Journal of the Operational Research Society (2015) 66, 116–133 © 2015 Operational Research Society Ltd. All rights reserved. 0160-5682/15
www.palgrave-journals.com/jors/

A great deluge algorithm for a real-world examination timetabling problem

MN Mohmad Kahar[1,2*] and G Kendall[1,3]
[1]University of Nottingham, Nottingham, UK; [2]University Malaysia Pahang, Pahang, Malaysia; and [3]The University of Nottingham Malaysia Campus, Selangor Darul Ehsan, Malaysia

The examination timetabling problem involves assigning exams to a specific or limited number of timeslots and rooms with the aim of satisfying all hard constraints (without compromise) and satisfying the soft constraints as far as possible. Most of the techniques reported in the literature have been applied to simplified examination benchmark data sets. In this paper, we bridge the gap between research and practice by investigating a problem taken from the real world. This paper introduces a modified and extended great deluge algorithm (GDA) for the examination timetabling problem that uses a single, easy to understand parameter. We investigate different initial solutions, which are used as a starting point for the GDA, as well as altering the number of iterations. In addition, we carry out statistical analyses to compare the results when using these different parameters. The proposed methodology is able to produce good quality solutions when compared with the solution currently produced by the host organisation, generated in our previous work and from the original GDA.

Journal of the Operational Research Society (2015) **66**(1), 116–133. doi:10.1057/jors.2012.169
Published online 18 December 2013

Keywords: examination timetabling; great deluge algorithm; scheduling

1. Introduction

Examination timetabling has been the subject of research studies for many years. The exam timetable generated by any institution has a significant impact on students, lecturers and administrators. The timetable should aim to satisfy all interested parties, and hence the solution needs to take into account many factors such as ensuring that there are no clashes (ie two exams at the same time) for students, there is sufficient marking time for lecturers, and that administrators are also happy with the timetable (McCollum, 2007). Many papers discussing the examination timetabling problem can be found in the literature, with a good source being the PATAT conference series of selected papers (ie Burke and Ross, 1996; Burke and Carter, 1998; Burke and Erben, 2001; Burke and De Causmaecker, 2003; Burke and Trick, 2005; Burke and Rudova, 2007).

In this paper, we present a modification of the great deluge algorithm (GDA) that allows the boundary, which acts as the acceptance level, to dynamically change during the search. The proposed algorithm will accept a new solution if the cost value is less than or equal to the boundary, which is lowered at each iteration according to a decay rate. The proposed GDA uses a simple parameter setting and allows the boundary to increase if there is no improvement after several iterations. In addition, when the new solution is less than the desired value (estimation of the required cost value), the algorithm calculates a new boundary and a new desired value.

In order to investigate the proposed algorithm, we use a real-world capacitated examination problem taken from the Universiti Malaysia Pahang (UMP). This data set has several new constraints, in addition to those commonly found in the scientific literature. This work is an extension of our previous work described in Kahar and Kendall (2010), in which we presented a constructive heuristic. We are now attempting to improve on the (initial) solution returned from the construction heuristic. The rest of the paper is organised as follows. In Sections 2 and 3, we describe the examination timetabling problem and related work. In Sections 4 and 5, we describe the GDA and our proposed modification. A description of the UMP examination timetabling problem, including the constraints, is discussed in Section 6. In Section 7, we describe the experimental setup to allow reproducibility for other researchers. The result from the improvement phase is shown in Section 8, followed by statistical analysis in Section 9. Discussion on the results is presented in Section 10. Lastly, in Sections 11 and 12 we summarise the contribution and present our conclusions.

Correspondence: MN Mohmad Kahar, Automated Scheduling, Optimisation and Planning (ASAP) Group, School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK; and FSKKP, Universiti Malaysia Pahang, Lebuhraya Tun Razak, 26300, Gambang, Kuantan, Pahang, Malaysia
E-mail: mnizam@ump.edu.my

2. Examination timetabling problem

The examination timetabling problem involves assigning exams to a limited number of timeslots and rooms with the aim of satisfying the *hard constraints* and minimising the violations of *soft constraints* (eg Carter and Laporte, 1996; Qu *et al*, 2009). Hard constraints cannot be broken and a timetable is considered feasible if all the hard constraints are satisfied. An example of a hard constraint is that no student is assigned to sit for more than one exam at the same time. Soft constraints are requirements that are not essential but should be satisfied as far as possible; hence, they are used to evaluate the quality of the solutions through (perhaps weighted) summation of the number of violations of soft constraints. An example of a soft constraint is to spread exams as evenly as possible from a student's point of view. A list of commonly used constraints is given in Qu *et al* (2009), Merlot *et al* (2003), Burke *et al* (1996b) and Cowling *et al* (2002).

Examination timetabling problems can be categorised into un-capacitated or capacitated variants. Unlike the un-capacitated examination timetabling problem, in capacitated problems room capacities are treated as a hard constraint (which more closely reflects real-world problems), in addition to the other commonly used hard constraints (Abdullah, 2006; Pillay and Banzhaf, 2009). Most of the research seen in the literature has investigated the un-capacitated examination timetabling problem, the Toronto data set (Carter *et al*, 1996) being a good example. Research on this data set has mainly focussed on the algorithmic performance to produce solutions effectively and quickly (see Qu *et al*, 2009). However, McCollum (2007), Qu *et al* (2009) and Carter and Laporte (1996) believe that researchers are not dealing with all aspects of the problem. That is, they are only working on a simplified version of the examination problem, which only addresses a few common hard constraints (ie no exams with common students assigned simultaneously) and soft constraints (ie spreading conflicting exams as evenly as possible).

Capacitated problems more closely resemble real-world problems as they include a room capacity constraint, and the survey by Burke *et al* (1996b) showed that 73% of universities agree that accommodating exams is a difficult problem. The difficulties are caused by (a) the lack of available exam rooms due to (eg) the room being used for teaching purposes and (b) the splitting of exams between more than one room. The size of an exam alone could easily cause the exams to be split across more than one room and this should result in additional constraints such as splitting exams onto different sites and/or the distance between rooms (Burke *et al*, 1996b).

However, the capacitated problem has received less attention from the research community, probably due to the lack of benchmark data sets. Furthermore, it requires more comprehensive data as it has to include the room capacities and this information can be difficult to collect (McCollum, 2007). Examples of capacitated examination data sets available in the literature are the Nottingham data set (Burke *et al*, 1996a), the Melbourne data set (Merlot *et al*, 2003), the International Timetabling Competition (ITC2007) data set (McCollum *et al*, 2010) and the Toronto data set (which includes the total seating capacity introduced by Burke *et al*, 1996a).

The Nottingham, Melbourne and (*modified*-)Toronto data set is concerned with the total seating capacity (allowing multiple exams in the same room). That is, the total number of students sitting in all exams, in the same timeslot, must be less than some specified number. This represents a simplified problem, whereas normally in solving a real-world problem we would have to take into account individual room capacities (Merlot *et al*, 2003), but this obviously depends on institutional requirements. The ITC2007 data set provides more realistic problems than the (*modified*-)Toronto, Nottingham and Melbourne data set problems, as it considers individual room capacities. However for the UMP data set, besides considering individual room capacities, it does not allow multiple exams to share a single room. This complicates the problem even further. A description of UMP constraints is described in Section 6 (also see Kahar and Kendall, 2010).

3. Related work

A variety of techniques have been utilised in solving the examination timetabling problem (see Carter and Laporte, 1996; Schaerf, 1999; and Qu *et al*, 2009). One popular technique is Hill climbing (HC). HC accepts a candidate solution, if it is an improving solution. HC is simple and easy to implement; however, it is easily trapped in local optima (Burke and Kendall, 2005). Recently, Burke and Bykov (2008) have proposed a late acceptance strategy for HC. The method delays the comparison step between a candidate solution and the current (best) solution. The late acceptance HC methodology is able to produce good quality solutions compared to other works for the Toronto data sets. Tabu search (Glover, 1986) works in a similar manner to HC but incorporates a memory. To prevent the search from becoming stuck in a local optima, a tabu list (memory) is used to hold recently seen solutions or, more likely, some of the attributes of the neighbourhood move and ignore solutions that are in the tabu list (Di Gaspero and Schaerf, 2001; White *et al*, 2004).

Simulated annealing (SA; Kirkpatrick *et al*, 1983) always accepts better solutions but also accepts worse solutions with a decreasing probability as the search progresses (Merlot *et al*, 2003; Burke *et al*, 2003). Great deluge (Dueck, 1993) works in almost the same manner as SA but it uses a boundary as the acceptance criteria, which is gradually reduced. New solutions are only accepted if they improve on the current solution or are within the boundary value. A further discussion on this methodology follows in the next section.

Variable Neighbourhood Search (VNS) (Hansen and Mladenovic, 1999) is based on the strategy of using more than one neighbourhood structure and changing them systematically during the search. The use of many neighbourhoods allows VNS to more effectively explore the search space (Abdullah *et al*, 2005; Burke *et al*, 2010a). Other techniques in the literature include ant colony optimisation (Dorigo *et al*, 1996). ACO is inspired by the behaviour of ants, and the way they forage for food. Only a few works using ACO for exam timetabling have been reported in the literature (Eley, 2006; Eley, 2007).

Genetic Algorithms (Burke and Kendall, 2005) are a population-based search that uses the principle of biological evolution (ie selection, mutation, crossover) to generate better solutions from one generation to another (Ross and Corne, 1995; Burke *et al*, 2010a). Memetic algorithms are a hybridisation of GA, by incorporating a local-search algorithm. MAs have been shown to work effectively compared to GAs (Burke *et al*, 1996a; Abdullah *et al*, 2009b).

Hyper-Heuristics (HH; Burke *et al*, 2010b) are methods that attempt to raise the level of generality at which search methodologies operate. HH can be categorised as a method that selects heuristics or generates new heuristics from components of existing heuristics (Burke *et al*, 2010b). HH are concerned with the exploration of a heuristic space rather than dealing directly with solutions to the problem (Kendall and Hussin, 2004; Hussin, 2005; Burke *et al*, 2007).

Many other techniques can be found in the PATAT series of conference volumes.

