A harmony search algorithm for university course timetabling

Mohammed Azmi Al-Betar[1a]; Ahamad Tajudin Khader[2]; Taufiq Abdul Gani[3]

School of Computer Sciences, Universiti Sains Malaysia, 11800 USM, Pinang, Malaysia[1,2,3]

{Mohbetar; tajudin; taufiq}@cs.usm.my

Abstract One of the main challenges for any university administrations is building a timetable for course sessions. Such a challenge not only relates to how to build a usable timetable but also how to build an optimal timetable at the same time. The university course timetable is classified as an NP-Complete problem. In general, it means assigning predefined courses to certain rooms and timeslots under specific constraints. Harmony search algorithm is a new metaheuristic population based algorithm derived from a natural phenomena of musicians' behavior when they cooperatively play their musical instruments to achieve a fantastic harmony. The major thrust of this algorithm lies in its ability to integrate exploitation and exploration in a parallel optimization environment. In this paper, a harmony search algorithm is applied to university course timetabling against standard benchmarks. The results show that the proposed harmony search is capable of providing a viable solution compared to those in previous works.

Keywords: university course timetabling problem, harmony search algorithm, metaheuristic algorithms, exploration, exploitation.

1 Introduction

The university course timetabling problem is quite common at universities and is considered as an NP-complete problem (Garey and Johnson 1979). In general, it means assigning a set of courses to timeslots and rooms under a set of constraints. Though each university may have different constraints, two types of constraints are commonly considered (Burke et al. 1997): Hard constraints, which must be satisfied in a timetable to be usable (feasible) and soft constraints, which is desired but not absolutely essential. Being not so essential but rather desirable, soft constraints might be violated. Yet, the more they are met, the more the solution can gain qualitatively.

Several algorithms have introduced to solve timetabling problems. The earliest of which depended on graph coloring heuristic methods whereby courses are assigned to rooms and timeslots one by one in a particular order. Although, those algorithms show great efficiency in small timetabling instances, they are mostly not efficient in larger instances.

Later, metaheuristic algorithms came to the fore to solve the timetabling problems. Commonly, metaheuristic algorithms are divided into two types. *The first type* is a local based algorithm which starts with one solution and tries to satisfy the constraints iteratively based on a fitness function until a global optimal

[a] Corresponding author.
Email addresses: mohbetar@cs.usm.my, betar79@gmail.com.
(🖂): School of Computer Sciences, Universiti Sains Malaysia,11800 USM, Pinang, Malaysia.
Tel. No :+60172686566.

solution is reached (ex. simulated annealing (Abramson 1991; Thompson 1996; Tuga et al. 2007),very large neighborhood search (Abdullah et al. 2005a). and so on). The main drawback of those local based algorithms is that they may get stuck in the local optimal solution in which the current solution is considered the best notwithstanding the fact that there are other solutions in different areas of the search space with a better quality. The main cause for the local optimal problem is that local based algorithms focus on exploitation rather than exploration which means that the local based algorithms move in one direction without performing a wider scan of the search space.

The second type of metaheuristic algorithms is a population based algorithm which starts with many different solutions and refines them in a parallel optimisation environment until a global optimal solution is reached. The most popular population based algorithms that tackle the timetabling problems are evolutionary algorithms (Ross et al. 1994; Burke et al. 1994; Colorni et al. 1990; Erben 2001), ant colony algorithm (Socha et al. 2002) and artificial immune system (Rozi et al. 2006) and their combination between local based and population based algorithms (Burke et al. 1995b; Duong and Lam 2004; Ross et al. 1998; Thanh 2007). Overviews of previous approaches for university timetabling problems are available in the following surveys (Carter and Laporte 1997, Lewis et al. 2008; Burke et al. 2007). Unfortunately, the quality of the solution produced by population based algorithms is inferior to local based algorithms mainly due to the fact that the population based algorithms normally experience *premature convergence,* the main reason for this problem in population based algorithms are that they are more concerned with exploration rather than exploitation. That is, the population based algorithms scan the solutions in the whole search space without rigorous concentration on those current solutions in addition to other drawbacks such as the need for more time (Marco et al. 2006). This is why several timetable researchers have lately focused attention on local based rather than population based algorithms (Abdullah et al. 2005a; Marco et al. 2006).

In light of the above, we suggest that the best way to design a timetabling algorithm that tackles university course timetabling is to provide a good balance among local based algorithms and population based algorithms to reach a suitable balance between exploration (global improvement) and exploitation (local improvement).

