Applied Soft Computing 13 (2013) 3608–3620

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

A hybrid self-adaptive bees algorithm for examination timetabling problems

Salwani Abdullah*, Malek Alzaqebah

Data Mining and Optimisation Research Group, Center for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

ARTICLE INFO

Article history:
Received 3 August 2012
Received in revised form 4 January 2013
Accepted 11 April 2013
Available online 25 April 2013

Keywords:
Bees algorithm
Examination timetabling problems
Late acceptance hill climbing algorithm
Selection strategy
Self-adaptive mechanism
Simulated annealing algorithm

ABSTRACT

A hybrid self-adaptive bees algorithm is proposed for the examination timetabling problems. The bees algorithm (BA) is a population-based algorithm inspired by the way that honey bees forage for food. The algorithm presents a type of neighbourhood search that includes a random search that can be used for optimisation problems. In the BA, the bees search randomly for food sites and return back to the hive carrying the information about the food sites (fitness values); then, other bees will select the sites based on their information (more bees are recruited to the best sites) and will start a random search. We propose three techniques (i.e. disruptive, tournament and rank selection strategies) for selecting the sites, rather than using the fitness value, to improve the diversity of the population. Additionally, a self-adaptive strategy for directing the neighbourhood search is added to further enhance the local intensification capability. Finally, a modified bees algorithm is combined with a local search (i.e. simulated annealing, late acceptance hill climbing) to quickly descend to the optimum solution. Experimental results comparing our proposed modifications with each other and with the basic BA show that all of the modifications outperform the basic BA; an overall comparison has been made with the best known results on two examination timetabling benchmark datasets, which shows that our approach is competitive and works well across all of the problem instances.

1. Introduction

Examination timetabling problems (ETTPs) are combinatorial optimisation problems that consist of allocating a number of examinations into a predefined number of timeslots, while satisfying a set of hard constraints that cannot be violated and soft constraints that must be minimised as much as possible [1]. It is known that the ETTPs fall under the *NP-hard* problems [2,3]. Furthermore, it is a dynamic and perturbed problem. Many contributions related to the ETTPs have appeared for the last thirty years. This scenario could result from the fact that ETTPs are often dynamic problems, and the optimisation criteria are difficult to identify.

Several meta-heuristic approaches have been developed for solving ETTPs which can be classified into two main types, i.e. single-based approaches (e.g. tabu search, simulated annealing, great deluge and variable neighbourhood search) and population-based approaches (e.g. genetic algorithms, ant colony optimisation and memetic algorithms) [4]. Single-based approaches have gained interest by many researchers due to the ability of these approaches to exploit the search space in a short time, but these approaches have some limitations such as a weak exploration and it is easy to get stuck in a local optima [4]. In addition, researchers have introduced population-based approaches to solve the examination timetabling problems. The main idea behind the population-based is that the algorithms iteratively improve a number of solutions [5]. However, these approaches have some limitations such as they are more concerned with exploration rather than exploitation, premature convergence and low convergence speed [5]. To overcome the limitations of the single-based and population-based approaches, the hybridisation between population-based approaches with a single-based approach has been addressed for timetabling problems. The aim of the hybridisation is to utilise the benefit of population-based approaches that has the ability of identifying promising areas in the search space and single-based approaches that are good in exploiting the promising area [6–9]. It is believed that the hybridisation approach is able to give a better performance in obtaining a preferred solution for a given problem [9].

Population-based approaches can be categorised as either evolutionary algorithms or swarm intelligence based algorithms [10]. These two categories depend on the nature of the phenomenon simulated by the algorithm. Most common evolutionary algorithms introduced for timetabling problems can be found in [11–14]. Swarm intelligence relies on the cooperative behaviour of

* Corresponding author. +60 389216183.
E-mail addresses: salwani@ftsm.ukm.my, drsalwaniabdullah@gmail.com (S. Abdullah), malek.zaqeba@gmail.com (M. Alzaqebah).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.04.010

self-organised systems to develop meta-heuristics that mimic such a system's problem solving [15]. Local communication between individuals and with their environment contributes to the collective intelligence of the social colonies [10]. These swarm intelligence characteristics motivated a number of researchers to employ such behaviour in algorithms for timetabling problems (e.g. ant algorithms, fish swarm optimisation algorithm, honey-bee mating optimisation algorithm and artificial bee colony algorithm).

Eley [16] proposes ant algorithms for the examination timetabling problem. The author investigated two approaches which were coded as Max–Min and ANTCOL. In both approaches the hybridisation between a simple ant system and hill climber was made. The initial solutions were constructed based on the inverse of saturation degree meta-heuristic. The author found that the simple ANTCOL outperformed the hybrid Max–Min Ant system with hill climber. The author also found that the parameters sitting in ant system much affects the performance of such approaches. This approach is good in discovering new solutions and able to avoid the premature convergence, the drawback are, it is reported as a slow algorithm, Involves a number of parameters and Not effective as an improvement algorithm. Turabieh and Abdullah [17] proposed the fish swarm optimisation algorithm for examination timetabling problems. In this work, the solutions in the population are classified into three groups based on the solution quality. In addition, two local search algorithms are used in order to enhance the solutions quality i.e. great deluge and steepest descent algorithms. In order to reduce the computational time of the original fish swarm algorithm, the authors modified the algorithm by selecting only one solution for exploitation based on a roulette wheel selection rather than exploit all the solutions in the population. This approach shows that it's good in exploration and diversification but not good in intensification. Sabar et al. [18] proposed honey-bee mating optimisation algorithm for solving examination timetabling problems. As a result of their approach has showed that it is able to explore and exploit the search space. The limitation of their approach is the number of parameters that should be set carefully. Alzaqebah and Abdullah [19] proposed a hybrid artificial bee colony with simulated annealing algorithm, in this work the authors have investigate the use of disruptive selection strategy and a self-adaptive mechanism for examination timetabling.

The bees algorithm (BA) is based on the foraging behaviour of honeybees and was proposed in 2005 by Pham et al. [20]. Pham et al., in 2006, proved that BA outperforms other techniques in terms of the speed of the optimisation and the accuracy of the results [21]. With our research, we first seek to improve the local search efficiency of a BA by applying a local search algorithm in order to achieve the following: combine the advantages of BA, improve the ability to cooperatively explore the search space and local search algorithms, improve the ability to quickly find good solutions within a small region of the search space, and prevent deterioration. Second, we enhance the population diversity in the BA population, and third, we introduce a self-adaptive method for neighbourhood search to manage the neighbouring search. Experimental results show that the algorithm behaviour improves when we employ the three modifications that are stated above. Meanwhile, all of the modification comparisons are listed and analysed to show the effects that are produced by our modifications. The comparison with the best known results indicates that our results are comparable.

The main contribution to the BA in this work is, first, maintaining a population of individuals that evolve according to the nature of a disruptive selection strategy. The second is to avoid the algorithm getting *stuck* (no available improving neighbours) at "*local optima*" by performing a self-adaptive mechanism that monitors the neighbourhood search. The third contribution is to support the BA neighbourhood search with a novel local search algorithm (late acceptance hill climbing algorithm).

The details of the ETTP benchmarks are given in Section 2, which is followed by the details of the original BA (Section 3). Section 4 presents the proposed modifications to the original BA. Section 5 presents the proposed BA algorithm. Section 6 presents our experimental results analysis and comparisons. This section is followed by conclusions and comments in Section 7.

2. Examination timetabling problems

In this paper, the BA is tested in two examination timetabling problems, as listed below:

2.1. The Toronto benchmark

This problem is considered to be an uncapacitated examination timetabling problem, where a room capacity requirement is not taken into account. The *Toronto benchmark* was introduced by Carter et al. [22] in 1996. This problem has one hard constraint (*clash-free*), where conflicting exams cannot be assigned to the same timeslot. A timetable which meets all the hard constraints given is called a feasible timetable. The soft constraint spreads conflicting exams away from each other during the timetable as much as possible. The description of the problem is adapted from Burke and Newall [23]. In this benchmark, the examination timetabling problems consist of these inputs, as stated below:

- N is the number of exams.
- E_i is an exam, $i \in \{1,\ldots,N\}$.
- T is the given number of available timeslots.
- M is the number of students.
- $C = (c_{ij})_{NxN}$ is the conflict matrix, where each element denoted by c_{ij}, $ij \in \{1,\ldots,N\}$ is the number of students taking exams i and j.
- t_k $(1 \le t_k \le T)$ specifies the assigned timeslot for exam k $(k \in \{1,\ldots,N\})$.

