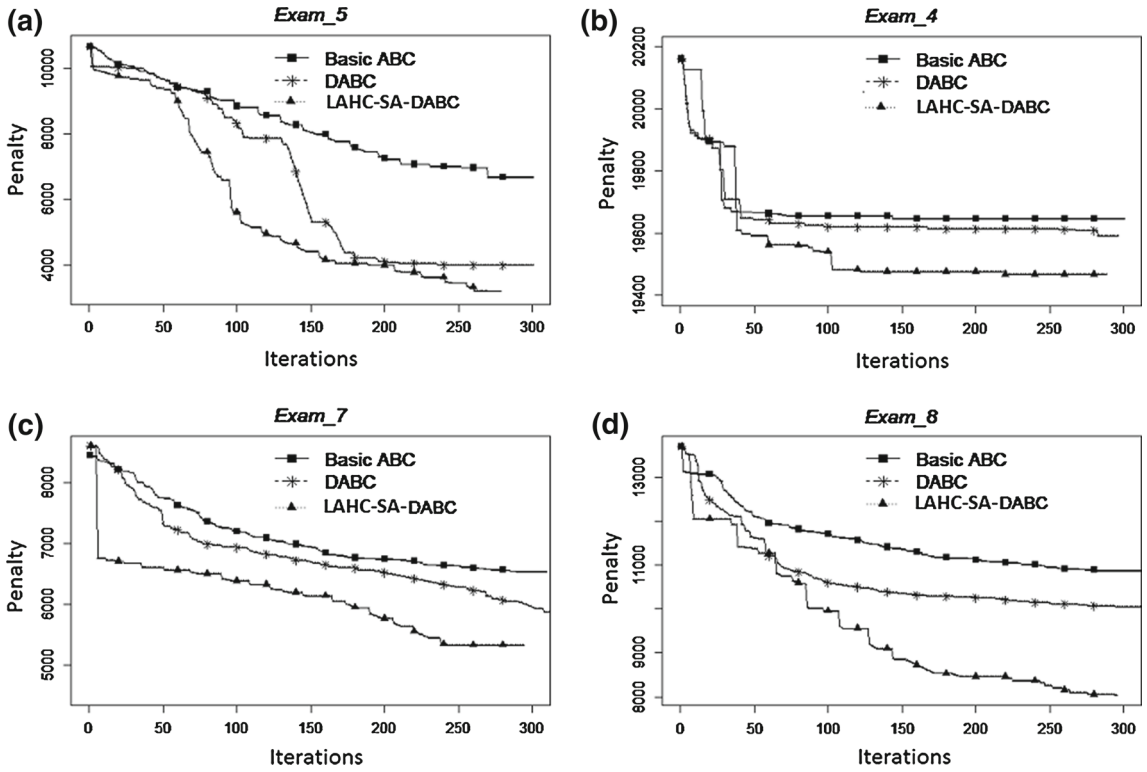

Fig. 5 Convergence graphs for the ITC2007 datasets

is obvious from the presented figures that the embedded local search proves to be the dominant factor in obtaining better solutions in the ABC algorithm, where the local search helps to intensify the selected solution obtained by the onlooker bees during the exploitation process (as shown in Fig. 2).

Based on the graphs presented in Fig. 5, it can be seen that the ABC and DABC algorithms reach the stopping condition at about 300–320 iterations, but the LAHC-SA-DABC algorithm reaches the stopping condition at about 200–300 iterations (given the same stopping condition). This shows that, even with a smaller number of iterations (representing a smaller number of candidate solutions), the LAHC-SA-DABC algorithm is still able to obtain better results. It is believed that this is due to the ability of the adaptive mechanism in selecting the correct neighbourhood search operations (based on the quality of the current solution in hand rather than a random selection of neighbourhood search operations), and the embedded local search can further improve the performance of the algorithm in finding better solutions. Thus, these results clearly show the effectiveness (in terms of the quality of the solution obtained in a smaller number of iterations) of the modified algorithm compared with the basic ABC approach.

It is also observed that the ABC algorithm with the disruptive selection strategy is able to better explore the search space and transport all the solutions to converge together. The introduction of the self-adaptive strategy (compared with the stochastic selection strategy) in selecting neighbourhood structures manages to help the algorithm to escape from local optima. The results from Table 9 also show that the LAHC-SA-DABC algorithm is able to obtain better solutions compared with the other proposed approaches on all the tested datasets.