The harmony search algorithm is a new metaheuristic algorithm proposed by Geem et al. (2001). It is derived from the natural phenomena of musicians' behavior when they play their musical instruments together to come up with a fantastic harmony. It is considered a population based algorithm with local based aspects (Lee and Geem 2004). The aim of this paper is to apply harmony search for university course timetabling. Results show that harmony search can tackle this problem intelligently and find a near optimal solution.

This paper is organized as follows: section two discusses the university course timetabling problem based on hard and soft constraints. Section three discusses the harmony search algorithm and its adaptability to university course timetabling. Section four discusses the experimental results and compares it to those in the previous literatures. In the final section, we present a conclusion and future directions to our proposed algorithm.

2 The university course timetabling problem

The University Course Timetabling Problem(UCTP) version tackled in this paper was produced by Socha et al.(2002) using generator written by Paechter[1] and that can be described as follows: There are *n* courses(c_1, c_2,…c_n) each of which contains certain students and needs particular features, *k* rooms (r_1, r_2,…r_k) each of which has a seat capacity and contains specific features, *l* students (s_1, s_2,…s_l) each of them assigned one or more courses, *m* features (f_1, f_2,…f_m), and *p* timeslots(t_1, t_2,…t_p) where p=45 (5 days with 9 timeslots on each day). A binary matrix $X_{l,n}$ called *Student-Course* matrix where ($x_{i,j}$ =1) shows student *i* took course *j*, The *Room-Feature* matrix $Y_{k,m}$ is described as a room *i* that contains feature *j* if and only if ($y_{i,j}$ =1), and *Course-Feature* matrix $Z_{l,m}$ means that a course *i* needs feature *j* if and only if ($z_{i,j}$ =1).

The following hard constraints must be satisfied:

H1. *Student must not be double booked for courses.*
H2. *Room size and features must be suitable for assigned courses.*
H3. *Room must not be double booked for courses.*

And the following soft constraints should be minimized:

S1. *A student shall not have a class in the last slot of the day.*
S2. *A student shall not have more than two classes in a row.*
S3. *A student shall not have a single class on a day.*

The main objective for this context of UCTP is to produce a feasible solution as well as to minimize the violations of soft constraints. It is worth mentioning that the context of UCTP here reflects a real course timetabling problem at Napier University in Edinburgh, UK.

Originally, the context of UCTP used by Socha et al. (2002) was determined by Metaheuristics Network[2](MN) which is a European commercial research project shared by five Europe institutions between 2000 to 2004 to investigate the efficiency of different metaheuristics on different computational optimisation problems.

The same context of UCTP was used for the first International Timetabling Competition[3]. Twenty data instances and three more hidden ones were constructed using the same generator written by Paechter. Those data instances were proposed mainly in order to motivate the competitors to focus attention on generating brilliant approaches to UCTP. In fact, those data instances observed soft constraints minimization rather than hard constraints fulfillment. Some works that have lately appeared used the same data instances to measure the efficiency of their approaches (Marco et al. 2006; Kostuch 2005; Lewis and Paechter 2004; Burke et al. 2003).

The computational optimisation problems are difficult to solve due to the complexity and size of the problem and university community has increased rapidly in the last five decade. As such, Lewis and Paechter (2005) used the same Paechter's generator to construct 60 hard data instances to measure the capability

[1] Ben Paechter is a Professor in the School of Computing at Napier University, UK and a member of Metaheuristics Network. His official home page is "http://www.dcs.napier.ac.uk/~benp/". (27 June 2008).

[2] Metaheuristics Network official website "http://www.metaheuristics.net/". (27 June 2008).

[3] First International Timetabling Competition was organised by Metaheuristics Network members and was sponsored by PATAT. The Official website is "http://www.idsia.ch/Files/ttcomp2002/". (27 June 2008).

of grouping genetic algorithm to find feasible timetables. Tuga et al. (2007) used the same data instances to evaluate the performance of simulated annealing with kempe chain to find feasible timetables.

The post enrollment course timetabling problem (Lewis et al. 2007) was tracked on Second International Timetabling Competition[1]. This is similar to UCTP context of MN with slight differences: in the Second International Timetabling Competition two more hard constraints were addressed, the twenty one problem instances constructed for this track tackle different sizes, and the *distance to feasibility*[2] is considered to be another measurement for quality of solutions.

3 A Harmony search algorithm for UCTP

The harmony search (HS) is a new metaheuristic algorithm considered to be of the population based type which was proposed by Geem et al. (2001) and applied to several computational optimization problems such as structural design (Lee and Geem 2004), water network design (Geem 2006b), dam scheduling (Geem 2007b), school bus routing (Geem 2005), Sudoku game (Geem 2007a) and music composition (Geem 2006a). HS was derived from the natural phenomena of musicians' behavior when they collectively play their musical instruments (population members) to come up with a fantastic harmony (global optimal solution). This fantastic state is determined by an aesthetic standard (fitness function). Figure 2 shows the five steps in HS applied to UCTP.