The objective function has been formulated to space out students' exams throughout the exam period (Eq. (1)), which can then be formulated as the minimisation of the following [24].

$$Min \frac{\sum_{i=1}^{N-1} F_1(i)}{2M} \tag{1}$$

where

$$F_i(i) = \sum_{j=i+1}^{N} C_{ij} \cdot proximity \quad (t_i, t_j) \tag{2}$$

and

$$proximity(t_i, t_j) = \left\{ \begin{array}{cc} 2^5/2^{|t_i - t_j|} & \text{if} \quad 1 \le |t_i - t_j| \le 5 \\ 0 & \text{otherwise} \end{array} \right\} \tag{3}$$

subject to:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C_{ij} \cdot \gamma(t_i, t_j) = 0 \quad \text{where} \quad \gamma(t_i, t_j) = \left\{ \begin{array}{cc} 1 & \text{if} \quad t_i = t_j \\ 0 & \text{otherwise} \end{array} \right. \tag{4}$$

Eq. (2) presents the cost for an exam I, which is given by the proximity value multiplied by the number of students in conflict. Eq. (3) represents a proximity value between two exams [22]. For example, if a student has two consecutive exams, then a proximity value of 16 (i.e. $2^5/2^1$) is assigned. If a student has two exams with a free timeslot in between then a value of 8 (i.e. $2^5/2^2$) is assigned. The value will be 4 if there are two timeslots in between and so on. These values are summed up and divided by $2M$, to give an average

Table 1
Characteristics of Toronto benchmark dataset (Toronto dataset).

Datasets	Number of timeslots	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	1125	0.27
hec92 I	18	81	2823	0.42
kfu93	20	461	5349	0.06
lse91	18	381	2726	0.06
pur93 I	42	2419	30,032	0.03
rye92	23	486	11,483	0.07
sta83 I	13	139	611	0.14
tre92	23	261	4360	0.18
uta92 I	35	622	21,267	0.13
ute92	10	184	2750	0.08
yor83 I	21	181	941	0.29

penalty per student (Eq. (1)) since all clashes would otherwise be counted twice (e.g. c_{12} and c_{21}).

Eq. (4) represents a *clash-free* requirement so that no student is asked to sit in two exams at the same time. The clash-free requirement is considered to be a hard constraint. Table 1 presents the characteristics of this dataset.

2.2. The International Timetabling Competition (ITC2007) dataset

ITC2007 introduces three tracks of problems, examination timetabling, curriculum-based course timetabling and post-enrolment course timetabling. In this paper, we concentrate on the first track, which represents the examination timetabling problems that include a number of real-world constraints [25]. Details of the benchmark instances can be found in [26]. Eight instances are available and four are hidden for testing purposes. Table 2 shows the characteristics of these datasets.

A feasible timetable is one in which all of the examinations have been assigned to a period and a room, and there is no violation of the hard constraints. The set of hard constraints are listed below:

- There are no students sitting for more than one exam at the same time.
- The total number of students assigned to each room cannot exceed the room capacity.
- The length of the exams that are assigned to each timeslot should not violate the timeslot length.
- The exam sequences must be respected; for example, *Exam_A* must be scheduled after *Exam_B*.
- Room-related hard constraints must be satisfied; for example, *Exam_A* must be scheduled in *Room2*.

In addition, the objective function is to minimise the violation of the soft constraints, as given in Eq. (5) [25]. Each dataset has its own weight (W) as shown in Table 3.

Table 2
Characteristics of ITC 2007 examination datasets (ITC2007 datasets).

Dataset	D1	D2	D3	D4	D5	D6	CD
Exam_1	7833	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	4421	273	21	1	40	0	15.0
Exam_5	8719	1018	42	3	27	0	0.87
Exam_6	7909	242	16	8	23	0	6.16
Exam_7	13,795	1096	80	15	28	0	1.93
Exam_8	7718	598	80	8	20	1	4.55

Note. D1, number of students; D2, number of exams; D3, number of timeslots; D4, number of rooms; D5, period hard constraints; D6, room hard constraints; CD, conflict density.

Table 3
Associated weight of the ITC2007 dataset.

Datasets	W^{2D}	W^{2R}	W'	W^{NMD}	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

$$\min \sum_{s \in S} (W^{2R}C_S^{2R} + W^{2D}C_S^{2D} + W^{pS}C_S^{pS}) + (W^{NMD}C_S^{2NMD} + W^{FL}C^{FL} + W^{p}C^{p} + W^{R}C^{R}) \quad (5)$$

The soft constraints (C) are listed below [25]:

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

3. The bees algorithm

In the natural life of the honeybee, bees can discover food sources using global and local search methods. The former consists of sending scout bees to search at random around the hive for food sources. Once the scout bees discover food sources, they determine the food source information, including the locations of the food, the food quality and the distance from the hive, and return back to their hive and start recruiting more bees to exploit those food sources [27]. Logically, more recruited bees will be sent to search around the better food sources, where they perform a search similar to a local search. To ensure that the search continues until good food sources are found, including the best, scout bees would continue on a global random search, while some bees are recruited to perform a local search. This intelligent optimisation process has motivated researchers to formulate such a process into an algorithmic form, as in the basic BA [20]. The applications of the BA algorithm are many and varied in the optimisation field [20,21,28–30].

Fig. 1 shows the main steps of the basic Bees Algorithm (BA), as stated in [21]. The algorithm starts with an Initial population (*scout bees*) that is generated randomly in the search space; then, the fitness of the population is evaluated, and the BA search process is started until the stopping criterion is met.

In the first step, the bees are ranked from the highest to the lowest fitness; the highest bees are designated as "*selected bees*", and the sites visited by the selected bees are chosen for a

```
Initialise population with random solutions.
Evaluate fitness of the population.
While (stopping criterion not met)
     //Forming new population.
     Select sites for neighbourhood search.
     Recruit bees for selected sites (more bees for best e
     sites) and evaluate fitness.
     Select the fittest bee from each patch.
     Assign remaining bees to search randomly and evaluate their
     fitness.
End While.
```

Fig. 1. Original bees algorithm.

```
for i=1:n
     a_i ← 0;
    for j=1:n
      if f_i ≤ f_j
         a_i = a_i +1;
      end if
    end for
endfor
```

Fig. 2. Tournament selection pseudo code.

neighbourhood search (selected solutions). In the second step, more bees are assigned to search near the selected sites. Searching nearby to the best sites shows that, more promising solutions are made more detailed by recruiting more bees to search near the selected sites compared to other sites. Together with scouting, this scenario is a key operation of the bees algorithm [21,30]. However, in the third step, for each patch, only the bee with the highest fitness will be selected for the next population. This constraint is initiated here to maintain the same population size. In the fourth step, the remaining bees in the population are assigned randomly around the search space, scouting for new solutions. The new population after each iteration will have two parts; the first part is from each selected patch, and the second part is from conducting a random search.

4. Proposed modifications on the original BA algorithm

4.1. Selection strategy

The selection strategies are used in many population based algorithms, over the year a number of research papers have studied different selection strategies in order to improve the algorithms' performance, e.g. artificial bee colony (ABC) [31,19] algorithm and genetic algorithm (GA) [32]. Karaboga and Basturk [31] introduced ABC algorithm for solving constrained optimisation problems. In this work, authors have modified the ABC algorithm by changing from the greedy selection (the highest quality solution is selected) to a tournament selection strategy. As a result, the modified version of ABC is able to solve constraint optimisation problems. Also in [19], ABC algorithm is modified by applying three selection strategies (i.e. disruptive, rank and tournament) with an aim is to compare the performance of the ABC algorithm with different selection strategies. Experimental results have shown that the ABC with three selection strategies performs better than the original ABC, where the disruptive selection shows the best performance in comparison with the rank and the tournament selections.

In GA, the selection strategy is used to select the parents for crossover operation. Jadaan et al. [33] introduce a comparison between roulette wheel and rank-based selection in comparing several mathematical fitness functions. Experimental results have shown that the rank-based selection strategy outperformed the roulette wheel selection strategy. The observation shows that the rank-based is steadier, faster, certainty and more robust. Zhong et al. [34] compared a roulette wheel with a tournament selection on seven test functions. The authors have found that the GA with the tournament selection is more efficient in convergence than the roulette wheel selection. Kuo and Hwang [35] introduced the use of disruptive selection in order to maintain the diversity in genetic algorithms.