4. The great deluge algorithm

In 1993, Dueck introduced the great deluge algorithm (GDA) that operates in a similar way to SA. However, GDA uses an upper limit (often referred to as the water level) as the boundary of acceptance rather than a temperature. The algorithm starts with a boundary equal to the initial solution quality. It accepts worse solutions if the cost (objective value) is less than the boundary, which is lowered in every iteration according to a predetermined rate (known as the decay rate). GDA only involves one parameter (decay rate), which is an advantage over SA since the effectiveness of a meta-heuristic technique is often dependent upon parameter tuning (Petrovic and Burke, 2004).

Dueck (1993) applied GDA to the travelling salesman problem. The decay rate used was the difference between the boundary and the length of the current tour divided by 500 or 0.01. Dueck was able to find good quality solutions. Burke and Newall (2003) implemented GDA in order to solve examination timetabling problems. The decay rate is computed as the initial solution multiplied by a user-provided factor divided by the number of iterations. The algorithm was run for up to 200 000 000 iterations and the search terminated if there was no improvement in the last 1 000 000 iterations. They compared the performance of GDA with SA and HC, and reported that GDA was superior to both these algorithms.

Burke *et al* (2004) implemented time-predefined GDA for the examination timetabling problem. The algorithm includes two user-defined parameters: (a) computational time (amount of time allowed) and (b) the desired solution (an estimation of the evaluation function that is required). The decay rate is calculated as the difference between the initial solution and the desired solution divided by the computational time (or number of iterations). According to Burke *et al*, the time-predefined GDA was superior when compared to other methods. McMullan (2007) proposed the use of a steeper decay rate (with decay rate propotional to 50% of the entire run on the first stage and 25% on the remaining runs) compared to Burke *et al* (2004), in order to force the algorithm to reach better quality solutions as early as possible. He also allowed the algorithm to 're-heat' (similar to SA), allowing worse moves to be accepted. The re-heat mechanism is activated when there is no improvement. The method is able to find better solutions when compared to other methods (except on small instances).

Silva and Obit (2008) proposed a non-linear decay rate for the boundary control using an exponential function. They also included a feature that allows the boundary to rise when its value is about to converge to the penalty cost of the solution. Experimentation on the course timetabling problem revealed that the non-linear GDA outperforms some of the previous results. Abdullah *et al* (2009a) investigated the hybridisation of a GDA and tabu search in solving the course timetabling problem. Their algorithm applied four neighbourhood moves (at every iteration) and selected the best move. If there is no improvement within a specified time, the boundary is increased by a value between 0 and 3, selected at random. The combination of the two meta-heuristics produced good results.

In this work, we utilise GDA for the UMP examination problem, a real-world timetabling problem that contains several new constraints. Full details of the problem are given in Section 6.

5. Modified great deluge algorithm (*modified*-GDA)

Suitable parameter settings are important in meta-heuristics and it is often difficult to determine the best values to guarantee a good quality solution (Petrovic and Burke, 2004). The introduction of a simple and easy to understand parameter (ie computational time and desired value) to determine the decay rate in Burke *et al* (2004) made it straightforward for non-experts (eg university timetabling officers) to set the parameters, especially when compared to other meta-heuristic techniques (eg SA—cooling schedule, tabu search—tabu list size, genetic algorithm—mutation or crossover probability rate etc). Furthermore, they reported that their time-predefined GDA was able to produce good quality solutions.

The success of GDA and the simplicity in setting the parameters is the motivation for us to explore this method, with the aim of bringing GDA to the university timetabling officers as they are the persons responsible for producing the timetable at the UMP. Our proposed GDA is shown in Figure 1.

The algorithm starts by setting the desired value D, number of iterations I and the boundary level B (lines 1–5). The boundary level B is set slightly higher (3%) than the initial solution $f(s)$ obtained from a constructive heuristic (Kahar and Kendall, 2010). This is to allow acceptance of poorer solutions. We have tried several other percentages; a higher percentage leads to the search being unfocused, while a smaller percentage discourages exploration. On the basis of our observation, a value of 3% is suitable for the investigated data set. The decay rate is calculated as the difference between boundary level B and the desired solution D, divided by the number of iterations (line 6). While the stopping condition is not met, we apply the chosen neighbourhood heuristics. We calculate the new cost value $f(s^*)$, where $s^* \in N(s)$ is selected at random (line 10). s^* is accepted if $f(s^*)$ is less than or equal to $f(s)$ or if $f(s^*)$ is less than or equal to boundary B (lines 11–12). If $f(s^*)$ is

```
1.  Set the initial solution s from the constructive heuristic (Kahar
    and Kendall, 2010a)
2.  Calculate initial cost function $f(s)$
3.  Set the desired value $D$
4.  Set the number of iterations $I$
6.  Set Initial Boundary Level $B = 0.03f(s)+f(s)$
5.  Set initial decay Rate $\Delta B = (B-D)/I$
7.  Set $s_{best} = s$
8.  While stopping criteria not met do
9.  Apply neighbourhood Heuristic on $S$
10. Calculate $f(s^*)$
11. If $f(s^*) \leq f(s)$ or $f(s^*) \leq B$ Then
12. Accept $s = s^*$
13. If $f(s^*) \leq f(s_{best})$ Then
14. $s_{best} = s^*$
15. Lower Boundary $B = B - \Delta B$
16. If no improvement in W iterations or $B \leq f(s_{best})$ or $f(s) \leq D$ then
17. Set $s = s_{best}$
18. If $f(s) \leq D$ then
19. $D = f(s)*0.8$
20. Set new decay rate $\Delta B = (f(s)-D)/I_{remaining}$
21. Set $B = 0.03f(s)+f(s)$
```

Figure 1 Our proposed great deluge algorithm.

less than or equal to $f(s_{best})$, set $s_{best} = s^*$ (lines 13–14). Next, the boundary B is lowered based on the decay rate, ΔB (line 15). However, if there is no improvement for several iterations, W ($W = 5$ in this work) or boundary B is less than or equal to $f(s_{best})$ or $f(s)$ is less than or equal to the desired value; then set $s = s_{best}$ (line 17). The new decay rate ΔB is calculated as the difference between $f(s)$ and the desired value divided by the remaining number of iterations (line 20). However, if $f(s)$ is less than or equal to the desired value, then a new desired value is calculated as 80% of $f(s)$ (lines 18–19). This dynamically allows the search to continue by having a new desired value. The value of 0.8 is chosen to avoid having a steep boundary thus encouraging exploration of the search space. Hence, the boundary is set 3% above $f(s)$ (line 21). Having this condition enables the algorithm to dynamically adjust the boundary, decay rate and desired value during the search.

We are going to compare the *modified*-GDA performance with the GDA proposed by Dueck (1993) (which will be referred to as *Dueck*-GDA in the following section) and solution produced by the UMP and from our previous work (Kahar and Kendall, 2010).

6. University Malaysia Pahang (UMP): examination timetabling problem

The UMP was established in 2002 and currently operates from a temporary campus. This presents many challenges in terms of available space, logistics and human resources to manage the university operation. Furthermore, new faculties are being introduced along with new programmes being offered. This results in an increase in the number of students, which, according to the timetable officer, makes it difficult to generate a feasible examination timetable. In addition to these limitations, the UMP examination timetable problem has other challenging constraints. The hard and the soft constraints for the UMP *examination assignment* are listed in Table 1. Further details and the mathematical model are available in Kahar and Kendall (2010).

6.1. UMP examination timetabling data set

Our investigations were carried out using two different data sets from semester1-2007/08 and semester1-2008/09. In semester 1-2007/08, the total number of examination papers is 157, across 17 programmes offered by five faculties. The total number of students is 3550 with 12 731 enrolments. The conflict density matrix is 0.05, which means that 5% of the students are in conflict among the examination papers. The total available exam space for this data set is 24 rooms, with each room having a given capacity. For semester1-2008/09 the data set contains a total of 165 examination papers across 23 programmes offered by seven faculties. The total number of students is

Table 1 The exam timetabling constraints

Hard constraints
1. No student should be required to sit two examinations simultaneously
2. The total number of students assigned to a particular room(s) must be less than the total room capacity
3. Exams that have been split into several rooms must be in the same building
4. Only one examination paper is scheduled to a particular room. That is, there is no sharing of rooms with other exam papers (even if enough seats are available)

Soft constraints

5. Each set of student examinations should be spread as evenly as possible over the exam period
6. The distance between exam rooms, for the same exam, should be as close as possible to each other (and within the same building)
7. A penalty value is applied when splitting an exam across several rooms, as we prefer an exam to be in a single (or as few as possible) room whenever possible

Table 2 Summary of the UMP data set

Categories	*Semester1-2007/08*	*Semester1-2008/09*
Exams	157	165
Students	3550	4284
Enrolments	12 731	15 416
Conflict density	0.05 (5%)	0.05 (5%)
Timeslots per day	2	2
Rooms	24	28

4284 with 15 416 enrolments. The conflict density matrix is 0.05. The total available exam space for this data set is 28 rooms. Table 2 summarises these data sets.

The number of exam days and timeslots are 10 and 20, respectively, with two timeslots on each examination day. There are no exams during the weekend (Saturday and Sunday), hence we exclude the weekend timeslots in our timeslot indices. The timeslot indices are as follows: *1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23* and *24*. Indices *1* and *2* refer to day 1, timeslots 3 and 4 refer to day 2 and so on. There are no indices *11–14* because these indices would refer to Saturday and Sunday.

7. Experimental setup

We run *Dueck*-GDA and our *modified*-GDA using several initial solutions selected randomly within the minimum to maximum values of the constructive solution presented in Kahar and Kendall (2010). Note that, in Kahar and Kendall (2010), the minimum and maximum values produced in semester 1-2007/08 is 4.74 and 20.74, and in semester1-2008/09 it is 6.16 and 23.11, respectively. Hence, the (randomly) selected initial solutions in semester1-2007/08 are 16.68, 13.74, 10.30 and 7.82, and for semester1-2008/09 they are 18.40, 15.25, 12.30 and 9.21. We ran both methods with 1500 and 3000 iterations. The following neighbourhood heuristics are used in our experiments. Note that, unless stated otherwise, all the exam, timeslots and rooms are selected randomly.

(Nh1) Move an exam to a different timeslot and room(s). This move is only possible when the destination room and timeslot is empty.

(Nh2) Move an exam to a different room(s) within the same timeslot. This move is only possible when the destination room is empty.

(Nh3) Move an exam to a different timeslot maintaining the currently assigned room(s).

(Nh4) Choose an exam from a candidate list of 30, where exams are chosen based on their contribution to the objective function. An exam is chosen using roulette wheel selection and moved to a different timeslot and room(s).