UCTP is represented as a matrix $A_{k,p}$ where $a_{i,j}$ shows that the position of this matrix may either contain course c in i^{th} room and j^{th} timeslot or -1 if it is empty. See Figure 1.

$$\begin{array}{ll} \begin{pmatrix} a_{0,0} & a_{0,1} & \ldots\ldots. & a_{0,p} \\ a_{1,0} & a_{1,1} & \ldots\ldots. & a_{1,p} \\ a_{2,0} & a_{2,1} & \ldots\ldots. & a_{2,p} \\ \ldots. & \ldots. & \ldots\ldots. & \ldots. \\ a_{k,0} & a_{k,1} & \ldots\ldots. & a_{k,p} \end{pmatrix} & \downarrow_{\mathit{Timeslots}} \\ \rightarrow_{\mathit{Rooms}} & \end{array}$$

Figure 1 UCTP representation.

This representation shown in Figure 1 as is used by Lewis (2006) to satisfy hard constraint *H3* directly.

[1] The Second International Timetabling Competition was organised by five educational institutions and was sponsored by PATAT and WATT .The official website is "http://www.cs.qub.ac.uk/itc2007/index.htm". (27 June 2008).

[2] The term' distance to feasibility' indicates the number of courses that are not scheduled in the timetable in which the number of students within each unscheduled courses is a factor for evaluation.

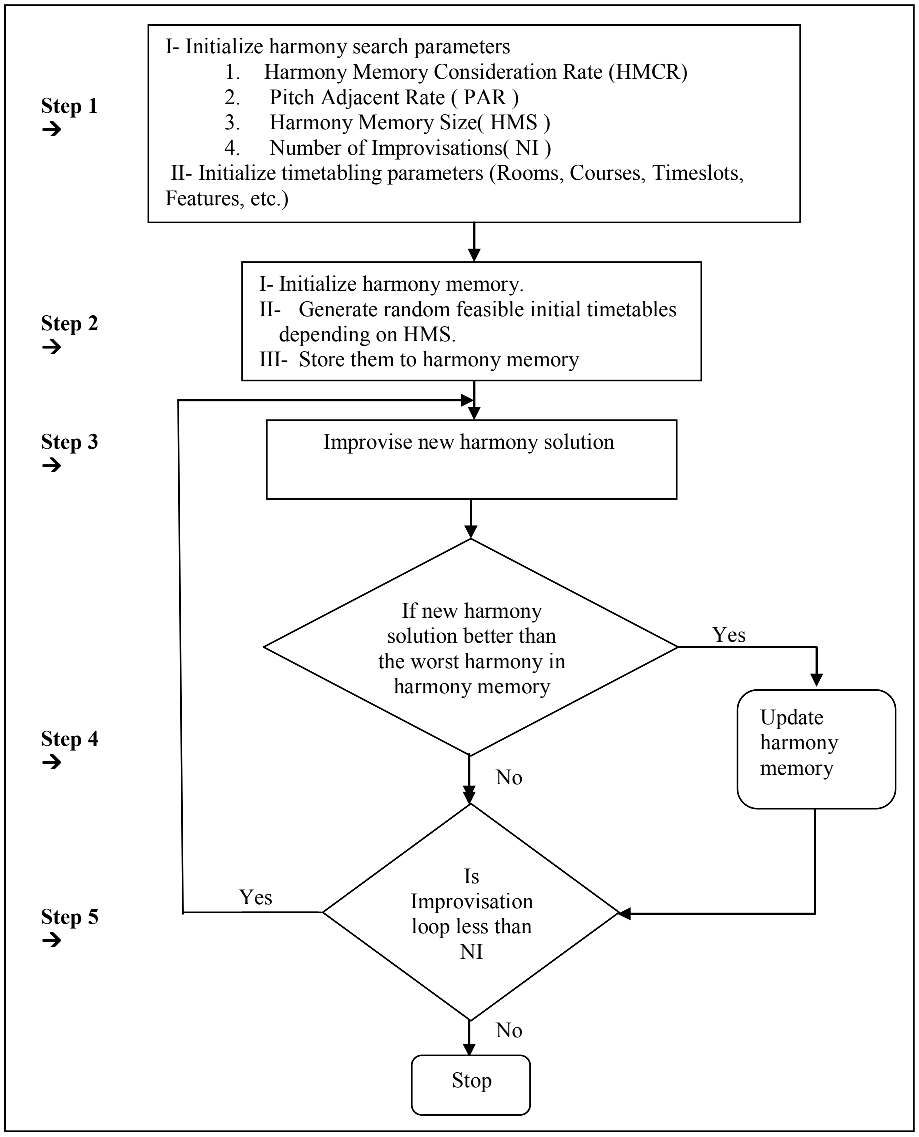
I- Initialize harmony search parameters
1. Harmony Memory Consideration Rate (HMCR)
2. Pitch Adjacent Rate (PAR)
3. Harmony Memory Size(HMS)
4. Number of Improvisations(NI)
II- Initialize timetabling parameters (Rooms, Courses, Timeslots, Features, etc.)
Step 1 ➔
I- Initialize harmony memory.
II- Generate random feasible initial timetables depending on HMS.
III- Store them to harmony memory
Step 2 ➔
Step 3 ➔
Improvise new harmony solution
If new harmony solution better than the worst harmony in harmony memory
Yes
Update harmony memory
Step 4 ➔
No
Is Improvisation loop less than NI
Yes
Step 5 ➔
No
Stop