In this work, three selection strategies (tournament, rank and disruptive) are incorporated into the BA algorithm and are tested on the examination timetabling problem, as below:

4.1.1. Tournament selection

In tournament selection, a comparison is made depending on the fitness, to take the best individual among a number of individuals (N_{tour}) that are chosen randomly from the population. Parameter N_{tour} is called the tournament size. In tournament selection, the individuals with a high fitness are more preferable [36]. In this work, we conceder a binary tournament in which we select two individuals from the population and compare their fitness values, then assign one score (coded as a) to the better individual. This process is then repeated for all of the individuals in the population, as shown in Fig. 2, where f_i is the fitness value of $i = 0,\ldots n$, and n is the population size adapted from [37]. Finally, the selection probability for each individual is calculated using Eq. (6).

$$P_i = \frac{a_i}{\sum_{j=0}^{n} a_j} \tag{6}$$

4.1.2. Rank selection

In rank selection, individuals are sorted in descending order based on the fitness value. The index k is given to each individual from the best to the worst, i.e. for the best fitness $k = 1$, and for the worst fitness, $k = n$, where n is the population size, and N is the maximum number of iterations. Finally, the selection probability is calculated using the expression below [38]:

$$P_k = \frac{1}{n} + a(t)\frac{n+1-2k}{n(n+1)}, \quad k = (1, 2, \ldots, n) \tag{7}$$

$$\text{where} \quad a(t) = 0.2 + \frac{3t}{4N}, \quad t = (1, 2, \ldots, N)$$

4.1.3. Disruptive selection

Disruptive selection gives more chances for higher and lower individuals to be selected, by changing the definition of the fitness function, as in Eq. (8) [39]. Disruptive selection changes the fitness value for each individual (fit_i) in the population to an absolute value of the difference between an individual's fitness (f_i) and the average value for all individuals ($\bar{f}_t$) in the population.

$$fit_i = |f_i - \bar{f}_t| \quad P_i = \frac{fit_i}{\sum_{i=0}^{n} fit_i} \tag{8}$$

There is a trend in the disruptive selection to keep the diversity longer because retaining the highest and lowest quality solutions is more preferable.

4.2. Self-adaptive method for neighbouring search

Neighbourhood searches use a self-adaptive mechanism to guide the neighbouring search process. We employed the following neighbourhood structure to increase the searching algorithm performance [40].

Nbs1: Select 2 exams randomly and swap timeslots.
Nbs2: Select a single exam randomly and move it to a new random feasible timeslot.
Nbs3: Select 4 exams randomly and feasibly swap timeslots among them.
Nbs4: Select 2 exams randomly and move them to random feasible timeslots.

The neighbourhood structures are selected based on the self-adaptive method, as discussed below [41]:

1. This method fills a list (*NL*) randomly from four neighbourhood structure search methods, as above. In this step, the list (*NL*) is generated with a specified neighbour lengths.
2. During the search process, the neighbourhood structure is selected from *NL* to generate a new solution. If it is better than the current solution, then the neighbourhood structure is inserted into a new list, called a winning neighbouring list (*WNL*). The same process is repeated until *NL* becomes empty.
3. *NL* is refilled by 75% of the neighbourhood structures from the *WNL* list and 25% randomly from the four neighbourhood structure. Note that the idea of filling the *NL* list is adopted from [41] where we used the same number of neighbourhood structures. The percentages of 75% and 25% used here are based on our preliminary experiments; the *WNL* is also reset to repel any accumulation. If the *WNL* is empty, which may occur when a search is close to optimal with a negligible population variety, then the last *NL* is reused. Note that the same percentage of *NL* and *WNL* can be used in other applications since BA is not sensible to them.

With a self-adaptive method, the suitable neighbouring search operation can be learned based on the current search process where the wining neighbourhood structure during the search is kept and reused. The chance of using the wining neighbourhood structures is higher than others due to the fact that the *NL* list is filled by 75% of neighbourhood structures taken from the *WNL* and 25% is randomly filled. In this paper, we performed a tuning on the length of *NL* and the final length is set to 200.

4.3. Local search algorithms

We propose two local search algorithms, simulated annealing and late acceptance hill climbing, to improve the local search of the basic BA. The basic BA accepts the improved solution and eliminates the worst solution. These two local search algorithms are discussed and explained below:

4.3.1. Simulated annealing algorithm (SA)

Kirkpatrick et al. [42] proposed simulated annealing for combinatorial optimisation problems. The SA algorithm works on a single solution and improves it by finding neighbouring solutions and slowly modifying a parameter called temperature (*Temp*) [43]. The acceptance criteria of the neighbouring SA solutions depend on a certain probability (*P*) between 0 and 1. If *P* is less than the value of $e^{-\delta/Temp}$, where $\delta = f(Sol^*) - f(Sol)$ (the difference between the penalty cost of the new and current solutions), then the algorithm accepts the new solution, even if it is worse than the current solution; otherwise, the algorithm accepts the new solution only if it is better than the current solution. The process repeats until the *Temp* is less than the final temperature T_f, as in Fig. 3.

The parameters used for the SA algorithm are set based on experimental results, as in Section 6.3.

4.3.2. Late acceptance hill climbing algorithm (LAHC)

LAHC also applies to the basic BA algorithm to compare different local searches in the BA algorithm. The LAHC algorithm is a local search algorithm that was proposed by Burke and Bykov [44]. The LAHC algorithm works with a single solution and iteratively improves it. This algorithm keeps the fitness values after each single move in a memory ($\hat{C}$ = LAHC list) with size (*L*). Accepting the new solution in LAHC relies on a comparison between the new and previous solutions that were obtained in the *L*th step. Fig. 4 shows the late acceptance hill climbing pseudo-code.

```
Set the working Solution Sol;
Set initial temperature T0;
Set final temperature Tf;
Set number of iteration NumOfIteSA;
Set decreasing temperature rate as α
where α = (log(T0) - log(Tf))/NumOfIteSA;
Set Temp ← T0;
Set SolbestSA ← Sol;
Set SolSA ← Sol;
do while (Temp > Tf)
  SolSA* ← Apply neighbourhood structure on SolSA;
  Calculate cost function f(SolSA*);
   if (f(SolSA*) < f(SolbestSA))
      SolSA ← SolSA*;
      SolbestSA ← SolSA*;
   else
       Generate a random number, RandNum;
       if(RandNum ≤ e^(-δ/Temp)) where δ = f(SolSA*)-f(SolSA)
          SolSA ← SolSA*;
       end if
   end if
   Temp = Temp-Temp*α;
end while
```

Fig. 3. Simulated annealing pseudo-code.

5. A hybrid self-adaptive bees algorithm

The general pseudo-code for our proposed algorithm is presented in Fig. 5. The general pseudo-code shows the employment of the three modifications mentioned in Section 4, where we linked each modification with the section number within the pseudo-code.

As shown in Fig. 5, the algorithm starts with a feasible initial solution; then, an initial population is randomly generated, and the number of solutions in the population is equal to the number of scout bees. The scout bees evaluate the solution using the fitness function, which we explain in Section 2.

```
Set the working Solution Sol;
s ← Sol;
Ĉ: LAHC List;
Set number of iteration, itr;
I= number iteration;
L= the length of the list;
Calculate initial cost function C(s)
for all k ∈ {0...L-1} do Ĉk ← C(s)
I ← 0;
do while (I > itr)
 s*← Apply neighbourhood structure ons*
 Calculate its cost function C(s*)
 v ← I mod L
 if C(s*) ≤ Ĉv
     accept candidate (s ← s*)
     Ĉv ← C(s)//Insert cost value into Ĉv list
 End if
 I ← I+1;
end do
```

Fig. 4. LAHC pseudo-code.

```
Initialisation:
Initialise the initial population and evaluate the fitness;
Calculate the initial fitness value, f(Sol);
Set best solution, Solbest ← Sol;
Set maximum number of iteration, NumOfIte;
 Set the population size;
Set ne:  number of elite sites;
Set nre: recruited bees for elite sites;
Set nb: number of best sites;
Set nrb: recruited bees for remaining best sites;
Set stlim: limit of stagnation cycles for site abandonment;
iteration ←   0;
Improvement:
do while (iteration < NumOfIte)
   for i=1: populationsize
       Scout bees evaluate the solutions;
       Calculate the selection probability Pi, based on the three
       selection stratigy as in section (4.1);
       Rank the solutionsbased on Pi (For basic BA use the objective
       function to rank the solutions)
    end for
    nbSet ← Select the top nb solutions from the population;
    neSet ← Select the top ne solutions from the nbSet;
   for i=1: nrb
      for j=1: nb
        Sol*  ← neSetj;
        Apply neighborhood structure on Sol* using self-adaptive
        method as in section (4.2)(For the basic BA use random
        neighborhood structure);
        Update the population by the Improved solutions;
       end for
   end for
   for j=1: nre
      for h=1: nrb
         Sol**  ← neSetj;
          Start local search on Sol**;
          Choose one local search from section (4.3)
          end local search
       end for
   end for
     Solbest ← best solution found so far;
    recruit the remaining bees for random search and update the
    population;
    iteration++;
end do
```

Exploration Process

Exploration Process

Fig. 5. The pseudo code for the bees algorithm.