(Nh5) Same as Nh4, but move the exam to a different room(s) within the same timeslot.

(Nh6) Same as Nh4, but move the exam to a different timeslot maintaining the currently assigned room(s).

(Nh7) Select two exams and swap the timeslot and room(s) between them.

(Nh8) Select two timeslots and swap the timeslot between them. As an example if timeslot 2 and timeslot 6 were selected, we would move the exams in timeslot 2 to timeslot 6 and vice versa.

(Nh9) Same as Nh4, but instead of moving the exam, we swap the selected exam with another exam.

(Nh10) Select two timeslots and move all exams between the two timeslots. As an example, if timeslot 2 and timeslot 6 were selected, move exams in timeslot 2 to timeslot 3; followed by moving exams in timeslot 3 to timeslot 4 and so on until exams in timeslot 6 are moved to timeslot 2.

In the next section, we show the results when using each of these neighbourhoods.

8. Examination assignment: results

In this section, we compare the examination timetable generated by the UMP proprietary software, our constructive heuristic (Kahar and Kendall, 2010), and *Dueck*-GDA with our proposed GDA (*modified*-GDA). We used the Delphi programming language. Each experiment was run 50 times on a Pentium core2 processor. The running time for 1500 iterations is around 480 s while 3000 iterations take about 960 s. The result for semester1-2007/08 is shown in Table 3 and for semester1-2008/09 is shown in Table 4.

Table 3 GDA result for semester1-2007/08

Neighbourhood moves	*Initial cost*	*Modified-GDA*								*Dueck-GDA*							
		480 s				*960 s*				*480 s*				*960 s*			
		Ave	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*
(Nh1) Move an exam to a different timeslot and room(s)	16.68	6.19	0.37	5.51	7.08	5.50	.30	4.99	6.03	13.72	0.39	12.80	14.40	12.92	0.31	12.22	13.53
	13.74	5.81	0.31	5.24	6.64	5.32	0.25	4.76	5.88	11.70	0.29	10.94	12.12	10.53	0.26	9.76	10.95
	10.30	5.16	0.26	4.65	5.67	4.59	0.23	4.10	5.04	8.79	0.11	8.49	9.02	7.47	0.06	7.29	7.57
	7.82	4.79	0.15	**4.53**	5.30	4.38	0.15	**4.01**	4.73	6.44	0.06	6.31	6.55	5.16	0.08	5.01	5.35
(Nh2) Move an exam to a different room(s) within the same timeslot.	16.68	16.55	0.04	16.48	16.58	16.54	0.05	16.48	16.58	16.55	0.04	16.48	16.58	16.56	0.04	16.48	16.58
	13.74	13.22	0.00	13.21	13.22	13.22	0.00	13.21	13.22	13.22	0.00	13.21	13.22	13.22	0.00	13.21	13.22
	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30
	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82
Nh3) Move an exam to a different timeslot maintaining the current assigned room(s)	16.68	7.44	0.66	5.97	9.83	6.60	0.53	5.22	8.21	12.23	0.63	10.23	13.68	11.66	0.37	10.75	12.67
	13.74	6.90	0.31	6.12	7.61	6.34	0.27	5.84	6.89	11.13	0.36	10.03	11.81	10.13	0.25	9.58	10.55
	10.30	5.97	0.78	5.00	8.12	5.78	0.74	5.03	7.86	8.48	0.21	7.89	8.82	7.25	0.15	6.87	7.52
	7.82	5.71	0.33	4.90	6.39	5.49	0.32	4.79	6.06	6.41	0.23	6.16	7.82	5.15	0.19	4.80	5.86
(Nh4) Move an exam selected among the highest penalty value to a different timeslot and room(s).	16.68	6.87	0.38	5.89	7.86	6.76	0.26	6.25	7.35	9.28	0.34	8.62	9.97	9.01	0.30	8.29	9.87
	13.74	6.76	0.38	5.88	7.58	6.64	0.45	5.77	7.88	9.48	0.34	8.79	10.22	9.15	0.33	8.50	9.71
	10.30	5.62	0.36	4.98	6.36	5.65	0.38	4.81	6.46	7.44	0.32	6.63	8.05	7.05	0.15	6.64	7.35
	7.82	5.22	0.19	4.90	5.73	5.22	0.21	4.57	5.84	6.22	0.13	5.90	6.47	5.62	0.24	5.03	6.50
(Nh5) Same as in (Nh4), but move the exam to a different room(s) within the same timeslot	16.25	16.54	0.00	16.54	16.54	16.54	0.00	16.54	16.54	16.54	0.00	16.54	16.54	16.54	0.00	16.54	16.54
	13.74	13.50	0.01	13.49	13.51	13.50	0.01	13.49	13.51	13.50	0.01	13.49	13.51	13.50	0.01	13.49	13.51
	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30
	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82
(Nh6) Same as in (Nh4), but move the exam to a different timeslot maintaining the current assigned room(s).	16.68	7.96	0.79	6.82	10.10	7.91	0.76	6.70	10.20	7.75	0.54	6.85	9.03	7.61	0.47	6.61	8.98
	13.74	8.21	0.59	7.13	9.22	7.97	0.67	7.05	9.78	8.45	0.40	7.78	9.18	8.05	0.39	7.26	8.99
	10.30	6.52	0.48	5.61	8.88	6.49	0.47	5.95	8.82	6.31	0.35	5.68	7.17	6.29	0.36	5.52	7.17
	7.82	6.19	0.30	5.40	6.92	6.07	0.38	4.79	6.83	5.89	0.48	**5.07**	7.31	5.96	0.45	5.16	7.00
(Nh7) Select two exams randomly and swap the timeslot and room(s) between them	16.68	7.12	0.43	6.35	8.27	6.66	0.34	6.04	7.38	12.05	0.54	10.87	13.03	11.36	0.53	9.94	12.39
	13.74	7.37	0.43	6.53	8.85	6.97	0.43	6.20	8.08	10.85	0.40	9.83	11.52	10.02	0.27	9.22	10.65
	10.30	5.63	0.47	4.63	6.53	5.55	0.53	4.42	6.58	8.40	0.19	7.92	8.74	7.22	0.17	6.50	7.51
	7.82	5.40	0.31	4.66	6.01	5.09	0.30	4.66	5.79	6.29	0.12	5.91	6.47	5.17	0.15	**4.94**	5.54
(Nh8) Select two timeslots and swap the timeslot between them	16.68	8.45	0.38	7.68	9.48	8.36	0.43	7.40	9.59	9.50	0.45	8.25	10.82	8.93	0.35	8.19	9.67
	13.74	8.43	0.39	7.72	9.37	8.42	0.39	7.73	9.21	8.78	0.34	8.07	9.38	8.37	0.35	7.52	9.16
	10.30	7.20	0.33	6.64	7.82	7.17	0.28	6.66	7.83	7.25	0.34	6.67	8.03	7.13	0.31	6.57	7.81
	7.82	6.55	0.22	6.18	7.13	6.52	0.30	5.77	7.34	6.54	0.23	6.11	6.94	6.57	0.26	6.08	7.07

Table 3 *Continued*

Neighbourhood moves	*Initial cost*	*Modified-GDA*								*Dueck-GDA*							
		480 s				*960 s*				*480 s*				*960 s*			
		Ave	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*
(Nh9) Same as in (Nh4), but instead of moving exam, we swap the selected exam with another exam	16.68	8.21	0.50	7.32	9.41	8.07	0.45	7.16	9.36	9.04	0.42	8.11	9.76	8.82	0.34	8.05	9.54
	13.74	8.13	0.68	6.91	9.77	7.76	0.47	6.90	8.86	8.47	0.48	7.49	9.54	8.10	0.25	7.52	8.61
	10.30	6.26	0.46	5.34	7.30	6.04	0.60	4.92	7.67	6.71	0.27	6.17	7.29	6.39	0.20	6.02	6.80
	7.82	6.02	0.28	5.44	6.72	6.04	0.35	5.35	6.78	5.79	0.20	5.38	6.41	5.43	0.30	4.91	6.14
(Nh10) Select two timeslots and move all the timeslot between them	16.68	8.77	0.46	7.79	9.88	8.60	0.44	7.82	9.61	9.77	0.48	8.58	10.65	9.11	0.41	8.19	9.88
	13.74	8.55	0.35	8.01	9.40	8.48	0.37	7.88	9.44	8.88	0.39	8.02	9.60	8.42	0.35	7.59	9.07
	10.30	7.48	0.32	6.79	8.34	7.31	0.38	6.57	8.44	7.32	0.31	6.65	8.10	7.13	0.33	6.57	7.99
	7.82	6.65	0.39	5.83	7.07	6.56	0.38	5.92	7.07	6.59	0.37	5.93	7.48	6.51	0.34	5.81	7.00

Ave = average; var = variance; stdev = standard deviation; min = minimum; max = maximum.
Minimum values are shown in bold.