Figure 2 a harmony search algorithm for UCTP.

The following data structures are used to build a university course timetable:

- *Conflict matrix*: is a matrix $B_{n,n}$ where $b_{i,j}$ contains either 0 if there is no one or more students sharing course *i* and course *j* or $b_{i,j} \geq 1$ if there is one student or more sharing course *i* and course *j*. This matrix is used to deal with the hard constraint *H1*.

- *Course room matrix*: is a binary matrix $D_{n,k}$ where $d_{i,j}$ contains either 1 if and only if course *i* and room *j* is compatible with both aspects of size and features or 0 otherwise. This matrix is used to deal with hard constraints *H2*.

- *Course position matrix*: is a matrix $Q_{n,HMS}$ where $q_{i,j}$ changes iteratively in a HS which contains either ($q_{i,j}$ = 1) if and only if a course *i* has a valid position for new harmony solution in the solution *j* that is stored in harmony memory (HM) or ($q_{i,j}$ =0) otherwise.

Step1: Initialize harmony search parameters and UCTP parameters

In step 1, fitness function (1) as used by Marco et al. (2006) is utilized to calculate the fitness value for each solution

$$f(A^i) = \sum_{s=1}^{l} f_1(a,s) + f_2(a,s) + f_3(a,s) \qquad (1)$$

Where $f(A^i)$ is a fitness function of i^{th} solution in HM, and $f_1(a,s)$, $f_2(a,s)$ and $f_3(a,s)$ describes the violation in soft constraint S1, S2 and S3 consecutively.

Following are the HS parameters:

1. HMCR: Harmony Memory Consideration Rate similar to crossover rate in genetic algorithm.
2. PAR: Pitch Adjusting Rate plays a significant role in determining the number of courses which will be moved to another position or swapping them with other courses like local search neighborhood structures.
3. HMS: Harmony Memory Size is similar to population size.
4. NI: Number of Improvisations is similar to number of iterations in optimisation algorithms.

Step2: Initialize HM with random feasible timetables based on HMS parameter

$$\begin{pmatrix} a_{0,0}^{1} & a_{0,1}^{1} & .... & a_{0,p}^{1} \\ a_{1,0}^{1} & a_{1,1}^{1} & .... & a_{1,p}^{1} \\ a_{2,0}^{1} & a_{2,1}^{1} & .... & a_{2,p}^{1} \\ .... & .... & .... & .... \\ a_{k,0}^{1} & a_{k,1}^{1} & .... & a_{k,p}^{1} \end{pmatrix} \begin{pmatrix} a_{0,0}^{2} & a_{0,1}^{2} & .... & a_{0,p}^{2} \\ a_{1,0}^{2} & a_{1,1}^{2} & .... & a_{1,p}^{2} \\ a_{2,0}^{2} & a_{2,1}^{2} & .... & a_{2,p}^{2} \\ .... & .... & .... & .... \\ a_{k,0}^{2} & a_{k,1}^{2} & .... & a_{k,p}^{2} \end{pmatrix} \cdots \begin{pmatrix} a_{0,0}^{HMS} & a_{0,1}^{HMS} & .... & a_{0,p}^{HMS} \\ a_{1,0}^{HMS} & a_{1,1}^{HMS} & .... & a_{1,p}^{HMS} \\ a_{2,0}^{HMS} & a_{2,1}^{HMS} & .... & a_{2,p}^{HMS} \\ .... & .... & .... & .... \\ a_{k,0}^{HMS} & a_{k,1}^{HMS} & .... & a_{k,p}^{HMS} \end{pmatrix}$$

Figure 3 harmony memory structures

In step 2, HS algorithm will generate feasible solutions with size of HMS, HM will be filled with those solutions; also the fitness value of all solutions in HM will be calculated. (see Figure 3).