5.3 Results analysis and comparison with the best known results for the Toronto datasets

A comparison between the best results obtained from the tests run in this paper, i.e. using LAHC-SA-DABC, with the best known results (the three best approaches from the survey by Qu et al. (2009), i.e. Caramia et al. (2009), Yang and Petrovic (2005) and Burke and Bykov (2008)) for the Toronto datasets is presented in Table 10. The best results are highlighted in bold. In addition, statistical analysis of the LAHC-SA-DABC and the best literature approaches was conducted to identify any significant differences between them. These are presented in Tables 11 and 12.

For the statistical analysis, the Friedman test was first employed, followed by the Holm and Hochberg tests as post hoc methods (if a significant difference was detected) to obtain the adjusted p value for each comparison between a control algorithm (the best performing one) and the rest (Garcia et al. 2010, 2009). Table 11 summarises the ranking obtained by the Friedman test, where the lowest rank reflects

Table 10 Toronto datasets: experimental comparisons with the best approaches

Dataset	LAHC-SA-DABC		Caramia	Yang and Petrovic		Burke and Bykov	
	Best	Average	Best	Best	Average	Best	Average
car91I	4.62	4.74	6.6	**4.50**	4.53	4.58	4.68
car92I	4.00	4.08	6.2	3.93	3.99	**3.81**	3.92
ear83 I	33.14	33.72	**29.3**	33.7	34.87	32.65	32.91
hec92 I	10.43	10.59	**9.2**	10.83	11.36	10.06	10.22
kfu93	13.59	13.86	13.8	13.82	14.35	**12.81**	13.02
lse91	10.75	11.00	**9.6**	10.35	10.78	9.86	10.14
rye92	9.17	9.54	**6.8**	8.53	8.79	7.93	8.06
sta83 I	157.06	157.16	158.2	158.3	158.02	**157.03**	157.05
tre92	8.00	8.14	9.4	7.92	8.1	**7.72**	7.89
uta92 I	3.27	3.33	3.5	**3.14**	3.2	3.16	3.26
ute92	25.16	25.37	**24.4**	25.39	26.1	24.79	24.82
yor83 I	35.58	36.32	36.2	36.53	36.88	**34.78**	35.16

Table 11 Average (Friedman) ranking of the algorithms on the Toronto datasets

Algorithm	Ranking
Burke and Bykov (2008)	1.5
LAHC-SA-DABC	1.66
Yang and Petrovic (2005)	2.833

Table 12 Adjusted (Friedman) p values on the Toronto datasets

Algorithm	Adjusted	PHolm	PHoch
Yang and Petrovic (2005)	0.6833	0.6833	0.6833
LAHC-SA-DABC	0.0011	0.0020	0.0020

the best algorithm. Note that Caramia et al. (2009) were not included in the statistical analysis because their average values are not available.

The p value computed by the Friedman test was 0.0018, which is below the critical level ($\alpha = 0.05$). This value shows that there is a significant difference in the observed results. The post hoc methods (Holm and Hochberg tests) were also applied for the LAHC-SA-DABC algorithm. Table 12 presents the adjusted (Friedman) p value, where the Friedman test considers the Burke and Bykov algorithm as a control algorithm.

The Holm and Hockberg procedures show a significant difference when using the Burke and Bykov algorithm as the control. The LAHC-SA-DABC algorithm is better than the Yang and Petrovic algorithm and comparable to the Burke and Bykov algorithm, with $\alpha = 0.05$ and $\alpha = 0.01$ (1/3 algorithms).

This comparison with the best known results shows that, even though these tests were unable to beat any of the best known results in the literature, they were still able to produce "good enough" solutions.

5.4 Results analysis and comparison with the best known results on the International Timetabling Competition (ITC2007) datasets

Table 13 compares the proposed algorithm with the four best approaches in the literature. Note that these approaches and the approach employed here use 11 runs for each dataset and the same CPU time (as set in the ITC2007 computation rule). The best results are highlighted in bold.

The LAHC-SA-DABC algorithm is able to obtain solutions that are better than some of those obtained with the compared approaches, such as those of Atsuta et al. and Pillay, on all the tested datasets. It is highlighted here that the difference between these results and the best known results in the literature is in the range of 5.2–26.4 %.