Scout bees rank the solutions in the population according to the selection probability, as in Section 4.1 (for the basic BA, the solutions are ranked based on the fitness value), and the *nb* solutions of highest rank are selected for local exploration by other bees (foragers) that are directed to the neighbourhood of the selected sites by the scouts. For each selected solution, the number of foragers is allocated deterministically as follows. Each scout that returned from one of the *nb* best sites performs the '*waggle dance*', which recruits *nest* mates for local exploration (or applies one local search, as in Section 4.3). The scout bees that visited the first *ne* elite solutions among the best *nb* sites recruit *nre* foragers for a random neighbourhood search (or using the self-adaptive method, as in Section 4.2). The scouts that visited the remaining (*nb*–*ne*) solutions recruit *nrb* < *nre* foragers for a random neighbourhood search (or using the self-adaptive method). The local search is thus more thorough in the neighbourhood of the elite sites, which are considered to be the most promising locations of the solution space.

6. Experimental results and comparison

To choose suitable modifications for the basic BA algorithm, three experimental comparisons were made for the two examination timetabling problem datasets, as explained in Section 2. We first compared using three selection strategies; we then compared using two local search algorithms, which include some parameter tuning. We also considered the effect of using the self-adaptive method in neighbourhood searches. Finally, we compared our results to the best known results in the literature.

In this paper, the parameters are experimentally chosen from a total of 5 runs to obtain the average results. Furthermore, we have performed good tuning on the parameters for the algorithm, and the final settings of the parameters are as follows:

- Population size = 50.
- Maximum number of iterations = 500.
- Population size = 50.
- *ne*: Number of elite sites = 2.
- *nre*: Recruited bees for elite sites = 30.
- *nb*: Number of best sites = 4.
- *nrb*: Recruited bees for remaining best sites = 10.
- *stlim*: Limit of stagnation cycles for site abandonment = 10.
- LAHC list size (L) = 5000.

6.1. Experimental comparison of selection strategies

To show the performance of the basic BA algorithm with different selection strategies, three selection strategies were implemented within the BA algorithm. The three different modified BA algorithms are the BA algorithm based on disruptive selection (DBA), rank selection (RBA) and tournament selection (TBA). We ran the experiments 10 times for each dataset. Tables 4 and 5 present comparisons for both problems, including the best and average results.

Comparing the DBA, RBA, TBA, and BA algorithms indicates that the performance of the BA algorithm with three selection strategies is better than the basic BA algorithm on both datasets. As a comparison with different selection strategies in the BA algorithm, DBA works better compared to the RBA and TBA algorithms. From Table 4, the DBA algorithm produces good results for 9 datasets out of the 12 datasets that are presented in bold, as well as in Table 5, in which the DBA algorithm produces good results for 6 datasets out of the total of 8. This result arises from the behaviour of the selection strategy, where the tournament selection randomly selects a number of solutions (N_{tour}) and compares them based on a probability. The solution with the highest fitness value will be chosen. In a rank selection, the solutions are ranked based on the fitness values; thus, this function is biased to work in a solution with a higher rank (i.e. a higher fitness), while the disruptive selection concentrates on both bad and good fitness solutions and attempts to maintain the diversity of the population by improving the worst fitness solutions in synchrony with high fitness solutions.

Fig. 6(a) and (b) represents the convergence of the sta83I and Exam_2 datasets, respectively, where the *x*-axis represents the number of solutions and the *y*-axis represents the penalty cost. These graphs show how the DBA, TBA and RBA spray the population at an initial stage (represented by the triangle symbol), and then, when the stop condition is met, the improved solutions are represented by the square symbol. From these figures, we can conclude that the DBA provides a chance for all of the solutions in the population to be improved and converge together. This result is evident in the fact that the plotted symbols are concentrated (not scattered) with each other, which represents the closeness of the solution qualities in the population. On the other hand, in TBA and RBA, the plotted symbols that represent the improved solution in the population are more scattered.

6.2. Experimental comparison on the self-adaptive method for the neighbourhood search

The DBA algorithm employed a self-adaptive method for neighbourhood searches. To show the significance of using this method in the DBA algorithm search process, Tables 6 and 7 show the self-adaptive DBA compared to the DBA algorithm.

The comparison shows that the self-adaptive DBA outperforms the DBA algorithm. Employing a self-adaptive method in a neighbourhood search to penalise unperformed neighbourhood structures used later in the search algorithm helps the search

Table 4
Comparison between the BA, DBA, RBA, and TBA algorithms on the Toronto dataset.

Datasets	BA	DBA		RBA		TBA	
	Best	Best	Avg.	Best	Avg.	Best	Avg.
car91	5.79	5.77	6.16	5.88	6.02	**5.67**	5.918
car92	4.76	**4.77**	5.18	4.83	4.9	4.78	4.9
ear83 I	38.93	37.61	38.7	**37.2**	38.76	37.67	38.87
hec92 I	11.64	11.47	11.97	**11.13**	11.81	11.64	12.15
kfu93	15.70	15.67	16.63	**15.32**	16.38	15.70	16.24
lse91	12.66	**12.42**	13.23	12.64	12.97	12.81	13.35
rye92	10.68	**10.52**	10.89	11.04	11.2	11.12	11.14
sta83 I	158.05	**157.37**	158.43	157.52	157.98	157.59	157.9
tre92	9.05	**8.99**	9.46	9.02	9.21	**9.00**	9.31
uta92 I	3.92	4.00	4.18	3.90	3.98	**3.88**	4.01
ute92	28.05	27.54	29.69	**26.90**	28.83	27.58	28.8
yor83 I	40.01	**39.18**	40.63	39.55	40.54	39.28	40.62

Table 5
Comparison between the BA, DBA, RBA, and TBA algorithms on the ITC2007 datasets.

Data sets	BA	DBA		RBA		TBA	
		Best	Avg.	Best	Avg.	Best	Avg.
Exam_1	6574	**6388**	6570.5	6394	6618.5	6394	6646
Exam_2	1417	**1357**	1489.2	1478	1539.7	1415	1507.7
Exam_3	12,404	**12,317**	12684.6	12,371	12,575	12,374	12659.6
Exam_4	19,625	**19,038**	19991.2	19,595	20130.6	19,709	20150.7
Exam_5	12,783	**9868**	10989.6	10,784	11,456	10,950	11537.3
Exam_6	27,090	26,730	26813.5	26,735	26,802	**26,665**	26,789
Exam_7	6771	6673	6999.6	**6500**	7167.5	6766	7106.7
Exam_8	11,655	**9833**	10193.2	9859	10278.2	10,132	10336.3