Regarding UCTP, backtracking algorithm (Carter and Laporte 1996) and proposed *MultiSwap algorithm* is applied to generate random harmony solutions equal to HMS after assigning the courses by the *weighted largest degree first* heuristic method (Arani and Lofti 1989). This strategy ensures that all harmony solutions are feasible. First, all courses that can not be assigned to the timetable after largest weighted degree first heuristic method finished its assigning process will be entered to a list called *unscheduled list.*

This list will be passed to a backtracking algorithm that will select each unscheduled course *c* from the unscheduled list and explore all courses in conflict. Those courses that share one or more students with *c* will be removed from the timetable and added to the unscheduled list again. After that, the backtracking algorithm attempts to assign timeslots and rooms to all courses in the unscheduled list. This process will iterate several times until no further places can be filled. Some courses may not be scheduled at the end of this process. In this case, the *MultiSwap* algorithm will be used.

The proposed *MultiSwap algorithm* shuffles courses in different rooms in the same time slot where shuffling such courses is applied to all time slots consecutively. It is worth mentioning that there are two reasons why unscheduled courses cannot find places. Firstly, the proper room for the unscheduled course is reserved by other courses while the second relates to timeslots which contain courses sharing a student or more with the unscheduled course. Backtracking handles the second reason while MultiSwap tackles the first one. In MultiSwap algorithm, courses are taken from the same timeslot and are shuffled to a different suitable room in the hope of opening proper rooms to unscheduled courses.

If this process with predefined iterations can not find a feasible solution, we propose to restart the whole process from the beginning.

Step 3: Improvise new harmony solution

$$a_{i,j}^{NEW} = \begin{pmatrix} a_{0,0}^{NEW} & a_{0,1}^{NEW} & \ldots\ldots. & a_{0,p}^{NEW} \\ a_{1,0}^{NEW} & a_{1,1}^{NEW} & \ldots\ldots. & a_{1,p}^{NEW} \\ a_{2,0}^{NEW} & a_{2,1}^{NEW} & \ldots\ldots. & a_{2,p}^{NEW} \\ \ldots. & \ldots. & \ldots\ldots. & \ldots. \\ a_{k,0}^{NEW} & a_{k,1}^{NEW} & \ldots\ldots. & a_{k,p}^{NEW} \end{pmatrix}$$

Figure 4 new harmony solution

In step 3, a new feasible harmony solution (Figure 4) is generated based on three operators:

1 Memory considerations

Memory consideration operator selects the courses positions of new harmony solution based on solutions stored in HM with probability HMCR. In other words, the new value of $a_{i,j}^{NEW}$ is chosen randomly from the historical positions of timetables stored in HM such that $a_{i,j}^{NEW} \in \{a_{i,j}^{1}, a_{i,j}^{2}, a_{i,j}^{3}, \ldots, a_{i,j}^{HMS}\}$ with probability HMCR that varies between 0 and 1 as (2):

$$a_{i,j}^{NEW} = \begin{cases} a_{i,j}^{NEW} \in \{a_{i,j}^{1}, a_{i,j}^{2}, a_{i,j}^{3}, \ldots, a_{i,j}^{HMS}\} & w.p. \quad (HMCR) \\ \\ a_{i,j}^{NEW} \in A_{i,j} & w.p. \quad (1 - HMCR) \end{cases} \qquad (2)$$

HMCR has a probability of selecting historical positions of $a_{i,j}^{NEW}$ from feasible timetables stored in HM. For instance, if ($HMCR$=0.95), this indicates the number of courses will be selected randomly from a historical stored positions in the HM with a probability rate of 95%. To elaborate, if there are 400 courses scheduled in each timetable in HM with different positions, the HS algorithm will choose ($95\% \times 400$) courses in the new harmony solution depending on historical positions of timetables in HM.

As far as memory is concerned, it is safe to mention that the HS consecutively selects historical values stored in HM from $a_{1,1}^{NEW}$ to $a_{k,p}^{NEW}$ (Geem et al. 2001). Due to the fact that UCTP is NP-hard problem and that feasibility should be kept at this stage, it is difficult to consecutively find feasible positions for all obtained courses in the new harmony solution. By analogy to the ordering priority of *largest saturation degree* (Br´elaz 1979) where the courses must be ordered iteratively one by one based on assigning difficulties. The *smallest position algorithm has been proposed* to be responsible for choosing the courses which have the minimum historical available positions iteratively found in the *Course Position matrix*. If there is more than one course at each iteration with the same minimum available positions, the proposed algorithm will select one course depending on the *weighted largest degree first* (Arani and Lofti 1989).