Table 4 GDA result for semester1-2008/09

Neighbourhood moves	*Initial cost*	*Modified-GDA*								*Dueck-GDA*							
		1500 *iterations* ≈ 8 *min*				3000 *iterations* ≈ 16 *min*				1500 *iterations* ≈ 8 *min*				3000 *iterations* ≈ 16 *min*			
		Ave	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*
(Nh1) Move an exam to a different timeslot and room(s).	18.40	8.42	0.33	7.61	9.37	7.55	0.28	6.96	8.26	15.67	0.40	14.76	16.48	14.78	0.24	14.30	15.27
	15.25	7.61	0.31	6.89	8.18	7.07	0.30	6.29	7.74	13.37	0.23	12.44	13.78	12.24	0.13	11.96	12.49
	12.30	6.89	0.24	6.33	7.61	6.38	0.20	6.03	6.89	9.50	0.06	9.36	9.61	9.48	0.08	9.22	9.61
	9.21	6.44	0.20	**6.11**	6.87	6.04	0.15	**5.63**	6.42	7.91	0.06	7.74	8.04	6.73	0.13	6.47	7.10
(Nh2) Move an exam to a different room(s) within the same timeslot	18.40	18.34	0.01	18.32	18.34	18.34	0.00	18.33	18.34	18.34	0.00	18.33	18.34	18.34	0.01	18.32	18.34
	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25
	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30
	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21
(Nh3) Move an exam to a different timeslot maintaining the current assigned room(s)	18.40	8.00	0.36	7.12	8.69	7.26	0.30	6.78	8.03	14.33	0.66	12.68	15.50	13.68	0.40	12.86	14.36
	15.25	8.81	0.33	8.07	9.85	8.23	0.28	7.70	8.71	12.86	0.33	12.00	13.40	11.91	0.26	11.09	12.27
	12.30	8.25	0.35	7.66	8.99	8.02	0.33	7.32	8.80	10.64	0.22	10.05	10.95	9.43	0.09	9.19	9.65
	9.21	7.55	0.25	6.95	8.25	7.13	0.31	6.38	7.82	7.85	0.08	7.53	7.98	6.68	0.14	6.49	7.26

(Nh4) Move an exam selected among the highest penalty value to a different timeslot and room(s).	18.40	10.15	0.52	9.11	11.41	9.91	0.52	9.09	11.22	12.33	0.35	11.42	13.01	12.07	0.34	11.21	12.81
	15.25	9.40	0.41	8.52	10.25	9.28	0.47	7.92	10.06	11.49	0.44	10.44	12.48	11.30	0.32	10.28	12.01
	12.30	8.46	0.31	7.66	9.18	8.38	0.37	7.72	9.55	10.18	0.24	9.66	10.66	9.56	0.16	9.30	9.97
	9.21	7.47	0.18	7.14	7.84	7.48	0.20	6.97	7.85	8.07	0.14	7.79	8.40	7.91	0.23	7.37	8.38
(Nh5) Same as in (Nh4), but move the exam to a different room(s) within the same timeslot	18.40	18.39	0.00	18.39	18.39	18.39	0.00	18.39	18.39	18.39	0.00	18.39	18.39	18.39	0.00	18.39	18.39
	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25
	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30
	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21
(Nh6) Same as in (Nh4), but move the exam to a different timeslot maintaining the current assigned room(s)	18.40	10.98	1.09	9.15	14.57	10.24	0.76	8.98	11.72	11.33	1.25	9.48	14.99	10.66	1.08	9.28	14.71
	15.25	10.98	0.24	10.30	11.61	9.28	0.47	7.92	10.06	10.99	0.31	10.34	11.54	10.88	0.25	10.12	11.37
	12.30	9.80	0.30	9.29	10.54	9.80	0.31	9.21	10.41	9.75	0.35	9.11	10.48	9.75	0.34	9.00	10.46
	9.21	8.96	0.20	8.15	9.14	9.00	0.22	7.86	9.21	8.50	0.37	7.62	9.02	8.50	0.38	7.71	9.09
(Nh7) Select two exams randomly and swap the timeslot and room(s) between them	18.40	8.99	0.39	8.34	10.01	8.53	0.29	7.99	9.30	14.08	0.51	13.20	15.30	13.62	0.40	11.98	14.39
	15.25	8.21	0.39	7.57	9.27	7.79	0.40	6.96	9.28	12.44	0.35	11.74	13.03	11.68	0.21	11.05	12.05
	12.30	7.03	0.26	6.62	8.07	6.73	0.25	6.19	7.40	10.24	0.26	9.52	10.70	9.13	0.19	8.66	9.45
	9.21	6.71	0.27	6.21	7.35	6.37	0.24	5.86	6.85	7.79	0.09	7.56	7.96	6.57	0.09	**6.39**	6.85
(Nh8) Select two timeslots and swap the timeslot between them	18.40	10.79	0.45	9.39	11.90	10.23	0.45	9.31	11.12	11.35	0.49	10.31	12.73	10.68	0.43	9.93	11.79
	15.25	10.38	0.43	9.05	11.27	9.79	0.50	9.02	10.88	10.32	0.43	9.49	11.12	9.85	0.46	8.87	11.02
	12.30	9.66	0.34	8.96	10.31	9.69	0.34	8.98	10.36	9.71	0.35	9.03	10.61	9.60	0.31	8.97	10.70
	9.21	8.57	0.06	8.48	8.75	8.58	0.06	8.48	8.76	8.57	0.07	8.48	8.75	8.56	0.05	8.48	8.64
(Nh9) Same as in (Nh4), but instead of moving exam, we swap the selected exam with another exam	18.40	10.96	0.52	9.79	11.93	10.74	0.52	9.72	11.99	11.81	0.36	11.00	12.72	11.46	0.40	10.51	12.21
	15.25	9.88	0.38	8.97	10.64	9.85	0.37	9.09	10.53	10.63	0.36	9.96	11.45	10.22	0.35	9.48	10.93
	12.30	8.49	0.31	7.82	9.15	8.39	0.36	7.67	9.38	8.79	0.32	7.70	9.52	9.72	0.26	9.16	10.55
	9.21	7.52	0.24	7.02	8.02	7.45	0.23	6.78	7.87	7.55	0.17	**7.20**	7.92	7.39	0.23	7.01	7.89
(Nh10) Select two timeslots and move all the timeslot between them	18.40	10.59	0.45	9.51	11.67	10.40	0.47	9.52	11.84	11.42	0.45	10.29	12.33	10.74	0.45	9.92	11.93
	15.25	10.23	0.49	9.25	11.41	10.06	0.49	8.96	10.99	10.49	0.34	9.66	11.41	10.06	0.43	9.10	10.85
	12.30	9.73	0.32	9.26	11.15	9.67	0.20	9.38	10.21	9.70	0.23	9.16	10.50	9.59	0.27	8.94	10.33
	9.21	8.49	0.11	8.30	8.73	8.48	0.09	8.30	8.76	8.50	0.09	8.30	8.66	8.46	0.07	8.30	8.61

Ave = average; var = variance; stdev = standard deviation; min = minimum; max = maximum.
Minimum values are shown in bold.

8.1. Semester1-2007/08

8.1.1. Modified-GDA versus UMP proprietary software. The UMP proprietary software solution is 13.16 with a violation of one of the hard constraints (violating the no-clash requirement, no.1 in Table 1; Kahar and Kendall, 2010). Table 3 presents our results using the *modified*-GDA. Note that all of our results respect all the hard constraints. Using *modified*-GDA with 1500 iterations, we are able to produce a solution that is 66% (13.16 compared with 4.53 ((13.16−4.53)/13.16 ×100%)) better with Nh1 when using an initial solution with a cost of 7.82 compared to the solution produced by the proprietary software. The same calculation of percentage is used throughout the discussion. Increasing the number of iterations to 3000, the solution produced with Nh1, using an initial cost of 7.82, is 70% (13.16 compared with 4.01) better when compared to the proprietary software and 11% (4.53 compared with 4.01) better compared to using 1500 iterations. However, increasing the number of iterations, obviously, increases the computational time.

8.1.2. Modified-GDA versus constructive heuristic. In the constructive heuristic (Kahar and Kendall, 2010), the best solution found was 10.98 and 4.74 using a candidate list of one and five, respectively. Comparing *modified*-GDA with the constructive heuristic using a candidate list of one, in *modified*-GDA with 1500 iterations (Table 3), we are able to produce a solution that is 59% (10.98 compared with 4.53) better with Nh1 using an initial solution of 7.82. Even with a poorer initial cost of 16.68, we are still able to improve the solution by 50% (10.98 compared with 5.51) with Nh1. Extending the search to 3000 iterations, initial cost of 7.82 and 16.68, Nh1 produced solutions with a 63% (10.98 compared with 4.01) and 55% (10.98 compared with 4.99) improvement when compared to the constructive heuristic solution. In the constructive heuristic, with a candidate list of five, *modified*-GDA is able to produce a better solution but with a smaller margin of improvement. Using an initial cost of 7.82 with 1500 and 3000 iterations, the GDA solution outperforms the constructive heuristic by 4% (4.74 compared with 4.53) and 15% (4.74 compared with 4.01), respectively. However, using a large initial cost of 16.68, with 1500 and 3000 iterations, the constructive heuristic outperforms the *modified*-GDA by 14% (5.51 compared with 4.74) and 5% (4.99 compared with 4.74), respectively.

8.1.3. Modified-GDA versus Dueck-GDA. In the *Dueck*-GDA approach, with 1500 iterations it is able to produce 5.07 cost value using Nh6 and with 3000 iterations producing 4.94 with Nh7. Comparing *modified*-GDA and *Dueck*-GDA with 1500 iterations (Table 3), the *modified*-GDA is able to produce a solution that is 11% (5.07 compared with 4.53) better than *Dueck*-GDA with Nh1 using an initial solution of 7.82. Even though with a poorer initial cost of 16.68, the *modified*-GDA is able to outperform *Dueck*-GDA by 20% (6.85 compared with 5.51) with Nh1. Extending the search to 3000 iterations, with initial costs of 7.82 and 16.68, Nh1 produced solutions with a 19% (4.94 compared with 4.01) and 25% (6.61 compared with 4.99) improvement when compared to *Dueck*-GDA. The best values found by each of the methods described above is shown in Figure 2.

Overall, the proposed *modified*-GDA gives an improvement when compared to the UMP proprietary software, the constructive heuristic and *Dueck*-GDA. From these results it appears that using a better quality initial cost outperforms both the UMP proprietary software and the constructive heuristic, but using a poorer quality initial solution, the *modified*-GDA does not guarantee to produce high-quality solutions—even for *Dueck*-GDA (when compared to the constructive heuristic with a candidate list of five).

8.2. Semester 1-2008/09

8.2.1. Modified-GDA versus UMP proprietary software. In semester 1-2008/09, the calculated UMP solution was 26.08, with a violation of all of the hard constraints (Kahar and Kendall, 2010). The *modified*-GDA, with 1500 iterations, the solution produced is 77% (26.08 compared with 6.11) better compared to the proprietary software solution (and the solution adheres to all the hard constraints) using Nh1 with an initial cost of 9.21. Increasing the number of iterations to 3000, the solution

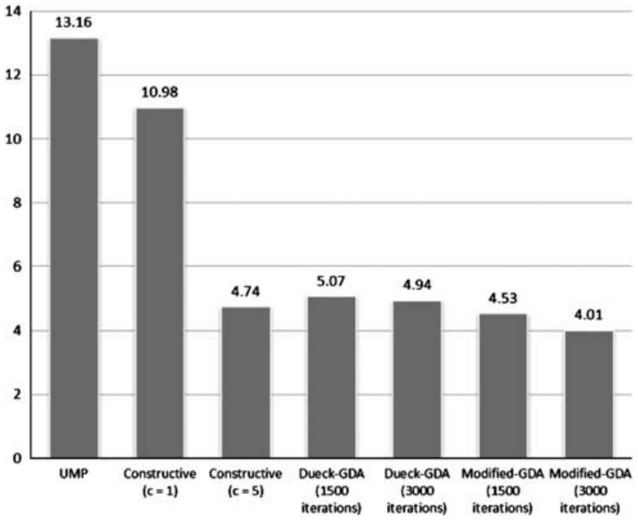

Figure 2 Best values of each method for semester 1-2007/08.

produced with Nh1, using an initial solution of 9.21, is 78% (26.08 compared with 5.63) better than the proprietary software and 9% (6.11 compared with 5.63) better compared to using 1500 iterations.