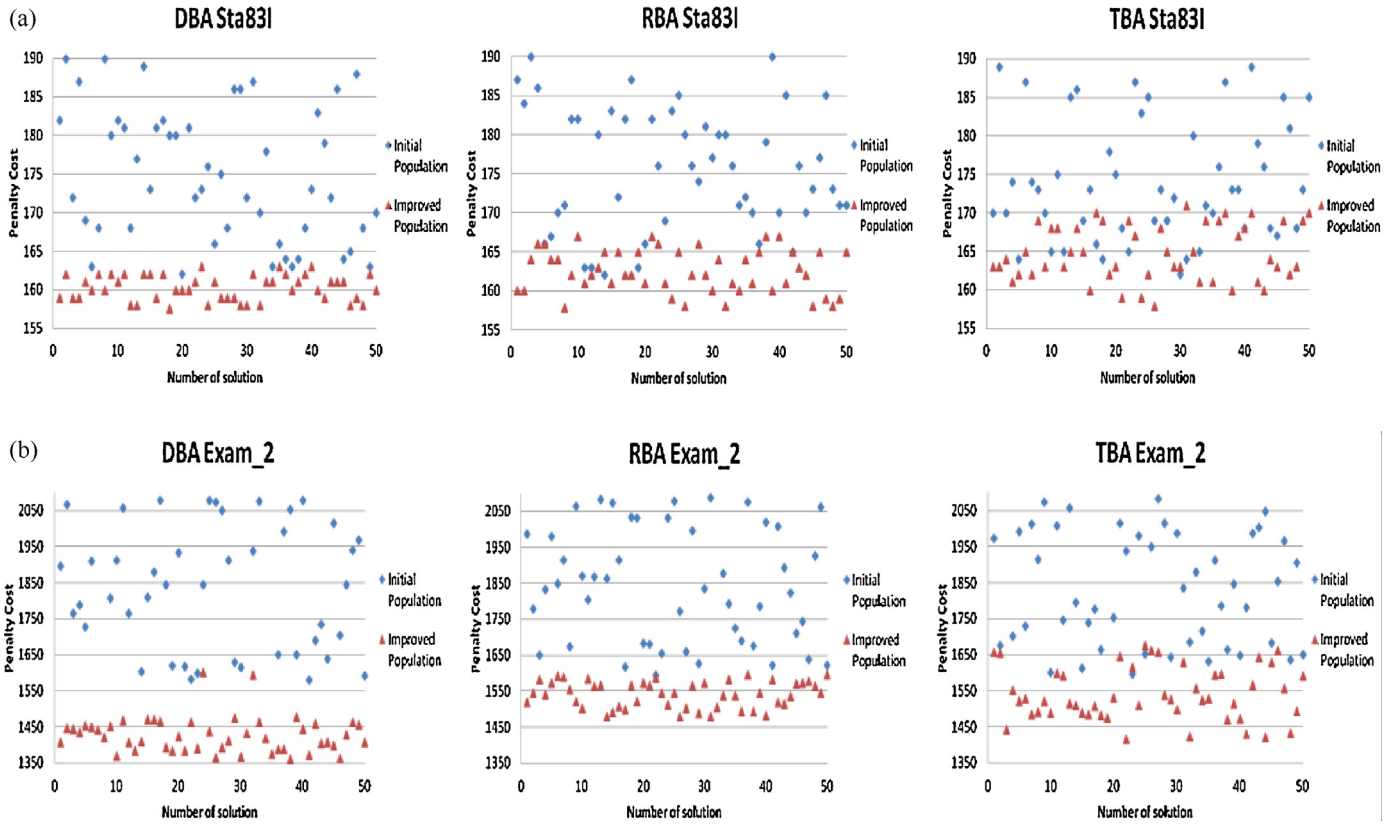

Fig. 6. Convergence of two datasets (a) sta83I and (b) Exam_2.

Table 6
Experimental results when applying self-adaptive on the DBA algorithm for the Toronto dataset.

Datasets	DBA		Self-adaptive DBA	
	Best	Avg.	Best	Avg.
car91	5.77	6.16	**5.68**	5.81
car92	4.77	5.18	**4.71**	4.80
ear83 I	37.61	38.7	**36.99**	38.25
hec92 I	11.47	11.97	**11.11**	11.46
kfu93	15.67	16.63	**15.00**	15.47
lse91	12.42	13.23	**12.38**	12.68
rye92	10.52	10.89	**10.05**	10.23
sta83 I	**157.37**	158.43	157.45	157.58
tre92	**8.99**	9.46	9.00	9.07
uta92 I	4.00	4.18	3.73	3.77
ute92	**27.54**	29.69	27.63	28.23
yor83 I	39.18	40.63	**38.97**	39.90

Table 7
Experimental results when applying self-adaptive on the DBA algorithm for the ITC2007 dataset.

Datasets	DBA		Self-adaptive DBA	
	Best	Avg.	Best	Avg.
Exam_1	**6388**	6570.5	6450	6670.4
Exam_2	1357	1489.2	**1142**	1415.9
Exam_3	12,317	12684.6	**11,956**	12340.3
Exam_4	**19,038**	19991.2	19,426	20513.8
Exam_5	9868	10989.6	**7111**	8330.8
Exam_6	26,730	26813.5	**26,625**	26733.5
Exam_7	6673	6999.6	**6664**	6946.8
Exam_8	**9833**	10193.2	10,163	10636.8

algorithm to explore the search space differently. This scenario allows for a higher possibility of obtaining better solutions.

6.3. Experimental comparison on different local search algorithms

From the previous section, we perform the BA disruptive selection strategy (coded as the DBA algorithm) because of its performance. In this section, we compare the two local search algorithms in the DBA algorithm.

6.3.1. Simulated annealing (SA) experimental comparison

Table 8 shows the experimental results when applying SA to a self-adaptive DBA algorithm, which is coded as the self-adaptive DBA_{SA}. We also used different values for the initial temperature (T_0), which was set to 5000, 2000, 1000 and 500. The results obtained are based on the best results from 10 runs.

Table 9
Experimental results when applying SA to the self-adaptive DBA algorithm (ITC2007 dataset).

Datasets	Self-adaptive DBA		Self-adaptive DBA_{SA} $T_0 = 1000$	
	Best	Avg.	Best	Avg.
Exam_1	6450	6670.4	**5645**	5768.7
Exam_2	1142	1415.9	**505**	536.6
Exam_3	11,956	12340.3	**10,515**	10586.4
Exam_4	19,426	20513.8	**16,496**	16719.6
Exam_5	7111	8330.8	**3921**	4048.8
Exam_6	26,625	26733.5	**26,330**	26,553
Exam_7	6664	6946.8	**4675**	4967.6
Exam_8	10,163	10636.8	**8912**	9129.1

Table 10
Late acceptance hill climbing experimental comparison (Toronto dataset).

Datasets	Self-adaptive DBA		Self-adaptive DBA_{LAHC}	
	Best	Avg.	Best	Avg.
car91	5.68	5.81	**4.76**	4.96
car92	4.71	4.80	**3.94**	4.16
ear83 I	36.99	38.25	**33.61**	34.44
hec92 I	11.11	11.46	**10.56**	10.76
kfu93	15.00	15.47	**13.44**	13.93
lse91	12.38	12.68	**10.87**	11.34
rye92	10.05	10.23	**8.81**	9.20
sta83 I	157.45	157.58	**157.09**	157.22
tre92	9.00	9.07	**7.94**	8.30
uta92 I	3.73	3.77	**3.27**	3.45
ute92	27.63	28.23	**25.36**	25.75
yor83 I	38.97	39.90	**35.74**	36.71

Table 8 shows that the best value for T_0 is 1000, so we used that value for ITC2007, as shown in Table 9.

6.3.2. Late acceptance hill climbing (LAHC) experimental comparison

We performed the same comparison on the self-adaptive DBA algorithm with the late acceptance hill climbing algorithm, coded as self-adaptive DBA_{LAHC}. Note that the self-adaptive method is also used to monitor the LAHC neighbourhoods search.

Comparing the self-adaptive DBA and the self-adaptive DBA_{LAHC} shows that the self-adaptive DBA_{LAHC} outperforms the self-adaptive DBA algorithm when it is applied to both problems, as shown in Tables 10 and 11. Applying the local search LAHC within a self-adaptive DBA produces better results because it accepts worse solutions during the LAHC search that later we believe lead to better solutions. A self-adaptive strategy can unstick a LAHC search from local optima.

Table 8
Experimental results when applying SA to the self-adaptive DBA algorithm (Toronto dataset).

Datasets	Self-adaptive DBA	Self-adaptive DBA_{SA}							
		$T_0 = 5000$		$T_0 = 2000$		$T_0 = 1000$		$T_0 = 500$	
		Best	Avg.	Best	Avg.	Best	Avg.	Best	Avg.
car91	5.68	4.93	5.04	4.95	5.18	4.95	5.113	**4.91**	5.09
car92	4.71	4.53	4.51	4.28	4.40	**4.26**	4.34	4.32	4.39
ear83 I	36.99	**34.35**	35.70	34.46	35.97	34.58	36.67	35.35	36.63
hec92 I	11.11	10.86	11.29	10.92	11.48	**10.77**	11.50	10.84	11.37
kfu93	15.00	14.36	15.23	14.13	14.84	**13.86**	15.06	14.05	15.08
lse91	12.38	11.34	12.09	11.45	12.09	**11.16**	11.90	11.27	11.70
rye92	10.05	9.12	9.43	**9**	9.52	**9**	9.59	9.09	9.29
sta83 I	157.45	157.26	157.43	157.31	157.86	**157.08**	157.7	157.13	157.42
tre92	9.00	8.25	8.39	8.34	8.72	**8.14**	8.36	8.25	8.36
uta92 I	3.73	3.44	3.68	3.46	3.57	**3.41**	3.51	3.44	3.53
ute92	27.63	26.57	27.54	25.79	27.72	25.86	26.81	**25.66**	26.64
yor83 I	38.97	36.99	38.49	37.51	38.65	**36.95**	37.66	37.16	38.85

Table 11
Late acceptance hill climbing experimental comparison (ITC2007 dataset).