2 Random considerations

(1-*HMCR*) courses that are not scheduled depending on memory consideration will be assigned to the new harmony solution randomly from the available course range. In fact, this process is very important to diversify the new harmony solution being similar to mutation operator in genetic algorithm (Geem 2005).

3 Pitch adjustments

After the new harmony solution is generated, the pitch adjusting operator will be applied to the new harmony solution with a probability Pitch Adjusting Rate (PAR). This operator will examine all courses that are scheduled out of harmony considerations. In other words, pitch adjusting selects the courses depending on memory consideration rather than random consideration. The obtained courses *move* to neighboring courses with a probability$(PAR \times HMCR)$ value where $0 \le PAR \le 1$. The other courses with a probability rate $((1-PAR) \times HMCR)$ are not changed. The HMCR parameter helps HS to focus on global improved solutions while PAR parameter helps HS to focus on local improved solutions (Geem 2005; Lee and Geem 2004).

Courses will be selected according to (3):

$$a_{i,j}^{NEW} = \begin{cases} Yes & w.p. \quad (HMCR \times PAR) \\ \\ No & w.p. \quad (HMCR \times (1-PAR)) \end{cases} \tag{3}$$

In case of UCTP, the selected course will move to a valid position or swap with another course with the probability of$(PAR \times HMCR)$. Here, pitch adjusting operator works similar to neighborhood structures of local based algorithms that is concerned with the exploitation of the new harmony solution while memory consideration operator is concerned with exploration.

Step 4: Update harmony memory

In step 4, the HS algorithm evaluates the new harmony solution. If the fitness value of the new harmony solution is better than the worst fitness value in HM, include new harmony solution in HM and exclude the worst solution from HM.

Step 5: Stop Criteria

In step 5, the HS algorithm will repeat step 3 and step 4 until the maximum number of iterations determined by NI parameter is met.

4 Experimental results and discussion

In this section, we present the performance of HS algorithm using the standard benchmarks for university course timetabling. The proposed method is coded in Microsoft Visual C++ 6 under windows XP on an Intel machine with 2 GHz processor and 256 RAM. We chose to test the proposed method using the data instances[1] prepared by Socha et al. (2002). Such data instances can measure the performance of approaches related to UCTP and were prepared carefully to mimic a real word UCTP at Napier University with different size and supersets of constraints. Socha et al. (2002) classified those instances into three classes: small, medium, and large and discussed the parameter value for each class. (see Table 1). We experimented five small data instances, five medium data instances and one large data instance; they were tested using 100000 iterations. The overall penalty of each dataset was assessed by the penalty function that adds up all violations of the soft constraints S1, S2 and S3. HMS= 50, HMCR =0.98 and PAR=0.02. Time consumed for each instance is approximately 20 minutes for each small instance and less than two hours for each medium instance. This time is considered reasonable for generating a timetable.

The results are compared with others (see Table 2): a tabu-search hyper-heuristic (Burke et al. 2003), a graph-based hyper-heuristic (Burke et al. 2007), max-min ant system, random restart local search (Socha et al. 2002), hybrid evolutionary approach (Abdullah et al. 2007), fuzzy multiple heuristic ordering (Asmuni et al. 2005), a variable neighborhood search (Abdullah et al. 2005b) and randomised iterative improvement (Abdullah et al. 2005a).

Table 1 parameter value for UCTP classes (Socha et al. 2002).

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

[1] Socha data instances were generated by a generator written by Ben Paechter available on this website: "http://iridia.ulb.ac.be/~msampels/tt.data/". (27 June 2008).

Table 2 comparison of results on the small / medium/large data instances

Datasets	*HS* (best)	*RII* (best)	*RRLS* (Avg)	*MMAS* (Avg)	*VNS* (best)	*GHH* (best)	*FMHO* (best)	*HEA* (best)	*THH* (best)
small1	5	**0**	8	1	1	6	10	**0**	1
Small2	3	**0**	11	3	2	7	9	**0**	2
Small3	2	**0**	8	1	**0**	3	7	**0**	**0**
Small4	3	**0**	7	1	1	3	17	**0**	1
Small5	1	**0**	5	**0**	**0**	4	7	**0**	**0**
Meduim1	316	242	199	195	317	372	243	221	**146**
Meduim2	243	161	202.5	184	313	419	225	**147**	173
Meduim3	255	265	77.5% inf.	248	357	359	249	**246**	267
Meduim4	235	181	177.5	**164.5**	247	348	285	165	169
Meduim5	215	151	100% inf.	219.5	292	171	132	**130**	303
Large	100% inf	100% inf.	100% inf.	851.5	100% inf.	1068	1138	**529**	80% inf. 1166