8.2.2. Modified-GDA versus constructive heuristic. In the constructive heuristic (Kahar and Kendall, 2010), the minimum solution produced is 13.89 and 6.61 using candidate lists of one and five, respectively. In a comparison between *modified*-GDA and the constructive heuristic with a candidate list of one, the *modified*-GDA with 1500 iterations produced a 56% (13.89 compared with 6.11) better solution with Nh1 using an initial cost of 9.21. Even with a poorer initial cost (18.40), the GDA solution is 46% (13.98 compared with 7.12) better using Nh3. Extending the search to 3000 iterations, when using an initial cost of 9.21 and 18.40, Nh1 produces 59% (13.89 compared with 5.63) and 51% (13.89 compared with 6.78), respectively, better solutions compared to the constructive heuristic.

Comparing the *modified*-GDA result with the constructive heuristic with a candidate list of five, *modified*-GDA with 1500 iterations outperforms the constructive heuristic by 8% (6.61 compared with 6.11). However, using a poorer initial cost (18.40), the constructive heuristic outperforms *modified*-GDA by 7% (7.12 compared with 6.61). In *modified*-GDA with 3000 iterations, it produces a 15% (6.61 compared with 5.63) better solution compared to the constructive heuristic. However, with the poorer initial cost (18.40), the constructive heuristic outperforms *modified*-GDA by just under 3% (6.78 compared with 6.61).

8.2.3. Modified-GDA versus Dueck-GDA. For *Dueck*-GDA, with 1500 iterations it is able to produce a solution of 7.20 using Nh9 and with 3000 iterations produces 6.39 with Nh7 (see Table 4). Comparing *modified*-GDA and *Dueck*-GDA with 1500 iterations (Table 4), the *modified*-GDA is able to produce a solution that is 15% (7.20 compared with 6.11) better than *Dueck*-GDA with Nh1 using an initial solution of 7.82. With a poorer initial cost of 16.68, the *modified*-GDA is able to outperform *Dueck*-GDA by 20% (9.48 compared with 7.12) with Nh3. Extending the search to 3000 iterations, with initial costs of 7.82 and 16.68, *modified*-GDA with Nh1 produced solutions with a 19% (6.39 compared with 5.63) and 27% (9.28 compared with 6.78) improvement when compared to *Dueck*-GDA. The best value found by each of the methods described above is shown in Figure 3.

Overall, our proposed *modified*-GDA is able to generate superior solutions than the UMP proprietary software, the constructive heuristic (see Kahar and Kendall, 2010) and *Dueck*-GDA. On the basis of the result from both data sets, it shows that using a good quality, initial solution will

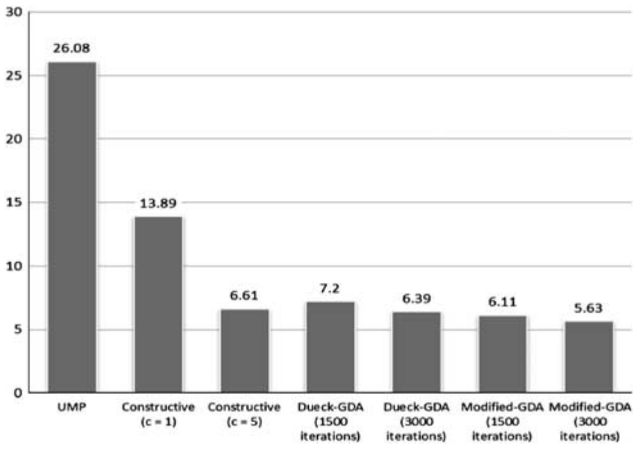

Figure 3 Best values of each method for semester 1-2008/09.

produce superior results, and possibly even better when using a larger number of iterations.

In the next section, we will further analyse the results.

9. Statistical analysis

This section presents a statistical analysis of our results. The aim is to compare the *modified*-GDA and *Dueck*-GDA as well as the parameters used in the experiments to ascertain whether there are statistical differences. In addition, we will determine suitable parameter values and neighbourhood heuristics. The comparisons include:

(a) Compare different initial solutions: Is there any significant difference in using an initial solution with a higher cost than using a better quality initial solution?
(b) Compare the number of iterations: Is there any significant difference in using a larger number of iterations?
(c) Compare neighbourhood heuristics: Is there any significant difference in the result by using different neighbourhood heuristics?

Note that the analyses in (a)–(c) concentrate on the *modified*-GDA only.

We are conscious that some of these may seem intuitively obvious (eg increasing the number of iterations produces superior results), but it is still informative to do the analysis as it is often not carried out.

The statistical tests are carried out using *t*-test, one-way ANOVA, Games Howell *post hoc*, Kruskal–Wallis or Mann–Whitney U tests to determine if there are significant differences. The hypotheses to be tested are: null hypothesis H_0 assumes that the samples are from identical populations, and the alternative hypothesis H_1 assumes that the sample comes from different populations. We reject H_0 when $p \leqslant 0.05$ and vice versa. The above hypotheses are used throughout the statistical tests described in the following section. The t-test and Mann–Whitney U test are used to compare two samples, while one-way ANOVA and Kruskal–Wallis tests are used to compare more than two samples. The Games Howell *post hoc* test is used in conjunction with one-way ANOVA to investigate the cause of H_0 rejection. The Games Howell *post hoc* test compares more than one pair of samples simultaneously. In addition, the Mann–Whitney U test is used to investigate the cause of rejection of H_0 in conjunction with the Kruskal–Wallis test.

The t-test and one-way ANOVA are used with a normally distributed sample, while the Kruskal–Wallis and Mann–Whitney U tests are used for a non-normally distributed sample. The normality tests are carried out using the Shapiro–Wilk test with the null hypothesis H_0 assuming that the samples are normally distributed, and the alternative hypothesis H_1 assuming that the sample is non-normal. We reject H_0 when $p \leqslant 0.05$ and vice versa.

We start the statistical test with a normality test using the Shapiro–Wilk test and then continue with the relevant statistical test (as described above) based on the normality test result.

9.1. Semester1-2007/08

9.1.1. Significance difference: modified-GDA and Dueck-GDA. We analyse the *modified*-GDA and *Dueck*-GDA results using t-test and the Mann–Whitney U test. Table 5 and Table 6 show the p-value result for 1500 iterations and 3000 iterations, respectively. For 1500 iterations (see Table 5), we notice that most of the results show significant difference except for the Nh2 (all initial), Nh5 (all initial), Nh6 (16.68), Nh8 (10.30 and 7.82) and Nh10 (7.82). For 3000 iterations (see Table 6), again, most of the results show significant difference except for Nh2 (all initial), Nh5 (all initial),

Table 5 Semester1-2007/08 p-value comparison between *modified*-GDA and *Dueck*-GDA for every neighbourhood heuristics with 1500 iterations

Neighbourhood heuristics	*Initial cost*			
	16.68	13.74	10.30	7.82
Nh1	0.000^{MwU}	0.000^{MwU}	0.000^{t}	0.000^{MwU}
Nh2	0.694^{MwU}	0.221^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	0.000^{MwU}	0.000^{t}	0.000^{MwU}	0.000^{MwU}
Nh4	0.000^{t}	0.000^{t}	0.000^{t}	0.000^{t}
Nh5	1.00^{MwU}	0.385^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	0.299^{MwU}	0.037^{MwU}	0.009^{MwU}	0.000^{t}
Nh7	0.000^{t}	0.000^{MwU}	0.000^{t}	0.000^{MwU}
Nh8	0.000^{t}	0.000^{t}	0.466^{t}	0.900^{MwU}
Nh9	0.000^{t}	0.005^{t}	0.000^{t}	0.000^{t}
Nh10	0.000^{t}	0.000^{MwU}	0.015^{t}	0.251^{MwU}

MwU = Mann–Whitney U; t = t-test.

Table 6 Semester1-2007/08 p-value comparison between *modified*-GDA and *Dueck*-GDA for every neighbourhood heuristics with 3000 iterations

Neighbourhood heuristics	*Initial cost*			
	16.68	13.74	10.30	7.82
Nh1	0.000^{t}	0.000^{MwU}	0.000^{t}	0.000^{t}
Nh2	0.019^{MwU}	0.427^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	0.000^{t}	0.000^{t}	0.000^{MwU}	0.000^{MwU}
Nh4	0.000^{t}	0.000^{t}	0.000^{t}	0.000^{t}
Nh5	1.00^{MwU}	0.688^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	0.018^{t}	0.115^{MwU}	0.024^{MwU}	0.216^{t}
Nh7	0.000^{t}	0.000^{t}	0.000^{MwU}	0.034
Nh8	0.000^{t}	0.503^{t}	0.509^{t}	0.296
Nh9	0.000^{t}	0.000^{t}	0.000^{t}	0.000^{t}
Nh10	0.000^{t}	0.375^{t}	0.013^{MwU}	0.652^{MwU}

MwU = Mann–Whitney U; t = t-test.

Nh6 (13.74, 7.82), Nh8 (13.74, 10.30, 7.82) and Nh10 (13.74, 7.82).

On the basis of both of the runs, generally the result that shows no significant difference involves a neighbourhood heuristic that performs poorly.

9.1.2. Comparing initial costs. We compare the initial cost based on the number of iterations for all neighbourhood heuristics. We use one-way ANOVA (for normally distributed data) and Kruskal–Wallis tests (for non-normal data) to compare between the initial costs (ie 16.68, 13.74, 10.30 and 7.82). At the 95% confidence interval, the statistical test shows that there exists enough evidence to conclude that there is a difference (reject H_0) among the results produced between the initial costs for all of the neighbourhood heuristics (see Table 7). Referring to Table 7, the *p*-values are all less than 0.05, which leads us to reject H_0.