Data sets	Self-adaptive DBA		Self-adaptive DBA_{LAHC}	
	Best	Avg.	Best	Avg.
Exam_1	6450	6670.4	**5375**	5550.5
Exam_2	1142	1415.9	**445**	471.8
Exam_3	11,956	12340.3	**10,246**	10289.5
Exam_4	19,426	20513.8	**16,019**	16304.5
Exam_5	7111	8330.8	**3511**	3607.9
Exam_6	26,625	26733.5	**26,130**	26192.5
Exam_7	6664	6946.8	**4418**	4495.1
Exam_8	10,163	10636.8	**8410**	8481.3

Table 12
Simulated annealing and late acceptance hill climbing experimental comparison (Toronto dataset).

Datasets	Self-adaptive DBA_{SA}		Self-adaptive DBA_{LAHC}		
	Best	Avg.	Best	Avg.	*p*-value
car91	4.95	5.113	4.76	4.96	**0.0274**
car92	4.26	4.34	3.94	4.16	**0.0003**
ear83 I	34.58	36.67	33.61	34.44	**8.3206E−05**
hec92 I	10.77	11.50	10.56	10.76	**0.0060**
kfu93	13.86	15.06	13.44	13.93	**0.0036**
lse91	11.16	11.90	10.87	11.34	**0.0363**
rye92	9	9.59	8.81	9.20	0.1003
sta83 I	157.08	157.7	157.09	157.22	**0.0102**
tre92	8.14	8.36	7.94	8.30	0.3279
uta92 I	3.41	3.51	3.27	3.45	**0.0707**
ute92	25.86	26.81	25.36	25.75	**0.0342**
yor83 I	36.95	37.66	35.74	36.71	**0.0001**

Table 13
Simulated annealing and late acceptance hill climbing experimental comparison (ITC2007 dataset).

Datasets	Self-adaptive DBA_{SA}		Self-adaptive DBA_{LAHC}		
	Best	Avg.	Best	Avg.	*p*-value
Exam_1	**5645**	5768.7	**5375**	5550.5	**0.0004**
Exam_2	**505**	536.6	**445**	471.8	**3.0899E−06**
Exam_3	**10,515**	10586.4	**10,246**	10289.5	**6.2161E−10**
Exam_4	**16,496**	16719.6	**16,019**	16304.5	**2.0033E−06**
Exam_5	**3921**	4048.8	**3511**	3607.9	**3.8992E−10**
Exam_6	**26,330**	26,553	**26,130**	26192.5	**6.8823E−08**
Exam_7	**4675**	4967.6	**4418**	4495.1	**2.3642E−06**
Exam_8	**8912**	9129.1	**8410**	8481.3	**4.2477E−08**

6.3.3. Results comparison between simulated annealing and late acceptance hill climbing

Tables 12 and 13 show an experimental comparison between simulated annealing and late acceptance hill climbing. A *t*-test has been conducted to show a significant difference between the two methods with confidence level of 95%. The *p*-value in Table 12 indicates significant differences in 10 out of 12 datasets. The percentage difference in Toronto dataset is 83%. The *p*-value in Table 13 shows that there are significant differences in all datasets. From the presented results, we can conclude that the self-adaptive DBA_{LAHC} outperforms self-adaptive DBA_{SA} on both categories of problems.

Table 15
The average rankings of the algorithms (Friedman) on the Toronto benchmark.

Algorithm	Ranking
Self-adaptive DBA_{LAHC}	1.6
Burke and Bykov (2008)	1.5
Yang and Petrovic (2005)	2.83

Table 16
Adjusted *p*-value (Friedman) on the Toronto dataset.

Algorithm	Adjusted *P*	P_Holm	P_Hoch
Yang and Petrovic (2005)	0.0011	0.0022	0.0022
Self-adaptive DBA_{LAHC}	0.6831	0. 6831	0.6831

6.3.4. Results comparison and analysis with the best known results on the Toronto benchmark

In this section, we first compare our best results with the best known results for the Toronto benchmark, as in Table 14. Using the results for the self-adaptive DBA_{LAHC} from the previous sections, we compare our results to three approaches that are able to produce best known results so far (taken from the survey by Qu et al. [4]).

An overall comparison with the best known results shows that we are unable to exceed any of the best known results in the literature. We are still able to produce sufficiently good solutions. The comparison also shows that our results are very close to the best approaches in the literature.

We also analyse our best results by conducting a statistical test to show the significant differences between our algorithms and the best approaches in the literature. As a statistical analysis, we first employ a Friedman test, followed by Holm and Hochberg tests as post hoc methods (if significant differences are detected), to obtain the adjusted *p*-values for each comparison between the control algorithm (the best-performing one) and the rest (Garcia et al. [47,48]). Table 15 summarises the ranking obtained by Friedman's test.

Table 15 shows that Burke and Bykov's 2008 [41] results rank first, followed by the self-adaptive DBA_{LAHC} and Yang and Petrovic's 2005 [25] results. The *p*-value computed by the Friedman test is 0.0017, which is below the critical level ($\alpha = 0.05$). This value shows that there is a significant difference among the observed results. We also performed post hoc methods (Holm's and Hochberg's test) for the self-adaptive DBA_{LAHC} algorithm. Table 16 shows the adjusted *p*-value (Friedman).

Table 14
Toronto benchmark experimental comparison.

Datasets	Self-adaptive DBA_{LAHC}		Caramia et al. [45]	Yang and Petrovic [46]		Burke and Bykov [41]	
	Best	Avg.	Best	Best	Avg.	Best	Avg.
car91I	4.76	4.96	6.6	**4.50**	4.53	4.58	4.68
car92I	3.94	4.16	6.2	3.93	3.99	**3.81**	3.92
ear83 I	33.61	34.44	**29.3**	33.7	34.87	32.65	32.91
hec92 I	10.56	10.76	**9.2**	10.83	11.36	10.06	10.22
kfu93	13.44	13.93	13.8	13.82	14.35	**12.81**	13.02
lse91	10.87	11.34	**9.6**	10.35	10.78	9.86	10.14
rye92	8.81	9.20	**6.8**	8.53	8.79	7.93	8.06
sta83 I	157.09	157.22	158.2	158.3	158.02	**157.03**	157.05
tre92	7.94	8.30	9.4	7.92	8.1	7.72	7.89
uta92 I	3.27	3.45	3.5	**3.14**	3.2	3.16	3.26
ute92	25.36	25.75	**24.4**	25.39	26.1	24.79	24.82
yor83 I	35.74	36.71	36.2	36.53	36.88	**34.78**	35.16

Table 17
Experimental results on the International Timetabling Competition (ITC2007) dataset.

Datasets	Müller [49]	Atsuta et al. [50]	Pillay [51]	Gogos et al. [52]	Self-adaptive DBA_{LAHC}	
					Best	Avg.
Exam_1	**4370**	8006	12,035	4699	5375	5550.5
Exam_2	**400**	3470	3074	385	445	471.8
Exam_3	10,049	18,622	15,917	**8500**	10,246	10289.5
Exam_4	18,141	22,559	23,582	**14,879**	16,019	16304.5
Exam_5	2988	4714	6860	**2795**	3511	3607.9
Exam_6	26,950	29,155	32,250	**25,410**	26,130	26192.5
Exam_7	4213	10,473	17,666	**3884**	4418	4495.1
Exam_8	7861	14,317	16,184	**7440**	8410	8481.3

Holm's and Hockberg's procedures show significant differences, using Burke and Bykov as a control algorithm. The self-adaptive algorithm is better than Yang and Petrovic (2005) and is comparable to Burke and Bykov (2008) with $\alpha = 0.05$ and $\alpha = 0.01$ (1/3 algorithms).