The results were also analysed and compared against five winners of the International Timetabling Competition 2007 that can be found at http://www.cs.qub.ac.uk/itc2007/ as listed below:

First place: Tomas Müller
Second place: Christos Gogos
Third place: Mitsunori Atsuta, Koji Nonobe and Toshihide Ibaraki
Fourth place: Geoffrey De Smet
Fifth place: Nelishia Pillay

Friedman's test was applied to the ITC2007 datasets, yielding a p value of 9.425E-6, which is below the significance level of $\alpha = 0.05$, showing that there is a significant difference in the observed results.

Table 14 presents the average algorithm rankings found by the Friedman test, where a lower value indicates a better rank.

Table 13 ITC2007 datasets: experimental comparison with the best approaches

Dataset	Muller (2009)	Atsuta et al. (2007)	Pillay (2007)	Gogos et al. (2008)	LAHC-SA-DABC	
					Best	Average
Exam_1	**4,370**	8,006	12,035	5,905	5328	5517.3
Exam_2	**400**	3,470	3,074	1,008	512	537.9
Exam_3	**10,049**	18,622	15,917	13,862	10,178	10,324.9
Exam_4	18,141	22,559	23,582	18,674	**16,465**	16,589.1
Exam_5	**2,988**	4,714	6,860	4,139	3,624	3,631.9
Exam_6	26,950	29,155	32,250	27,640	**26,240**	26,275
Exam_7	**4,213**	10,473	17,666	6,683	4,562	4,592.4
Exam_8	**7,861**	14,317	16,184	10,521	8,043	8,328.8

Table 14 Average (Friedman) rankings of the algorithms on ITC2007

Algorithm	Ranking
Muller	1.5
LAHC-SA-DABC	1.625
Gogos	2.875
Atsuta et al.	4.125
Pillay	4.875

Table 15 Adjusted (Friedman) p values on the ITCC2007 datasets

Algorithm	Adjusted	PHolm	PHoch
LAHC-SA-DABC	0.8743	0.8743	0.8743
Gogos	0.0819	0.1639	0.1639
Atsuta et al.	8.9891E-4	0.0026	0.0026
Pillay	1.9628E-5	7.8514E-5	7.8514E-5

Table 15 presents the adjusted (Friedman) p value and the further results of the post hoc methods (Holm and Hochberg tests).

5.5 Box and whisker plots for the Toronto and ITC2007 datasets

Figures 6 and 7 show box and whisker plots for the Toronto and ITC2007 datasets, respectively, comparing the distribution percentiles for the LAHC-SA-DABC algorithm. Each box has lines at the lower quartile, the median and the upper quartile for the set of 11 runs.

The figures show less dispersion of the solution points, particularly for the upper and lower quartiles, in Figs. 6 (car91, car92, hec92I and uta92I datasets) and 7 (Exam_2, Exam_5 and Exam_6). The LAHC-SA-DABC algorithm generally finds good-quality solutions for both problems. In all the Toronto datasets (Fig. 6), the median lies between the