In-depth analyses on the differences in pair (16.68 with 13.74, 10.30, 7.82; 13.74 with 10.30, 7.82 and so on) were investigated using the Games Howell *post hoc* (in conjunction with one-way ANOVA) and Mann–Whitney U tests (in conjunction with the Kruskal–Wallis test). On the basis of the analyses, only a few of the initial costs show no differences (accept H_0), which include:

- Nh3 between 10.30 and 7.82 for both iterations counts.
- Nh4 between 16.68 and 13.74 for both iterations counts.
- Nh6 between 16.68 and 13.74 with 3000 iterations.
- Nh8 between 16.68 and 13.74 for both iterations counts.
- Nh9 between 16.68 and 13.74 with 1500 iterations.
- Nh9 between 10.30 and 7.82 with 3000 iterations.
- Nh10 between 16.68 and 13.74 with 3000 iterations

Generally, the results that show no difference (accept H_0) involve using a solution with a large initial cost. From this analysis we conclude that it is beneficial to start with the best quality solution possible.

9.1.3. Comparing the number of iterations. We compare the number of iterations (1500 and 3000 iterations) based on the initial cost (ie 1500 *versus* 3000 with an initial cost of 16.68, 13.74, 10.30 and 7.82) using either *t*-test or the Mann–Whitney U test, depending on whether the data are normally distributed. Table 8 shows the *p*-value of the comparison between the number of iterations executed. At the 95% confidence interval, the result is as follows (see Table 8):

- Nh1 shows significant difference (reject H_0) across all initial costs.
- Nh3 and Nh7 show significant differences (reject H_0) for all initial costs except for 10.30 (accept H_0).
- Nh2, Nh4 and Nh8 show no significant differences (accept H_0) across all initial costs.
- Nh5 and Nh6 shows no significant differences (accept H_0) for all initial costs except during initial 13.74 (reject H_0).
- Nh9 show no significant differences (accept H_0) for all initial costs except for 13.74 and 10.30 (reject H_0).
- Nh10 show no significant differences (accept H_0) for all initial costs except during 10.30 (reject H_0).

On the basis of these tests, the result varies according to the neighbourhood heuristics. We notice that a diversified neighbourhood heuristics (ie Nh1 and Nh7) shows significance difference (reject H_0) between the two iterations compared to undiversified neighbourhood (ie Nh2, Nh5 etc). Therefore (considering the solution in Table 3), we conclude that it is best to use a large number of iterations. However, a search with a large

Table 7 Semester 1-2007/08 *p*-value comparison for the initial cost for each neighbourhood heuristic based on the number of iterations

Neighbourhood heuristics	*p-value*	
	1500 *iterations*	3000 *iterations*
Nh1	0.000^{kw}	0.000^{owA}
Nh2	0.000^{kw}	0.000^{kw}
Nh3	0.000^{kw}	0.000^{kw}
Nh4	0.000^{owA}	0.000^{owA}
Nh5	0.000^{kw}	0.000^{kw}
Nh6	0.000^{kw}	0.000^{kw}
Nh7	0.000^{kw}	0.000^{kw}
Nh8	0.000^{kw}	0.000^{owA}
Nh9	0.000^{owA}	0.000^{owA}
Nh10	0.000^{kw}	0.000^{kw}

owA = one-way ANOVA; *kw = Kruskal–Wallis.*

Table 8 Semester 1-2007/08 *p*-value comparison between 1500 and 3000 iterations for each neighbourhood heuristic based on initial cost

Neighbourhood heuristics	*Initial cost*			
	16.68	13.74	10.30	7.82
Nh1	0.000^{MwU}	0.000^{t}	0.000^{t}	0.000^{MwU}
Nh2	0.087^{MwU}	0.340^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	0.000^{MwU}	0.000^{t}	0.051^{MwU}	0.001^{t}
Nh4	0.090^{t}	0.141^{t}	0.755^{t}	0.992^{t}
Nh5	1.00^{MwU}	0.038^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	0.860^{MwU}	0.044^{MwU}	0.524^{MwU}	0.073^{t}
Nh7	0.000^{t}	0.000^{MwU}	0.456^{t}	0.000^{MwU}
Nh8	0.273^{t}	0.817^{t}	0.624^{t}	0.589^{MwU}
Nh9	0.129^{t}	0.002^{t}	0.040^{t}	0.692^{t}
Nh10	0.065^{t}	0.303^{MwU}	0.005^{MwU}	0.172^{MwU}

MwU = Mann–Whitney U; *t = t-test.*

number of iterations would only be worthwhile if it is being complemented with a good neighbourhood heuristic (to encourage exploration).

9.1.4. Comparing neighbourhood heuristics. We compare all the neighbourhood heuristics based on the initial cost and number of iterations using the Kruskal–Wallis test (ie Nh1 *versus* Nh2 *versus* Nh3 *versus* ... Nh10 using initial cost 16.86 with 1500 iterations, etc). Table 9 shows the *p*-values of the neighbourhood heuristics comparison. The result shows that there are significant differences (reject H_0) for the solutions produced using different neighbourhood heuristics.

Pair-wise comparison (analysis on the cause of H_0 rejection) using the Mann–Whitney U test on the neighbourhood heuristics shows that there are significant differences (reject H_0) for the solution produced by most of the neighbourhood heuristics, except for some. For example, Nh2 and Nh5 show no difference with initial costs of 7.82 and 10.30 for both iterations and an initial cost of 16.68 using 3000 iterations. Table 10 shows a summary of the non-significant differences (accept H_0) between the neighbourhood heuristics. Referring to Table 10, we notice that some of the neighbourhoods (ie Nh3 and Nh4, Nh4 and Nh7) show similarity, although the inner working of the heuristics are different.

Table 9 Semester 1-2007/08 *p*-value comparison for the neighbourhood heuristics based on the initial cost and the number of iterations

Initial value	1500 *iterations*	3000 *iterations*
16.68	0.000	0.000
13.74	0.000	0.000
10.30	0.000	0.000
7.82	0.000	0.000

Finally, we can summarise that Nh1 produces the best result followed by Nh7 and Nh4. Next are Nh3, Nh9, Nh6, Nh8, Nh10, Nh2 and Nh5. In our observation, Nh1 is a robust neighbourhood heuristic. Nh2 and Nh5 are the worst neighbourhood heuristics as they are unable to give any improvement on the initial cost during the search (especially Nh5). Nh7 works best with a better quality initial cost, while Nh4 works best with a large initial cost. Further discussion on the neighbourhood heuristics is given in Section 11.

9.2. Semester1-2008/09

9.2.1. Significance difference: modified-GDA and Dueck-GDA. As in the previous section (9.1.1), we used *t*-test and the Mann–Whitney U test to analyse the result. Table 11 and Table 12 show the *p*-value result for 1500 iterations and 3000 iterations, respectively. For 1500 iterations (see

Table 11 Semester1-2008/09 *p*-value comparison between *modified*-GDA and *Dueck*-GDA for every neighbourhood heuristics with 1500 iterations

Neighbourhood heuristics	*Initial cost*			
	18.4	15.25	12.30	9.21
Nh1	0.000^{t}	0.000^{MwU}	0.000^{t}	0.000^{MwU}
Nh2	0.080^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	0.000^{MwU}	0.000^{t}	0.000^{MwU}	0.000^{MwU}
Nh4	0.000^{t}	0.000^{t}	0.000^{t}	0.000^{t}
Nh5	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	0.074^{MwU}	0.879^{t}	0.467^{t}	0.000^{MwU}
Nh7	0.000^{t}	0.000	0.000^{MwU}	0.000^{t}
Nh8	0.000^{t}	0.466^{t}	0.476^{t}	0.734^{MwU}
Nh9	0.000^{t}	0.000^{t}	0.000^{t}	0.436^{t}
Nh10	0.000^{t}	0.003^{t}	0.844^{MwU}	0.330^{MwU}

MwU = Mann–Whitney U; *t = t-test.*

Table 10 Semester 1-2007/08 summary of the non-significant differences (accept H_0) when comparing the neighbourhood heuristics

	1500 *iterations*				3000 *iterations*			
	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82
Nh1	—	—	—	—	—	—	—	—
Nh2	—	—	Nh5	Nh5	Nh5	—	Nh5	Nh5
Nh3	—	Nh4	Nh4	—	Nh4, Nh7	—	Nh4, Nh7	—
Nh4	—	—	Nh7	—	Nh7	—	Nh7	—
Nh5	—	—	—	—	—	—	—	—
Nh6	—	Nh8, Nh9	—	—	Nh9	Nh9	—	Nh9
Nh7	—	—	—	—	—	—	—	—
Nh8	—	Nh10	—	Nh10	—	Nh10	Nh10	Nh10
Nh9	—	—	—	—	—	—	—	—
Nh10	—	—	—	—	—	—	—	—

'—' = result show rejecting H_0.

Table 12 Semester1-2008/09 p-value comparison between *modified*-GDA and *Dueck*-GDA for every neighbourhood heuristics with 3000 iterations

Neighbourhood heuristics	*Initial cost*			
	18.4	15.25	12.30	9.21
Nh1	0.000^{t}	0.000^{t}	0.000^{MwU}	0.000^{t}
Nh2	0.600^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	0.000^{t}	0.000^{MwU}	0.000^{t}	0.000^{MwU}
Nh4	0.000^{t}	0.000^{t}	0.000^{MwU}	0.000^{t}
Nh5	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	0.035^{MwU}	0.001^{t}	0.463^{t}	0.000^{MwU}
Nh7	0.000^{t}	0.000^{MwU}	0.000^{MwU}	0.000^{MwU}
Nh8	0.000^{t}	0.473^{MwU}	0.196^{t}	0.474^{MwU}
Nh9	0.000^{t}	0.000^{t}	0.164^{t}	0.144^{MwU}
Nh10	0.000^{t}	0.995^{t}	0.177^{MwU}	0.288^{MwU}

MwU = Mann–Whitney U; *t = t-test.*

Table 11), we notice that most of the result shows significant difference except for the Nh2 (all initial), Nh5 (all initial), Nh6 (18.40, 15.25 and 12.30), Nh8 (15.25, 12.30 and 9.21), Nh9 (9.21) and Nh10 (12, 30 and 9.21).

In 3000 iterations (see Table 12), most of the result show significant difference except for Nh2 (all initial), Nh5 (all initial), Nh6 (12.30), Nh8 (15.25, 12.30 and 9.21), Nh9 (12.30 and 9.21) and Nh10 (15.25, 12.30 and 9.21).