6.4. Results comparison and analysis with the best known results on the International Timetabling Competition (ITC2007) dataset

In addition, we have tested the performance of the self-adaptive DBA_{LAHC} algorithm on the International Timetabling

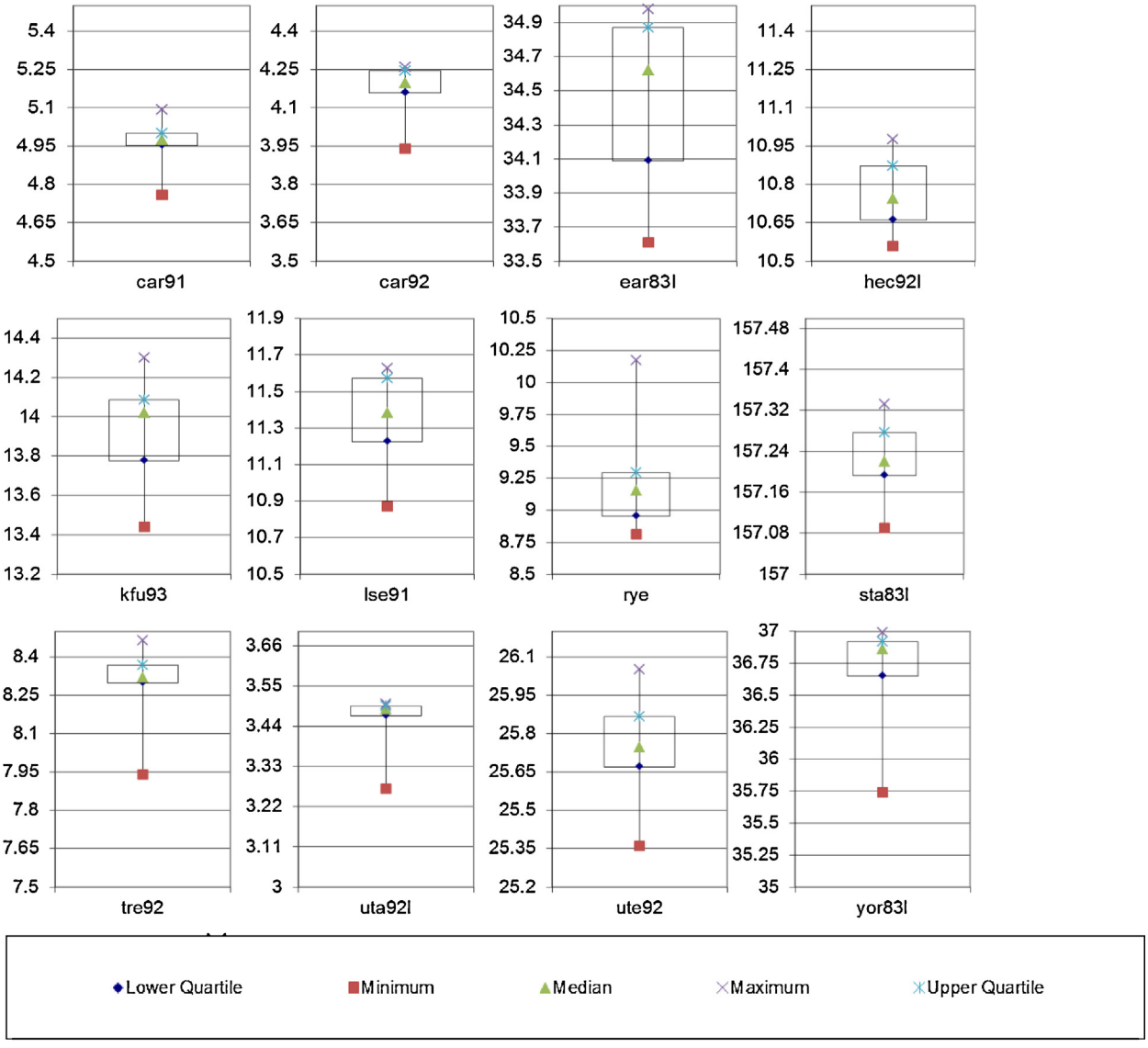

Fig. 7. Box-and-whisker plot obtained by self-adaptive DBA_{LAHC} on the Toronto dataset.

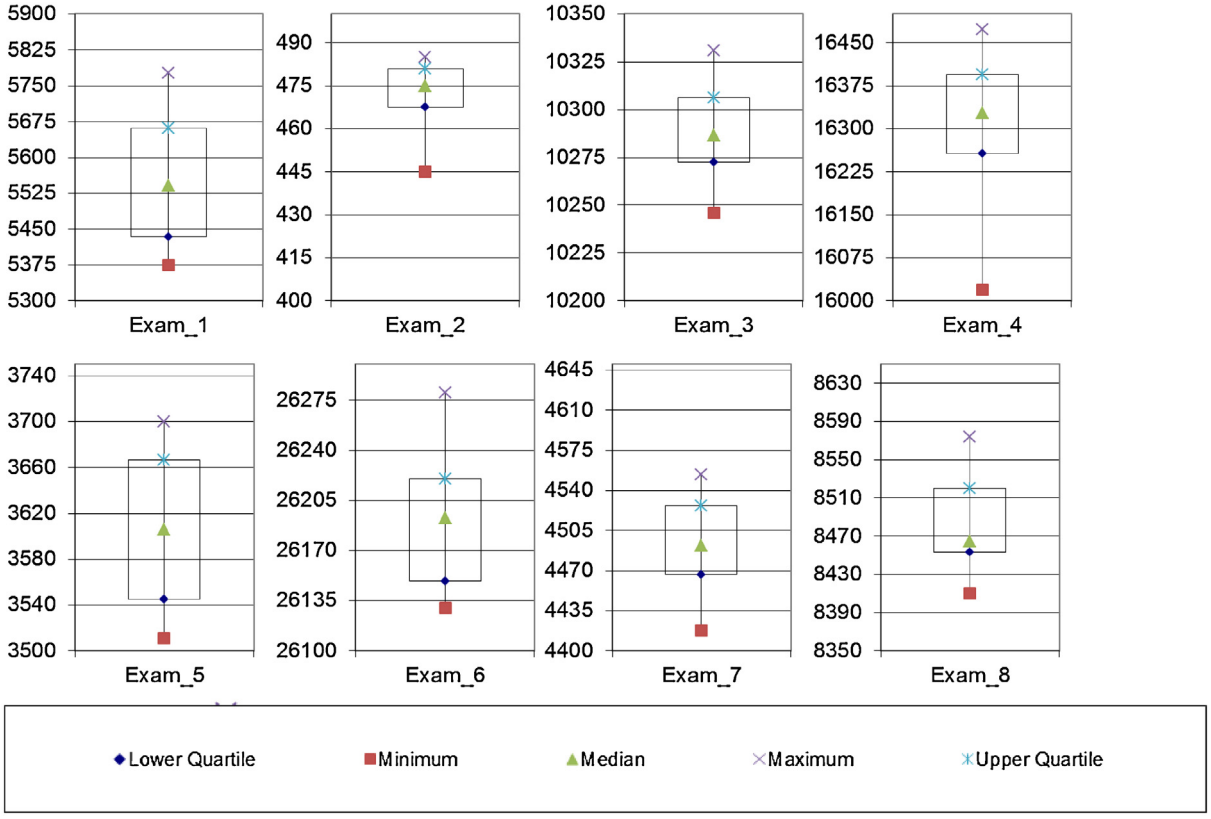

Fig. 8. Box-and-whisker plot obtained by *self-adaptive* DBA_{LAHC} on the ITC2007 dataset.

Competition (ITC2007) datasets. Table 17 shows the results and compares the proposed algorithm to five winners of the International Timetabling Competition 2007 that can be found at http://www.cs.qub.ac.uk/itc2007/, as listed below:

1st place: Tomas Müller
2nd place: Christos Gogos
3rd place: Mitsunori Atsuta, Koji Nonobe, and Toshihide Ibaraki
4th place: Geoffrey De Smet
5th place: Nelishia Pillay

Table 17 indicates that the self-adaptive DBA_{LAHC} results are comparable with state-of-the-art approaches and that the self-adaptive DBA_{LAHC} is able to obtain solutions that are better than other proposed approaches on almost all of the tested datasets. Note that the experiment performed here is terminated when the time reaches 600 s (as set in the ITC2007 computation rules).

We applied a Friedman test to the ITC2007 dataset. The *p*-value computed by Friedman's was 9.4250E-6, which is below the significance level $\alpha = 0.05$. There are significant differences among the observed results.

Table 18 shows the average algorithm rankings found by the Friedman test. From this table, Muller ranks first, followed by the self-adaptive DBA_{LAHC} algorithm, Gogos, Atsuta and Pillay, in descending order.

Table 18
The average rankings of the algorithms (Friedman) on ITC2007.

Algorithm	Ranking
Self-adaptive DBA_{LAHC}	1.875
Muller	1.125
Gogos	3.125
Atsuta et al.	4.375
Pillay	4.5

Table 19
Adjusted *p*-value (Friedman) on the ITCC2007 dataset.

Algorithm	Adjusted *P*	P_Holm	P_Hoch
Self-adaptive DBA_{LAHC}	0.342	0.342	0.342
Gogos	1.9628E−5	7.8510E−5	7.8510E−5
Atsuta et al.	3.9401E−5	1.1820E−4	1.1820E−4
Pillay	0.0114	0.0222	0.0222

Table 19 shows the adjusted *p*-value (Friedman). We conducted further post hoc methods (Holm's and Hochberg's tests).

6.4.1. The box-and-whisker plot for the Toronto benchmark and the International Timetabling Competition (ITC2007) datasets

Figs. 7 and 8 illustrate the box-and-whisker plot, which represents the distribution of solution qualities for the Toronto benchmark and International Timetabling Competition (ITC2007) datasets. It can easily be observed that the gap between the best, average and worse solution qualities are close, which indicates that the self-adaptive DBA_{LAHC} is robust. Each box has lines at the lower, median and upper quartile for the set of 10 runs. The figures show less solution point dispersion, particularly the upper and lower quartiles in Fig. 7 (car91, car92, sta83I, and uta92I datasets) and Fig. 8 (Exam_2, Exam_3 and Exam_7).

7. Conclusions

This paper describes three modifications on the BA, as follows: (i) the employment of a suitable selection strategy (i.e. a disruptive selection in this case) indicates the importance of choosing the individuals from the population for exploitation, to keep the population diversity longer, (ii) choosing a suitable adaptive method for

adaptively choosing the neighbourhood structure for a neighbourhood search, and (iii) combining the advantage of the bee algorithm with late acceptance hill climbing as a local search to increase the ability to cooperatively explore the search space and to find good solutions within a small region of the search space and to prevent deterioration. The performance of our algorithm has been tested on the Toronto benchmark and the ITC2007 datasets. The experimental results show that the three modifications of BA outperform the BA alone. Our algorithm can produce good quality solutions on both categories of the tested datasets and are comparable with state-of-the-art approaches.

References

[1] E.K. Burke, D.G. Elliman, P.H. Ford, R.F. Weare, 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 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 1996, pp. 76–90.

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

[3] T.B. Cooper, J.H. Kingston, The complexity of timetable construction problems Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling (PATAT 1995). Lecture Notes in Computer Science, vol. 1153, Springer-Verlag, Berlin, Heidelberg, 1995, pp. 283–295.

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

[5] E. Talbi, Metaheuristic: From Design to Implementation, Wiley Publishing, United States of America, 2009.

[6] E.K. Burke, J.P. Newall, R.F. Weare, A memetic algorithm for university exam timetabling, in: Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling, 1995, pp. 241–250.

[7] T.-A. Duong, K.-H. Lam, Combining constraint programming and simulated annealing on university exam timetabling, in: The Second International Conference in Computer Sciences, Research, Innovation and Vision for the Future (RIVF2004), 2004, pp. 205–210.

[8] N.D. Thanh, Solving timetabling problem using genetic and heuristic algorithms, in: ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, vol. 3, 2007, pp. 472–477.

[9] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Computing Surveys 35 (3) (2003) 268–308.

[10] Kamil, C. Alan, R. John, Krebs, R. Pulliam, Foraging Behavior, Plenum Press, New York/London, 1987.

[11] P. Ross, D. Corne, H. Terashima-Marìn, The phase-transition niche for evolutionary algorithms in timetabling, in: Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling, 1995, pp. 309–324.

[12] E.K. Burke, D. Elliman, R. Weare, A genetic algorithm for university timetabling, in: AISB Workshop on Evolutionary Computing, University of Leeds, Society for the Study of Artificial Intelligence and Simulation of Behaviour, UK, April 1994, 1994.

[13] A. Colorni, M. Dorigo, V. Maniezzo, Genetic algorithms and highly constrained problems: the time-table case, in: PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, Springer-Verlag, London, UK, 199155–59.

[14] W. Erben, J. Keppler, A genetic algorithm solving a weekly course-timetabling problem, in: Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling, 1996, pp. 198–211.

[15] M. Farooq, Bee-Inspired Protocol Engineering: From Nature to Networks, Springer, London, UK, 2008.

[16] M. Eley, Ant algorithms for the exam timetabling problem, in: E.K. Burke, H. Rudova (Eds.), PATAT 2007. Lecture Notes in Computer Science, vol. 3867, Springer, Heidelberg, 2007, pp. 364–382.

[17] H. Turabieh, S. Abdullah, A hybrid fish swarm optimisation algorithm for solving the examination timetabling problems Learning and Intelligent Optimisation Workshop (LION 5), Rome, Lecture Notes in Computer Science, vol. 6683, Springer-Verlag, Berlin, 2011539–551.

[18] N.R. Sabar, M. Ayob, G. Kendall, 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, 2009, pp. 399–408.

[19] M. Alzaqebah, S. Abdullah, Hybrid artificial bee colony search algorithm based on disruptive selection for examination timetabling problems, Lecture Notes in Computer Science 6831 (2011) 31–45.

[20] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi, The Bees Algorithm. Technical Note, Manufacturing Engineering Centre, Cardiff University, UK, 2005.

[21] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, 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, 2006.

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

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

[24] P. Lucic, D. Teodorovic, Bee system: modelling combinatorial optimization transportation engineering problems by swarm intelligence, in: Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis, Sao Miguel, Azores Islands, 2001, pp. 441–445.

[25] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. Parkes, L. DiGaspero, 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.

[26] B. McCollum, P. McMullan, E.K. Burke, A.J. Parkes, R. Qu, A new model for automated examination timetabling, Submitted Post PATAT08 special issue of Journal of Scheduling, Technical Report QUB/IEEE/Tech/ITC2007/Exam/v4.0/17, <http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index.htm>.

[27] M.S. Packianather, M. Landy, D.T. Pham, Enhancing the speed of the bees algorithm using pheromone-based recruitment, in: INDIN, 2009, pp. 789–794.

[28] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, M. Zaidi, The bees algorithm—a novel tool for complex optimisation problems, in: Proceedings of 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS2006), Cardiff, UK, 2006, pp. 454–459.

[29] D.T. Pham, A.A. Afify, E. Koç, Manufacturing cell formation using the bees algorithm, in: Proceedings of the Third International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2007), Whittles, Dunbeath, Scotland, 2007, pp. 523–528.

[30] D.T. Pham, M. Castellani, The bees algorithm—modelling foraging behaviour to solve continuous optimisation problems, Proceedings of Institution of Mechanical Engineers Part C 223 (12) (2009) 2919–2938.

[31] D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, in: Foundations of Fuzzy Logic and Soft Computing, 2007, pp. 789–798.

[32] D.E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, in: G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991, pp. 69–93.

[33] O.A. Jadaan, L. Rajamani, C.R. Rao, Improved selection operator for GA, Journal of Theoretical and Applied Information Technology (2005) 269–277.

[34] J. Zhong, X. Hu, M. Gu, J. Zhang, Comparison of performance between different selection strategies on simple genetic algorithms, in: Proceeding of the International Conference on Computational Intelligence for Modelling, Control and automation, and International Conference of Intelligent Agents, Web Technologies and Internet Commerce, 2005.

[35] T. Kuo, S.-H. Hwang, Using disruptive selection to maintain diversity in genetic algorithms, Applied Intelligence 7 (1997) 257–267.

[36] T. Blickle, L. Thiele, A mathematical analysis of tournament selection, in: Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, CA, 1995, pp. 2–8.

[37] L. Bao, J. Zeng, Comparison and analysis of the selection mechanism in the artificial bee colony algorithm, in: HIS (1), 2009, pp. 411–416.

[38] A.G. Song, J.R. Luo, A ranking based adaptive evolutionary operator genetic algorithm, Acta Electronica Sinica 27 (1) (1999) 85–88.

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

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

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

[42] S. Kirkpatrick, C. Gelatt Jr., M. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.

[43] C. Koulamas, S.R. Antony, R. Jaen, A survey of simulated annealing applications to operations-research problems, OMEGA–International Journal of Management Science (22) (1994) 41–56.

[44] E.K. Burke, Y. Bykov, A late acceptance strategy in hill climbing for exam timetabling problems, in: Proceeding PATAT'08. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, Universit de Montral, Montreal, Canada, August, 2008.

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

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

[47] S. Garcia, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization, Journal of Heuristics (15) (2009) 617–644.

[48] S. Garcia, A. Fernández, J. Luengo, F. Herrera, 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) (2010) 2044–2064.
[49] T. Muller:, ITC2007 solver description: a hybrid approach, Annals of Operation Research 172 (1) (2009) 429–446.
[50] M. Atsuta, N. Nonobe, T. Ibaraki, ITC2007 Track 1: an approach using general CSP solver. <http://www.cs.qub.ac.uk/itc2007>, 2007.
[51] A. Pillay, Developmental approach to the examination timetabling problem. <http://www.cs.qub.ac.uk/itc2007>, 2007.
[52] C. Gogos, G. Goulas, P. Alefragis, E. Housos, Pursuit of better results for the examination timetabling problem using grid resources, in: CI-Sched'09. IEEE Symposium on Computational Intelligence in Scheduling, 2009, pp. 48–53.