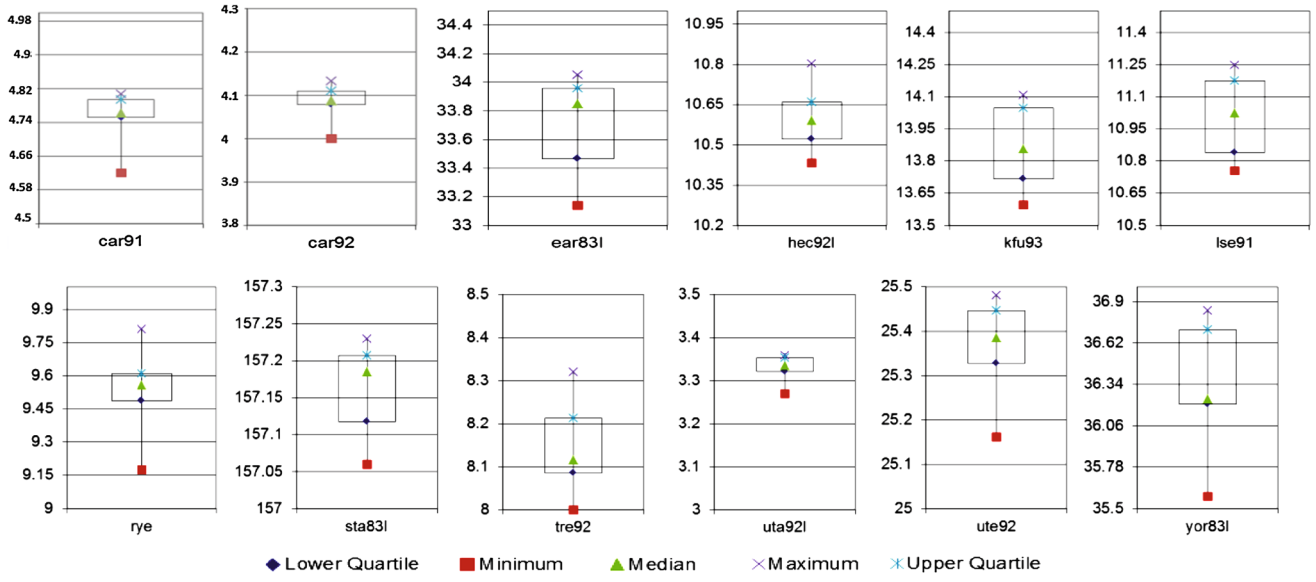

Fig. 6 *Box* and *whisker* plots obtained for LAHC-SA-DABC on the Toronto datasets

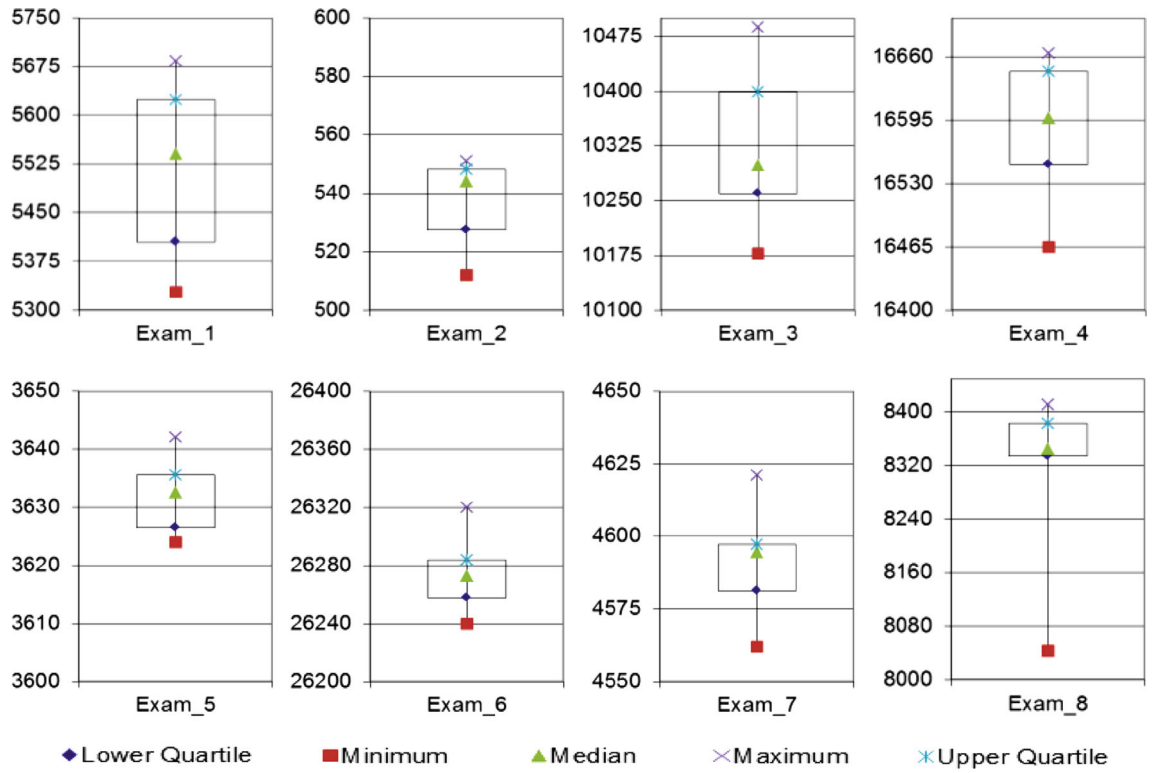

Fig. 7 *Box* and *whisker* plots obtained for the LAHC-SA-DABC algorithm on the ITC2007 datasets

best and worst runs, and their qualities are closer than for ITC2007, because of the complexity of the problem.