In Table 2, the following abbreviations have been used to mean:

HS– our Harmony Search algorithm.
RII– Randomised Iterative Improvement (Abdullah et al. 2005a).
RRLS– Random Restart Local search (Socha et al. 2002).
MMAS– MAX-MIN Ant System (Socha et al. 2002).
VNS– Variable Neighborhood Search (Abdullah et al. 2005b).
GHH– Graph-based Hyper-Heuristic (Burke et al. 2007).
FMHO– Fuzzy Multiple Heuristic Ordering (Asmuni et al. 2005).
HEA– Hybrid Evolutionary Approach (Abdullah et al. 2007).
THH– Tabu-search Hyper-Heuristic (Burke et al. 2003).
x%inf – as used by Abdullah et al. (2007) indicate that the percentage of such algorithm could not find a feasible timetable.

As shown in Table 2, the result seems competitive with those in other previous works. The HS algorithm is capable of refining the quality of the course timetable and producing a near optimal solution. The result also seems to fall within the range of previous works that used the same data instances.

It can be observed that the experimental results of HS algorithm show that HS is able to find feasible solutions for small and medium data instances. The proposed approach obtained better results than GHH and FMHO for almost all small data instances. Moreover, it obtained better results than VNS in medium data instances and some others in FMHO.

5 Conclusion and future work

This paper has presented HS algorithm for tackling UCTP. As it has been shown, the proposed algorithm can find near optimal solutions for UCTP and produces better results than several others in the previous literature.

HS stands out for being able to strike a balance between exploration through HMCR in memory consideration and exploitation through PAR in the pitch adjusting procedure.

For future research, we aim to improve HS for UCTP by introducing different neighborhood structures to pitch adjusting procedure and by trying to integrate HS with other metaheuristic algorithms. The tuning of HS parameters for UCTP is to be addressed as well.

6 References

Abdullah S, Burke EK, McCollum B (2005a) Using a randomised iterative improvement algorithm with composite neighbourhood structures for course timetabling. In Proceedings of MIC 05: The 6th Meta-Heuristic International Conference, Vienna, Austria, 22-26 Aug 2005.
Abdullah S, Burke EK, Mccollum B (2005b) An investigation of variable neighbourhood search for university course timetabling. Proceedings of MISTA 2005: The 2nd Multidisciplinary Conference on Scheduling: Theory and Applications. NY, USA.
Abdullah S, Burke EK, Mccollum B (2007) A hybrid evolutionary approach to the university course timetabling problem. In Proceedings of the IEEE Congress on Evolutionary Computation. Singapore, September 2007.
Abramson D (1991) Constructing school timetables using simulated annealing: sequential and parallel algorithms. Management Science 37(1):98–113.
Arani T, Lofti JA (1989) A three phased approach to final exam scheduling. IIE Transactions 21(4):86-96.
Asmuni H, Burke EK, GARIBALDI J (2005) Fuzzy multiple heuristic ordering for course timetabling. IN AL,S. M. A. E. (Ed.) Proceedings of the 5th United Kingdom Workshop on Computational Intelligence. London, UK, (UKCI05).
Br´elaz D (1979) New methods to color vertices of a graph. Communications of the ACM 22(4):251–256.
Burke EK, Elliman DG, Weare RF (1995) A hybrid genetic algorithm for highly constrained timetabling problems. 6th International Conference on Genetic Algorithms (ICGA'95, Pittsburgh, USA, 15th-19th July 1995). Morgan Kaufmann, San Francisco, CA, USA.
Burke EK, Bykov Y, Newall JP et al (2003) A time-predefined approach to course timetabling. Yugoslav Journal of Operations Research 13: 139-151.
Burke EK, Elliman DG, Weare RF (1994) A genetic algorithm for university timetabling. In proceedings of the AISB workshop on Evolutionary Computing. University of Leeds, UK.
Burke EK, Kingston J, Jackson K et al (1997) Automated university timetabling: the state of the art, The Computer Journal 40 (9): 565-571.
Burke EK, Kendall G, Soubeiga E (2003) A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9(6)**:** 451-470.
Burke EK, Mccollum B, Meisels A et al (2007) A Graph-Based Hyper-Heuristic for Educational Timetabling Problems. European Journal of Operational Research 176(1): 177-192.
Carter MW, Laporte G, Lee SY (1996) Examination timetabling: algorithmic strategies and applications. Journal of the Operational Research Society 74(3):373-383.
Carter MW, Laporte G (1997) Recent developments in practical course timetabling. In: E. Burke, M. Carter (Eds.), the Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference, Springer Lecture Notes in Computer Science 1408: 3-19.
Chiarandini M, Birattari M, Socha K et al (2006) An effective hybrid algorithm for university course timetabling. Journal of Scheduling 9(5):403-432.
Colorni A, Dorigo M, Maniezzo V (1990) Genetic algorithms and highly constrained problems: the timetable case. Proceedings of the First International Workshop on Parallel Problem Solving from Nature. Dortmund, Germany, Springer-Verlag ,55-59.
Duong T, Lam K (2004) Combining constraint programming and simulated annealing on university exam timetabling. In: Proceedings of RIVF 2004 Conference Hanoi,Vietnam.
Erben W (2001) A grouping genetic algorithm for graph colouring and exam timetabling. Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16-18, 2000, Selected Papers.
Fang H (1994) Genetic algorithms in timetabling and scheduling. Ph.D. thesis, Department of Artificial Intelligence. University of Edinburgh, Edinburgh, UK.
Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman and Company.New York.
Geem ZW (2005) School bus routing using harmony search. In Rothlauf, F. (Ed.) Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO'2005). Washington, D.C., USA.