On the basis of the results, the semester1-2008/09 data set shows more non-significant difference compared to semester1-2007/08. However, the result that shows non-significant difference mainly involves a neighbourhood heuristic that performs poorly (same as in semester1-2007/08 result).

9.2.2. Comparing initial costs. We compare the initial cost based on the number of iterations for all neighbourhood heuristics for the semester1-2008/09 data set. As in Section 9.1.2, we used one-way ANOVA (normally distributed data) and the Kruskal–Wallis test (non-normal data) to compare between the initial costs (ie 18.40, 15.25, 12.30 and 9.21). Referring to Table 13, at the 95% confidence interval, there are significant differences on all of the results as the p-values are all less than 0.05 (reject H_0). In a pair-wise comparison between each initial cost using the Games Howell *post hoc* and Mann–Whitney U tests, the result shows that only a few of the initial cost shows no significant differences (accept H_0), which include:

- Nh3 between 18.40 and 9.21 with 3000 iterations.
- Nh6 between 18.40 and 15.25 using 1500 iteration.
- Nh8 between 15.25 and 12.30 using 3000 iteration.

On the basis of the results, the majority of the neighbourhood heuristics show significant differences

Table 13 Semester1-2008/09 p-value comparison for the initial cost for each neighbourhood heuristic based on the number of iterations

Neighbourhood heuristics	*p-value*	
	1500 *iterations*	3000 *iterations*
Nh1	0.000^{kw}	0.000^{owA}
Nh2	0.000^{kw}	0.000^{kw}
Nh3	0.000^{kw}	0.000^{owA}
Nh4	0.000^{owA}	0.000^{kw}
Nh5	0.000^{kw}	0.000^{kw}
Nh6	0.000^{kw}	0.000^{kw}
Nh7	0.000^{kw}	0.000^{kw}
Nh8	0.000^{kw}	0.000^{kw}
Nh9	0.000^{owA}	0.000^{kw}
Nh10	0.000^{kw}	0.000^{kw}

owA = one-way ANOVA; *kw = Kruskal–Wallis.*

Table 14 Semester1-2008/09 p-value comparison between 1500 and 3000 iterations for each neighbourhood heuristic based on initial cost

Neighbourhood heuristics	*Initial cost*			
	18.40	15.25	12.30	9.21
Nh1	0.000^{t}	0.000^{t}	0.000^{t}	0.000^{MwU}
Nh2	0.907^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	0.000^{MwU}	0.000^{t}	0.001^{t}	0.000^{t}
Nh4	0.020^{t}	0.177^{t}	0.124^{MwU}	0.811^{t}
Nh5	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	0.000^{MwU}	0.094^{t}	0.992^{t}	0.029^{MwU}
Nh7	0.000^{t}	0.000^{MwU}	0.000^{MwU}	0.000^{t}
Nh8	0.000^{t}	0.000^{MwU}	0.742^{t}	0.757^{MwU}
Nh9	0.035^{t}	0.744^{t}	0.155^{t}	0.322^{MwU}
Nh10	0.033^{t}	0.084^{t}	0.450^{MwU}	0.752^{MwU}

MwU = Mann–Whitney U; *t = t-test.*

(accept H_0) and considering the result in Table 4, it is best to start with a good quality solution and thus reaffirm our conclusions in Section 9.1.2.

9.2.3. Comparing the number of iterations. As in Section 9.1.3, we compare the solution for the number of iterations (1500 and 3000 iterations) based on the initial cost (ie 1500 *versus* 3000 with an initial cost of 18.40, 15.25, 12.30 and 9.21). Table 14 shows the p-value of the comparison between the number of iterations. At the 95% confidence interval, the result is as follows (see Table 8):

- Nh1, Nh3 and Nh7 show significant differences (reject H_0) across all initial costs.
- Nh2 and Nh5 show no differences (accept H_0) in the result for all initial costs.

- Nh4, Nh9 and Nh10 show significant differences (reject H_0) only on initial costs 18.40.
- Nh6 show significant difference (reject H_0) only on initial costs 18.40 and 9.21.
- Nh8 show significant difference (reject H_0) only on initial costs 18.40 and 15.25.

The results show a similar pattern as for semester1-2007/08 and this reaffirms our conclusion (as in the previous data set) that is best to use a larger number of iterations.

9.2.4. Comparing neighbourhood heuristics. As in Section 9.1.4, we compare the set of neighbourhood heuristics based on the initial costs and the number of iterations using the Kruskal–Wallis test. Table 15 shows the *p*-value of the neighbourhood heuristics comparison. At the 95% confidence interval, the statistical result shows that there are significant differences (reject H_0) for the solutions produced between the neighbourhood heuristics. An in-depth analysis using the Mann–Whitney U test shows that there are significant differences (reject H_0) for the solutions produced by most of the neighbourhood heuristics. Table 16 summarises the significant differences (accept H_0) between neighbourhoods. Hence, we can summarise that Nh1 produced the best result, followed by Nh7 and Nh3. Next are Nh4, Nh9, Nh10, Nh8, Nh6, Nh2 and Nh5. Again, Nh1 is the best heuristic and Nh5 is the worst.

Overall, we can conclude that it is advisable to use the best quality solution as the initial solution and a larger number of iterations. In terms of neighbourhood heuristics, the results vary according to the neighbourhood heuristics, and performance is different between the two data sets. Hence, a neighbourhood that works for one data set might not necessarily work on other data set. Therefore, it is best to use a set of diversified neighbourhood heuristics (eg Nh1 and Nh7) as it will encourage exploration of the search space.

Table 15 Semester1-2008/09 *p*-value comparison for the neighbourhood heuristic based on initial cost and the number of iterations

Initial value	1500 *iterations*	3000 *iterations*
18.40	0.000	0.000
15.25	0.000	0.000
12.30	0.000	0.000
9.21	0.000	0.000

10. Discussion

The proposed GDA gives an improvement over the constructive heuristic and outperforms the UMP proprietary software. The success of the technique is because of its dynamic acceptance level that uses a boundary level that gradually decreases based on a decay rate, but also allows the boundary to increase when there is no improvement during search. In increasing the boundary level, the new boundary is set higher than the current solution $f(s)$ allowing the search to accept worse solutions. The algorithm also adjusts the boundary and a newly desired value is calculated when $f(s)$ is less than or equal to the desired value.

A comparison between *modified*-GDA and *Dueck*-GDA reveals that the *modified*-GDA is able to produce better quality solutions than *Dueck*-GDA. Some of the neighbourhood heuristics show non-significant difference. However, this mainly involves neighbourhood heuristics that perform poorly.

The *modified*-GDA gives an improvement over the initial cost (both 1500 and 3000 iterations) for the majority of the neighbourhood heuristics. Statistical analysis on the initial cost shows that some neighbourhoods (eg Nh3, Nh6, Nh8

Table 16 Semester1-2008/09 summary of non-significant differences (accept H_0) when comparing neighbourhood heuristics

	1500 *iterations*				3000 *iterations*			
	18.40	15.25	12.30	9.21	18.40	15.25	12.30	9.21
Nh1	—	—	—	—	—	—	—	—
Nh2	—	Nh5	Nh5	Nh5	—	Nh5	Nh5	Nh5
Nh3	—	—	—	Nh4, Nh9	—	—	—	—
Nh4	—	—	Nh9	Nh9	—	—	Nh9	Nh9
Nh5	—	—	—	—	—	—	—	—
Nh6	Nh8, Nh9, Nh10	—	Nh10	—	Nh8, Nh10	—	Nh8	—
Nh7	—	—	—	—	—	—	—	—
Nh8	Nh9	Nh10	Nh10	—	Nh10	Nh9	Nh10	
Nh9	—	—	—	—	—	—	—	—
Nh10	—	—	—	—	—	—	—	—

'—' = result show rejecting H_0.

etc) have similar performance, mostly between large initial costs, where semester 1-2007/08 shows more similarity compared to semester 1-2008/09. Only a few show similarity on a small initial cost (ie Nh3 and Nh9), which we believe is caused by the neighbourhood heuristics themselves. The neighbourhood heuristic is unable to diversify the search when it is near to a local optima. Referring to Tables 3 and 4, we can summarise that using a smaller initial cost produces a higher-quality solution when compared to using a larger initial cost. However, note that the computational time to find a small initial cost takes a bit longer during the constructive phase (Kahar and Kendall, 2010).

An analysis on the number of iterations reveals that some of the neighbourhoods (ie Nh2, Nh5 and Nh6) show no difference in their performance between the numbers of iterations. We notice that the result is very much dependent on the heuristics used. A diversified neighbourhood would make use of the large number of iterations to efficiently explore the search space. This led us to conclude that the number of iterations does play a role in the search but it is not as important as the neighbourhood heuristics that are used. Using a larger number of iterations gives better results because it enables the method to cover more of the search space, compared to small number of iterations. However, this does require extra computational time. A good compromise is to use a small initial cost with a large number of iterations.

An analysis on the neighbourhood heuristics shows that Nh1 is the best and Nh5 is the worst. The result also shows that the neighbourhood heuristics perform differently between the two data sets (except for the first and the last two neighbourhoods), although the data sets are similar in terms of the characteristics (see Table 2). In our observation, Nh1 (which produces the best result) is a robust neighbourhood heuristic (see Tables 3 and 4). Nh2 and Nh5 are the worst neighbourhood heuristics as it is unable to improve the initial cost except for an initial cost 13.74 on the semester 1-2007/08 data set. The result demonstrates the importance of the initial cost in order for the search to advance. Nh7 works best with a small initial cost while Nh3, Nh4 and Nh6 work best with large initial cost. Hence, we can conclude that the choice of neighbourhood heuristics is very important in the search in order to converge to a good quality solution (Thompson and Dowsland, 1998) in addition to a good choice of initial solution and number of iterations.

12. Conclusion and future research

In this paper, we have investigated a real-world examination timetabling problem aiming to improve on the constructive heuristic solution. The *modified*-GDA approach is able to produce good quality solutions compared to the UMP proprietary software, satisfying all the constraints (which the proprietary software fails to do), improve on the constructive result and perform better than the *Dueck*-GDA. The proposed *modified*-GDA uses a simple to determine parameter that can find a good solution. The selection of neighbourhood heuristics, iterations and initial cost plays a significant part in the search.

For future work, our aims are to further explore the *modified*-GDA approach:

- Due to the fact that the neighbourhood heuristics are very important, we are going to investigate the use of multiple neighbourhood. We are going to use each neighbourhood in succession. The next neighbourhood will be selected if the current neighbourhoods show no improvement.
- Use all of the neighbourhood heuristics in every iteration.
- Combine different neighbourhood heuristics. For example, using Nh4 or Nh6 in early stage of the search and then Nh1 or Nh7 in the latter stages. We believe this could save computational time if suitable neighbourhoods are combined in an intelligent manner.

Acknowledgements—The examination/invigilator data set has been provided by the Academic Management Office (AMO), UMP and the research has been supported by the Public Services Department of Malaysia (JPA) and the Universiti Malaysia Pahang (UMP).

References

Abdullah S (2006). *Heuristic approaches for university timetabling problems*. PhD Thesis, School of Computer Science and Information Technology, University of Nottingham, June 2006.

Abdullah S, Burke EK and McCollum B (2005). An investigation of variable neighbourhood search for university course timetabling. In: *Proceedings of MISTA 2005: The 2nd Multidisciplinary Conference on Scheduling: Theory and Applications*. 18–21 July, pp 413–427.

Abdullah S, Shaker K, McCollum B and McMullan P (2009a). Construction of course timetables based on great Deluge and Tabu search. In: *Proceedings of MIC 2009: VIII Metaheuristic International Conference*, 13–16 July, Hamburg, Germany.

Abdullah S, Turabieh H and McCollum B (2009b). A tabu-based memetic approach for examination timetabling problems. In: *AI-2009 Twenty-ninth SGAI International Conference on Artificial Intelligence*, 15–17 December, Cambridge, England.

Burke EK and Bykov Y (2008). 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*, 19–22 August, Universit de Montral, Montreal, Canada.

Burke EK and Carter MW (eds) (1998). *Second International Conference on Practice and Theory of Automated Timetabling II, PATAT '97*. Toronto, Canada, 20–22 August, Selected Papers. Lecture Notes in Computer Science, Vol. 1408. Springer-Verlag: Berlin Heidelberg.

Burke EK and De Causmaecker P (eds) (2003). *Fourth International Conference on Practice and Theory of Automated Timetabling IV, (PATAT 2002)*. Gent, Belgium, 21–23 August 2002, Selected Revised Papers. Lecture Notes in Computer Science, Vol. 2740. Springer-Verlag: Berlin Heidelberg.

Burke EK and Erben W (eds) (2001). *Third International Conference on Practice and Theory of Automated Timetabling III, (PATAT 2000)*. Konstanz, Germany, 16–18 August 2000, Selected Papers. Lecture Notes in Computer Science, Vol. 2079. Springer-Verlag: Berlin Heidelberg.

Burke EK and Kendall G (eds) (2005). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer.

Burke EK and Newall J (2003). Enhancing timetable solutions with local search methods. In: Burke EK and De Causmaecker P (eds). *Selected Papers from the 4th International Conference on Practice and Theory of Automated Timetabling (PATAT 2002)*. Gent, Belgium, 21–23 August 2002. Lecture Notes in Computer Science, Vol. 2740. 2740, Springer-Verlag, Berlin Heidelberg, pp. 195–206.

Burke EK and Ross P (eds) (1996). *First International Conference on Practice and Theory of Automated Timetabling*. Edinburgh, UK, 29 August – 1 September 1995, Selected Papers. Lecture Notes in Computer Science, Vol. 1153. Springer-Verlag: Berlin Heidelberg.

Burke EK and Rudova H (eds) (2007). *Sixth International Conference on Practice and Theory of Automated Timetabling VI, (PATAT 2006)*. Brno, Czech Republic, 30 August – 1 September 2006, Revised Selected Papers. Lecture Notes in Computer Science, Vol. 3867. Springer-Verlag: Berlin Heidelberg.

Burke EK and Trick M (eds) (2005). *Fifth International Conference on Practice and Theory of Automated Timetabling V, (PATAT 2004)*. Pittsburgh, PA, USA, 18–20 August 2004, Revised Selected Papers. Lecture Notes in Computer Science, Vol. 3616. Springer-Verlag: Berlin Heidelberg.

Burke EK, Newall J and Weare RF (1996a). A memetic algorithm for university exam timetabling. In: Burke EK and Ross P (eds). *Selected Papers from the First International Conference on Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science, Vol. 1153, Springer-Verlag, pp 241–250.

Burke EK, Elliman DG, Ford PH and Weare RF (1996b). Examination timetabling in British universities – A survey. In: Burke EK and Ross P (eds). *Selected Papers from First International Conference on the Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science, Edinburgh, UK, Vol. 1153. Springer-Verlag: Berlin Heidelberg, pp 76–92.

Burke EK, Eckersley A, McCollum B, Petrovic S and Qu R (2003). Using simulated annealing to study behaviour of various exam timetabling data sets. In: *Proceedings of Metaheuristic International Conference 2003 (MIC 2003)*, 25–28 August, Kyoto, Japan.

Burke EK, Bykov Y, Newall JP and Petrovic S (2004). A time-predefined local search approach to exam timetabling problems. *IIE Transactions* **36**(6): 509–528.

Burke EK, McCollum B, Meisels A, Petrovic S and Qu R (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research* **176**(1): 177–192.

Burke EK, Eckersley AJ, McCollum B, Petrovic S and Qu R (2010a). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research* **206**(1): 46–53.

Burke EK, Hyde M, Kendall G, Ochoa G, Ozcan E and Qu R (2010b). Hyper-heuristics: A survey of the state of the art, School of Computer Science and Information Technology. University of Nottingham, Computer Science Technical Report No. NOTTCS-TR-SUB-0906241418-2747.

Carter MW and Laporte G (1996). Recent developments in practical examination timetabling. In: Burke EK and Ross P (eds). *Selected Papers from the First International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*. Lecture Notes in Computer Science, Edinburgh, UK, Vol. 1153, Springer-Verlag: Berlin Heidelberg, pp 3–21.

Carter MW, Laporte G and Lee SY (1996). Examination timetabling: Algorithmic strategies and applications. *Journal of Operational Research Society* **47**(3): 373–383.

Cowling P, Kendall G and Hussin NM (2002). A survey and case study of practical examination timetabling problems. In: *Proceedings of the 4th international Conference on the Practice and Theory of Automated Timetabling (PATAT 2002)*. Gent, Belgium, 21–23 August, pp 258–261.

Di Gaspero L and Schaerf A (2001). Tabu search techniques for examination timetabling. In: Burke EK and Erben W (eds). *Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science, Vol. 2079. Springer-Verlag: Berlin Heidelberg, pp 104–117.

Dorigo M, Maniezzo V and Colorni A (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B* **26**(1): 29–41.

Dueck G (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics* **104**(1): 86–92.

Eley M (2006). Some experiments with ant colony algorithms for the exam timetabling problem. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 4150, LNCS, pp 492–499.

Eley M (2007). Ant algorithms for the exam timetabling problem. In: Burke EK and Rudova H (eds). *Selected Papers from the 6th International Conference on Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science, 30 August–1 September, Vol. 3867, Springer-Verlag: Berlin Heidelberg, pp 364–382.

Glover F (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research* **13**(5): 533–549.

Hansen P and Mladenovic N (1999). An introduction to variable neighborhood search. In: Voß S, Martello S, Roucairol C and Osman IH (eds). *Meta-Heuristics 98: Theory & Applications*. Kluwer Academic Publishers: Norwell, MA, pp 433–458.

Hussin NM (2005). Tabu search based hyper-heuristic approaches to examination timetabling. PhD Thesis, University of Nottingham, UK.

Kahar MNM and Kendall G (2010). The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. *European Journal of Operational Research* **207**(2): 557–565.

Kendall G and Hussin MN (2004). Tabu search hyper-heuristic approach to the examination timetabling problem at University Technology Mara. In: Burke EK and Trick M (eds). *Proceedings of the Fifth International Conference on the Practice and Theory of Automated Timetabling (PATAT V)*. Pittsburgh, USA, 18–20 August, Vol. 3616. Springer-Verlag: Berlin Heidelberg, pp 199–217.

Kirkpatrick S, Gelatt Jr CD and Vecchi MP (1983). Optimization by simulated annealing. *Science* **220**(4598): 671–680.

McCollum B (2007). A perspective on bridging the gap between theory and practice in university timetabling. In: Burke EK and Rudova H (eds). *Selected Papers from the Sixth International Conference on Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science, Vol. 3867, pp 3–23.

McCollum B *et al* (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing* **22**(1): 120–130.

McMullan P (2007). An extended implementation of the great deluge algorithm for course timetabling. Lecture Notes in Computer Science, Vol. 4487. Springer, pp 538–545.

Merlot LTG, Boland N, Hughes BD and Stuckey PJ (2003). A hybrid algorithm for the examination timetabling problem. In: Burke EK and De Causmaecker P (eds). *Selected Papers from Fourth International Conference on Practice and Theory of Automated Timetabling IV*. Lecture Notes in Computer Science, Vol. 2740, Springer-Verlag: Berlin Heidelberg, pp 207–231.

Petrovic S and Burke EK (2004). University timetabling. In: Leung J. (ed). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapter 45, University Timetabling, CRC Press, Taylor & Francis: USA.

Pillay N and Banzhaf W (2009). A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. *European Journal of Operational Research* **197**(2): 482–491.

Qu R, Burke EK, McCollum B, Merlot LTG and Lee SY (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling* **12**(1): 55–89.

Ross P and Corne D (1995). Comparing genetic algorithms, simulated annealing, and stochastic hillclimbing on timetabling problems. In: Fogarty TC (ed). *Evolutionary Computing 2*. Lecture Notes in Computer Science, Vol. 993. Springer-Verlag: Berlin Heidelberg, pp 94–102.

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

Silva DL and Obit JH (2008). Great deluge with non-linear decay rate for solving course timetabling problems. Intelligent Systems (IS'08). 4th International IEEE Conference, Varna, Vol. 1, 6–8 September pp 8–11, 8–18.

Thompson JM and Dowsland KA (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research* **25**(7–8): 637–648.

White GM, Xie BS and Zonjic S (2004). Using tabu search with longer-term memory and relaxation to create examination timetables. *European Journal of Operational Research* **153**(1): 80–91.

Received March 2011;
accepted November 2012 after two revisions