6 Conclusions and future work

The primary aim of this paper is to enhance the performance of the basic ABC algorithm by use of a disruptive selection strategy (DABC), a self-adaptive strategy for selecting neighbourhood structures and local search algorithms (late-acceptance hill climbing, in this case). The experimental results demonstrate that the LAHC-SA-DABC algorithm outperforms other modifications of the ABC algorithm and is comparable to state-of-the-art approaches when tested on examination timetabling problems. The experimental results also show that, with the disruptive selection strategy, the solutions in the population tend to converge together. In future research work, it is suggested to investigate the effect of these modifications on other categories of honeybee algorithms and to test their performance on a broader range of timetabling problems.

References

Abdullah, S., & Burke, E. K. (2006). A multi-start large neighbourhood search approach with local search methods for examination timetabling. In D. Long, S. F. Smith, D. Borrajo, & L. McCluskey (Eds.), *International Conference on Automated Planning and Scheduling (ICAPS 2006), Cumbria, UK, 6–10 June* (pp. 334–337).

Abdullah, S., Burke, E. K., & McCollum, B. (2007a). Using a randomised iterative improvement algorithm with composite neighbourhood structures for university course timetabling. In *Metaheuristics: Progress in Complex Systems Optimization (Operations Research/Computer Science Interfaces Series)* (Chap. 8, pp. 153–169). New York: Springer.

Abdullah, S., Ahmadi, S., Burke, E. K., & Dror, M. (2007b). Investigating Ahuja-Orlin's large neighbourhood search approach for examination timetabling. *OR Spectrum*, *29*(2), 351–372.

Abdullah, S., Turabieh, H., & McCollum, B. (2009). A hybridization of electromagnetic like mechanism and great deluge for examination timetabling problems. In *HM2009: 6th International Workshop on Hybrid Metaheuristics, Udine* (pp. 60–72).

Alzaqebah, M., & Abdullah, S. (2011a). Comparison of the selection strategy in the artificial bee colony algorithm for examination timetabling problems. *International Journal of Soft Computing and Engineering*, *1*(5), 158–163.

Alzaqebah, M., Abdullah, S. (2011b). Hybrid artificial bee colony search algorithm based on disruptive selection for examination timetabling problems. In *International Conference on Combinatorial Optimization and Applications (COCOA 2011)*. LNCS (Vol. 6831, pp. 31–45). Berlin: Springer-Verlag.

Atsuta, M., Nonobe, N., & Ibaraki, T. (2007). *ITC2007 Track 1: An Approach using general CSP solver*. http://www.cs.qub.ac.uk/itc2007.

Bao, L., & Zeng, J. (2009). Comparison and analysis of the selection mechanism in the artificial bee colony algorithm. *HIS*, *1*, 411–416.

Baykasoglu, A., Ozbakir, L., & Tapkan, P. (2007). Artificial bee colony algorithm and its application to generalized assignment problem. In

T. S. C. Felix, & K. T. Manoj (Eds.), *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization* (pp. 113–143). Vienna: Itech Education and Publishing.

Burke, E. K., & Bykov, Y. (2008). A late acceptance strategy in hill-climbing for exam timetabling problems. In *Conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*.

Burke, E. K., & Bykov, Y. (2012). *The late acceptance hill-climbing heuristic, technical report CSM-192*. Stirling: Computing Science and Mathematics, University of Stirling.

Burke, E. K., & Newall, J. P. (2004). Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of Operations Research*, *129*, 107–134.

Burke, E. K., Bykov, Y., Newall, J. P., & Petrovic, S. (2004). A time-predefined local search approach to exam timetabling problem. *IIE Transactions*, *36*(6), 509–528.

Burke, E. K., Eckersley, A. J., McCollum, B., Petrovic, S., & Qu, R. (2010). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operation Research*, *206*(1), 46–53.

Burke, E. K., Elliman, D. G., Ford, P. H., & Weare, R. F. (1996). Examination timetabling in British universities: A survey. In E. K. Burke & P. Ross (Eds.), *Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference* (Vol. 1153, pp. 76–90). Lecture Notes in Computer Science. Berlin: Springer-Verlag.

Caramia, M., Dellolmo, P., & Italiano, G. F. (2009). Novel local search-based approaches to university examination timetabling. *INFORMS Journal on Computing*, *20*(1), 86–99.

Carter, M. W. (1986). A survey of practical applications of examination timetabling algorithms. *Operations Research*, *34*(2), 193–202.

Carter, M. W., Laporte, G., & Lee, S. Y. (1996). Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, *47*, 373–383.

Chu, S. C., Chen, Y. T., & Ho, J. H. (2006). Timetable scheduling using particle swarm optimization. In *First International Conference on Innovation Computing, Information and Control* (pp. 324–327). Washington, DC: IEEE Computer Society.

Cooper, T. B., & Kingston, J. H. (1995). The complexity of timetable construction problems. In *Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling (PATAT 1995)*. Lecture Notes in Computer Science (Vol. 1153, pp. 283–295). New York: Springer-Verlag.

Dowsland, K. A., & Thompson, J. (2005). Ant colony optimization for the examination scheduling problem. *Journal of Operational Research Society*, *56*, 426–438.

Garcia, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, *180*(10), 2044–2064.

Garcia, S., Molina, D., Lozano, M., & Herrera, F. (2009). A study on the use of non-parametric tests for analysing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics*, *15*, 617–644.

Gogos, C., Alefragis, P., Housos, E. (2008). A multi-staged algorithmic process for the solution of the examination timetabling problem. In *Practice and Theory of Automated Timetabling (PATAT 2008), Montreal* (pp. 19–22).

Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Kang, F., Li, J., & Xu, Q. (2009). Structural inverse analysis by hybrid simplex artificial bee colony algorithms. *Computers and Structures*, *87*, 861–870.

Karaboga, N., (2005). *An idea based on honey bee swarm for numerical optimization*. Technical Report TR06. Engineering Faculty, Computer Engineering Department, Erciyes University, Turkey.

Karaboga, N. (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. *Journal of the Franklin Institute*, *346*(4), 328–348.

Karaboga, N., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, *1*(214), 108–132.

Karaboga, N., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*, 459–471.

Karaboga, N., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, *8*, 687–697.

Kuo, T., & Huang, S. Y. (1997). Using disruptive selection to maintain diversity in genetic algorithms. *Applied Intelligence*, *7*(3), 257–267.

Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, *30*(1), 167–190.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A., . . . Burke, E. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, *22*, 120–130.

Muller, T. (2009). ITC2007 solver description: A hybrid approach. *Annals of Operation Research*, *172*(1), 429–446.

Pan, Q. K., Tasgetiren, M. F., Suganthan, P. N., & Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences*, *181*(12), 2455–2468.

Pham, D. T., Ghanbarzadeh, A., Koc, E., & Otri, S. (2006). Application of the bees algorithm to the training of radial basis function networks for control 213 chart pattern recognition. In *5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering, CIRP ICME, Ischia, Italy* (pp. 711–716).

Pillay, A. (2007). *Developmental approach to the examination timetabling problem*. http://www.cs.qub.ac.uk/itc2007.

Qu, R., Burke, E. K., McCollum, B., & Merlot, L. T. G. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, *12*(1), 55–89.

Sabar, N. R., Ayob, M., & Kendall, G. (2009). Solving examination timetabling problems using honey-bee mating optimization (ETP-HBMO). In *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2009), Dublin, Ireland* (pp. 399–408).

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, *13*(2), 87–127.

Singh, A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied Soft Computing*, *9*, 625–631.

Turabieh, H. & Abdullah, S. (2011a). A hybrid fish swarm optimisation algorithm for solving the examination timetabling problems. In *Learning and Intelligent Optimisation Workshop (LION 5), Rome*. Lecture Notes in Computer Science (Vol. 6683, pp. 539–551). Berlin: Springer-Verlag.

Turabieh, H., & Abdullah, S. (2011b). An integrated hybrid approach to the examination timetabling problem. *OMEGA–The International Journal of Management Science*, *39*(6), 598–607.

Yang, Y., & Petrovic, S. (2005). A novel similarity measure for heuristic selection in examination timetabling. In E. K. Burke & M. Trick (Eds.), *Practice and Theory of Automated Timetabling V: Selected Papers from the 5th International Conference* (Vol. 3616, pp. 377–396). Lecture Notes in Computer Science. Berlin: Springer.