Geem ZW (2006a) Improved harmony search from ensemble of music players. Lecture Notes in Artificial Intelligence 4251:86–93.
Geem ZW (2006b) Optimal cost design of water distribution networks using harmony search. Engineering Optimization 38(3):259-280.
Geem ZW (2007a) Harmony search algorithm for solving sudoku. Lecture Notes in Artificial Intelligence 4692:371–378.
Geem ZW (2007b) Optimal scheduling of multiple dam system using harmony search algorithm. Lecture Notes in Computer Science 4507:316–323.
Geem ZW, Kim J, Loganathan G (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60-68.
Kostuch P (2005) The university course timetabling problem with a three-phase approach. In Edmund Burke and Michael Trick, editors, Proc. of the 5th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2004 Berlin-Heidelberg), selected papers, Springer Lecture Notes in Computer Science 3616:109–125.
Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Computers and Structures 82(9):781-798.
Lewis R (2008) A survey of metaheuristic-based techniques for University Timetabling problems. OR Spectrum 30: 167-190.
Lewis R (2006) Metaheuristics for university course timetabling. PhD thesis, School of Computing, Napier University, Edinburgh, October.
Lewis R, Paechter B, Mccollum B (2007) Post enrolment based course timetabling: a description of the problem model used for track two of the second international timetabling competition. Cardiff University, Cardiff Business School, Accounting and Finance Section.
Lewis R, Paechter B (2005) Application of the grouping genetic algorithm to university course timetabling. in Evolutionary Computation in Combinatorial Optimisation (EVOCop) V. G. Raidl and J. Gottlieb Eds.Berlin, Germany: Springer-Verlag, 2005. LectureNotes in Computer Science 3448: 144-153.
Lewis R, Paechter B (2004) New crossover operators for timetabling with evolutionary algorithms. In:Lofti A (ed) The fifth international conference on recent advances in soft computing RASC2004. Nottingham, England.
Marco C, Mauro B, Krzysztof S et al (2006) An effective hybrid algorithm for university course timetabling. Journal of Scheduling. 9(5): 403-432.Doi: 10.1007/s10951-006-8495-8.
Malim MR, Khader AT, Mustafa A (2006) Artificial immune algorithms for university timetabling. Proceedings of the 6th International Conference on the Practice & Theory of Automated Timetabling (PATAT 2006). 30 Ogos – 1 September 2006, Czech Republic.
Ross P, Hart E, Corne D (1998) Some observations about GA-based exam timetabling. In: E.K. Burke and M. Carter (eds.), LNCS 1408, Practice and Theory of Automated Timetabling II: Second International Conference, PATAT 1997, Toronto, Canada, selected papers. Springer-Verlag,115–129.
Socha K, Knowles J and Samples M (2002) A max-min ant system for the university course timetabling problem. Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002), Springer Lecture Notes in Computer Science 2463:1-13.
Thanh ND (2007) Solving timetabling problem using genetic and heuristic algorithms. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference.
Thompson GL, Dowsland KA(1996) Variants of Simulated Annealing for the Examination Timetabling Problem. Annals of Operations Research 63(1):105-128.
Tuga M, Berretta R, Mendes A (2007) A Hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem. Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference.