Hybrid Harmony Search with Great Deluge for UUM CAS Curriculum Based Course Timetabling

Juliana Wahid[1], Naimah Mohd Hussin[2]
[1]*School of Computing, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah.*
[2]*Faculty of Computer Science and Mathematical, Universiti Teknologi MARA (Perlis), 02600 Arau, Perlis.*
w.juliana@uum.edu.my

***Abstract*—Producing university course timetabling is a tough and complicated task due to higher number of courses and constraints. The process usually consisted of satisfying a set of hard constraints so as a feasible solution can be obtained. It then continues with the process of optimizing (minimizing) the soft constraints in order to produce a good quality timetable. In this paper, a hybridization of harmony search with a great deluge is proposed to optimize the soft constraints. Harmony search comprised of two main operators such as memory consideration and random consideration operator. The great deluge was applied on the random consideration operator. The proposed approach was also adapted on curriculum-based course timetabling problems of College of Arts and Sciences, Universiti Utara Malaysia (UUM CAS). The result shows that the quality of timetable of UUM CAS produced by the proposed approach is superior than the quality of timetable produced using the current software package.**

***Index Terms*—Harmony Search; Great Deluge; Curriculum Based Course Timetabling.**

I. INTRODUCTION

University curriculum based course timetabling problems (CBCTT) have been long classified as non-deterministic polynomial (NP) hard combinatorial optimization problems because of the exponential growth of this problem [1]. In other word, the entities involved in this problem such as courses, teachers and students are increase rapidly in numbers hence the allocation of these entities to a number of fixed timeslots and rooms becomes more complicated.

In addition, several constraints must be satisfied while assigning those entities. The aim is to generate a timetable that is feasible in which each lecture of a course must be scheduled in a distinct timeslot and room and any two lectures cannot be assigned to the same timeslot. These conditions are categorized as hard constraints in timetabling. Hard constraints are matters that are rigidly fulfilled in the timetable construction. In the process of assigning the courses, some other conditions, which are soft constraints, would also be considered, for instance the number of students attending the course for each lecture must be less than or equal to the capacity of the rooms hosting the lectures, the lectures of each course should be spread across a given number of days, etc. The soft constraints can be violated, but for a good quality timetable, the soft constraints violation should be minimized. At the end of the process, the overall objective is to satisfy all the hard constraints and to minimize the violation soft constraints.

The most common approach for solving the CBCTT is metaheuristic methods. According to the website of the Meta-heuristics Network (http://www.metaheuristics.org), which is a European Union (EU) sponsored research project that was run from 2000 until 2004, a meta-heuristic is a general algorithmic skeleton that can be employed to diverse optimization problems with some amendment so as they can be fitted to a particular problem.

Various metaheuristic search methods have been used in dealing with CBCTT. The metaheuristic approaches include tabu search [2,3] great deluge [4], simulated annealing [5,6], and ant bee colony [7,8].

The objective of this paper is two-fold: Presenting a hybrid metaheuristic approaches, i.e. harmony search with great deluge for solving the CBCTT problem and illustrating the proposed approach on CBCTT problem of College of Arts and Sciences, Universiti Utara Malaysia (UUM CAS). The second objective gives significant contribution in this paper, because of the use of real world data with the proposed approach, compared to other related works with the same domain that uses only the benchmark data sets.

The remainder of this paper is organized as follows: Section 2 presents the description of UUM CAS course timetabling problem such as the basic entities involved, constraints applied and the quality cost of the UUM CAS course timetabling. In Section 3, the proposed approach is described. Section 4 demonstrates the results on the experiment carried out using the proposed method on the UUM CAS data set. Finally, Section 5 presents some conclusions and suggestions for future work.

II. COLLEGE OF ART AND SCIENCES UNIVERSITI UTARA MALAYSIA COURSE TIMETABLING

College of Arts and Sciences Universiti Utara Malaysia (UUM CAS) consists of five academic schools, i.e. the School of Multimedia Technology and Communication, School of Education and Modern Languages, School of Computing, School of Quantitative Sciences and the School of Social Development, and offers fifteen programs. There is also a Centre for General Studies and a Language Centre to support the UUM CAS academic development activities.

This paper focuses on UUM CAS course timetabling of programs at the undergraduate level. There are two semesters per academic year. Based on the timetable obtained from Academic Affair Department of UUM for session 2013/2014 semester 1, there were 247 courses, 850 lectures, 32 rooms, 350 lecturers, and 20,000 students to be scheduled on a five-day week (Sunday to Thursday). This data set is now referred to as UUMCAS A131. There are several standard meeting lecture patterns implemented according to course requirements. Table 1 shows the pattern and the percentage of the courses involved with the pattern.

Table 1
Percentage of Courses Involved in Meeting Pattern

Meeting Pattern	Percentage courses involved
1.5 hours x 2 days per week	85 %
1.5 hours x 2 days + 1 hour x 1 day per week	5 %
2 hours x 2 days per week	8 %
3 hours x 1 day + 2 hours x 1 day per week	0.4 %
3 hours x 1 day per week	1.6 %

Most classes have all meetings taught in the same room, by the same lecturer, at the same time of the day, except if the course involves combination of class with laboratory or class with tutorial with different rooms and/or time. Each day is made up of 9 hours starting from 8.30 a.m. to 5.30 p.m. with no classes after 2.30 p.m. on Tuesday to allow for meetings and other extra-curricular activities.

The current implementation of the Academic Affairs Department in scheduling the lectures is by using a ready-made software package which provides a set of tools that the timetable officer can use to simplify the work. The timetable is essentially created manually, using a set of tools that can help to detect clashes and suggest suitable timeslots. This is a long process and a semester timetable takes an average of three weeks to be prepared given that all necessary data have been entered into the system. The necessary data includes student registration details, lecturer course assignments from all academic schools, course requirements, and updated room capacities.

Based on the set of standardized meeting patterns shows in Table 1, it is quite complex to implement different blocks of timeslot such as 1 hour, 1.5 hours, 2 hours and 3 hours. Rather, it seems possible to apply the same block of timeslot, i.e. a half hour, which can be composed to fulfill 1 hour, 1.5 hours, 2 hours and 3 hours blocks of timeslot. As each day consists of 9 hours, the total of half hour in a day is 18 which is equal to the number of timeslots. The total number of periods is 84 not 90 because on Tuesday the lectures end at 2.30 p.m. The number of lectures, which is 850 as stated earlier, consists of lectures from timeslot of 1 hour, 1.5 hours, 2 hours and 3 hours. Therefore, when considering the timeslot of half hour, the total number of lectures increases to 2300. This is the highest number of lectures exist in a dataset, in which most of the benchmark datasets consist of only several hundreds of lectures. Table 2 shows the basic entities of UUMCAS A131 course timetabling data.

In brief, Table 3 shows the main features of the UUMCAS A131 data set such as the number of courses (C), total number of lectures (L), number of rooms (R), total periods per day (PpD), number of days (D), number of curricula (Cu), and number of unavailability constraints (UC).

The hard and soft constraints considered in UUM CAS course timetabling problem are same with the constraints used by Cesco et al. [2] as follows:

i) Hard constraints:
H1 - Lectures: All lectures of a course must be scheduled and assigned to distinct periods.
H2 - Conflicts: Lectures in the same curriculum or taught by the same teacher must all be scheduled in different periods.
H3 - Room occupancy: Two lectures cannot be located in the same room at the same time.
H4 - Availability: If the teacher of the course is not available to teach that course at a given period, then no lecture of the course can be scheduled at that time.

ii) Soft constraints:
S1 - Room Capacity: For each lecture, the number of students that attend the course must be less or equal than the number of seats in all rooms that host the lectures.
S2 - Min Working Days: The lectures of each course must divided into the given minimum number of days.
S3 – Isolated Lectures: Lectures that belong to a curriculum should be adjacent to each other (i.e., in consecutive periods).
S4 - Room Stability: All lectures of a course should be given the same room.

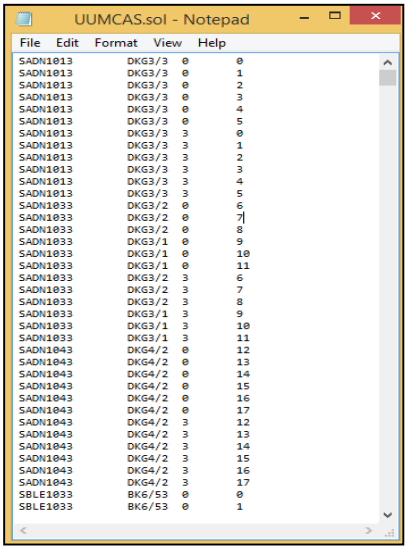

Figure 1: UUMCAS A131 Course Timetable

Table 2
Basic Entities of UUM CAS Course Timetabling Data

Entity	Definition
Days (d)	Number of teaching days in the week, d = 5
Timeslots (ts)	Each day is split into a fixed number of timeslots, which is equal for all days, ts = 18
Periods (p) = d X ts	A pair composed of a day and a timeslot. The total number of scheduling periods is the product of the days times the day timeslots. A set of *p*periods, T={$T_1, \ldots T_P$}. p = 5 X 18 = 90 – 6 (Tuesday only 12 timeslot) = 84
Courses and Teachers	A set of n courses, C= {$C_1,\ldots,C_n$}, each course is composed of the number of lectures (L), to be scheduled and each lecture is associated to a *teacher*, n = 247, L = 2298
Rooms	Each room has a capacity, expressed in terms of number of available seats (c), and a location expressed as an integer value representing a separate building (l). Some rooms may not be suitable for some courses (because they miss some equipment). A set of m rooms, R={$R_1, \ldots R_m$}. m = 32
Curricula	A curriculum is a group of courses such that any pair of courses in the group have students in common. Based on curricula, we have the conflicts between courses and other soft constraints. Set of q curricula Cu = {Cu_1, Cu_2, ..., Cuq}, q = 178

Table 3
Main Features of UUM CAS Data Set

Main Features	Total
C	247
L	2300
R	44
PpD	18
Cu	178
UC	1482

The quality of solution is calculated as the total penalties of the constraints: H1 + H2 + H3 + H4 + S1 + S2 + S3 + S4.

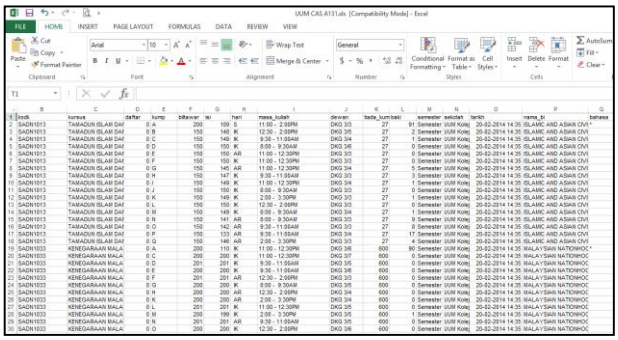

Figure 2: UUMCAS A131 Solution Text File

The UUMCAS A131 course timetable obtained from the Academic Affairs Department which was produced by the ready-made software package is validated using validator algorithm to calculate the quality of solution (hard constraints and soft constraints). The actual timetable is in a form of spreadsheet as shown in Figure 1.

In order to be validated by the validator algorithm, the course timetable needs to be converted into a text file and the file name has extension "sol" (shortcut from the word solution). The solution file contains lines that represent the assignment of the room and the timeslot to one lecture (lines can be in any order) according to the following format: <CourseID><RoomID><Day><Day_Period> as shown in Figure 2. For example, the first line states that a lecture of SADN1013 takes place at room DKG3/3 on Sunday (0) in the first period (0).

The validator algorithm file can be downloaded from the CBCTT community web site: http://tabu.diegm.uniud.it/ctt as C++ source code. The validator algorithm produces standard output of the evaluation of the solution along with the detailed description of all violations (hard and soft) as shown in Figure 3. The total cost of UUMCAS A131 course timetable is 1230, which consists of 1178 hard constraints and 52 soft constraints.

```
[H] Courses STIJ2024 and STIV2013 have both a lecture at period 22 (day 1, timeslot 4)
[H] Courses STIJ2024 and STIV2013 have both a lecture at period 23 (day 1, timeslot 5)
[H] Courses STIJ2024 and STIV2013 have both a lecture at period 76 (day 4, timeslot 4)
[H] Courses STIJ2024 and STIV2013 have both a lecture at period 77 (day 4, timeslot 5)
[H] Courses STIJ3044 and STIV2013 have both a lecture at period 30 (day 1, timeslot 12)
[H] Courses STIJ3044 and STIV2013 have both a lecture at period 31 (day 1, timeslot 13)
[H] Courses STIJ3044 and STIV2013 have both a lecture at period 32 (day 1, timeslot 14)
[H] Courses STIJ3044 and STIV2013 have both a lecture at period 33 (day 1, timeslot 15)
[H] Courses STIJ3044 and STIV2013 have both a lecture at period 84 (day 4, timeslot 12)
[H] Courses STIJ3044 and STIV2013 have both a lecture at period 85 (day 4, timeslot 13)
[H] Courses STIJ3044 and STIV2013 have both a lecture at period 86 (day 4, timeslot 14)
[H] Courses STIJ3044 and STIV2013 have both a lecture at period 87 (day 4, timeslot 15)
[H] Courses STIK1013 and STIN1013 have both a lecture at period 12 (day 0, timeslot 12)
[H] Courses STIK1013 and STIN1013 have both a lecture at period 13 (day 0, timeslot 13)
[H] Courses STIK1013 and STIN1013 have both a lecture at period 14 (day 0, timeslot 14)
[H] Courses STIK1013 and STIN1013 have both a lecture at period 66 (day 3, timeslot 12)
[H] Courses STIK1013 and STIN1013 have both a lecture at period 67 (day 3, timeslot 13)
....
....
[H] Courses STQM1203 and STQS1023 have both a lecture at period 61 (day 3, timeslot 7)
[H] Courses STQM1203 and STQS1023 have both a lecture at period 62 (day 3, timeslot 8)
[H] Courses STQM1203 and STQS1023 have both a lecture at period 76 (day 4, timeslot 4)
[H] Courses STQM1203 and STQS1023 have both a lecture at period 77 (day 4, timeslot 5)
[S(1)] Course SADN1033 uses 2 different rooms
[S(1)] Course SBLE1033 uses 2 different rooms
....
....
[S(1)] Course STIW3103 uses 2 different rooms
[S(3)] Course STQM1203 uses 4 different rooms

Violations of Lectures (hard) : 0
Violations of Conflicts (hard) : 1178
Violations of Availability (hard) : 0
Violations of RoomOccupation (hard) : 0
Cost of RoomCapacity (soft) : 0
Cost of MinWorkingDays (soft) : 0
Cost of IsolatedLectures (soft) : 0
Cost of RoomStability (soft) : 52

There are 0 warnings!
Summary: Violations = 1178, Total Cost = 52
Press any key to continue . . .
```

Figure 3: UUMCAS A131 Validator Output

From the validator output, it is found there are 1178 lectures that are in conflict. From the observation, even with conflicts existing in the timetable, the timetable of UUMCAS A131 can actually be implemented as the student chose groups that do not clash with their other courses. For example, 12 half-hours of lectures consist of two groups (each with 6 half-hours or 3-hours timeslot). Therefore, students have the option to choose groups that do not clash with their other courses.

III. Hybridization of Harmony Search with Great Deluge

The harmony search algorithm (HSA) is a metaheuristic algorithm that imitates the improvisation of musical process in searching for a perfect state of harmony according to audio-aesthetic standard [9]. HSA is categorized as population based metaheuristic in which the optimization involves population of solutions.

The HSA requires six step such as:

i) Setting the algorithm parameter such as Harmony Memory Consideration Rate (HMCR), Harmony Memory Size (HMS) (that is, equivalent to population size), Pitch Adjustment Rate (PAR), and Maximum Improvisations (MI) (that is, the maximum number of generations)

ii) Harmony Memory (HM) initialization – process of constructing the population of initial solutions. The number of initial solutions is determined by the value of HMS stated in the first step.

iii) Harmony improvisation – the solution is optimized (improved) using the following operators:
   a. Memory consideration (MC) – choosing the lecture (from the HM) to be assigned in the timetable slot.
   b. Random consideration (RC) – choosing the lecture from all lectures that are available.
   c. Pitch adjustment (PA) – replacing the lecture assigned by memory consideration operator.

iv) Update memory with the solution found – the new solution that is better from the previous solution is included into HM

v) Determine the termination criteria – the termination criteria used is the number of the iteration process, i.e. Maximum Improvisations (MI) that are defined in the first step

vi) Cadenza (musical terminology) – return the best harmony.

For step 1, the HSA parameters are set as HMS = 10, HMCR = 0.8, PAR = 1.0, MI = 1000

For step 3, there are six neighborhood structures used, five in PA operator, while another one is in the RC operator. The neighborhood structures in PA operator are selected using a random number between 0 and 1 which is multiplied by PAR value listed as follows:

- The *Move timeslot.* With probability between 0% × PAR and 20% × PAR, the lecture is randomly *moved* to any feasible timeslot in the same room.
- The *Swap timeslot.* With probability between 20% × PAR and 40% × PAR, the lecture is swapped with the timeslot of another lecture, while the rooms of both lectures are not changed.
- The *Move room.* With probability between 40% × PAR and 60% × PAR, the lecture is randomly *moved*

to any feasible timeslot that is located in the same period with a different room.

- The *Swap room.* With probability between 60% × PAR and 80% × PAR, the lecture is swapped with the timeslot of another lecture located in the same period with a different room.
- The *Kempe* chain move. With probability between 80% × PAR and 100% × PAR, the lecture is moved using *Kempe* chain move.

The RC operator, which is selected based on 1 - HMCR probability, will move or swap the lecture randomly to another timeslot (whether it is in the same room and timeslot or different room and timeslot) that is available and feasible.

The GD algorithm, which is a local search based metaheuristic, begins with initial solution quality (water level, *B*, which is usually set according to the quality of the initial solution) and decreased by specific rate (decay rate) at each iteration. The original decay rate proposed by Dueck [10] is:

$$B = B - \Delta B \text{ in which } \Delta B = \frac{s-\beta}{MaxIter} \quad (1)$$

where s is the current solution, β is minimum expected penalty corresponding to the best solution, and *MaxIter* is the number of iterations in the algorithm.

During the process, at each iteration, an improving solution is always accepted, i.e. evaluation function of neighborhood moves, S^{new} is better than evaluation function of current solution, s ($S^{new} \leq s$) while a worsening one is accepted if it is better than the initial water level, B ($S^{new} \leq B$).

In a hybridization of HSA with GD as shown as pseudocode in Figure 4, GD were hybridized with HSA in the RC operator. The objective of hybridizing a local search based within a population based method is to attain a balance between exploration and exploitation of the search space utilizing the advantage of population-based and local search based methods [11,12].

As shown in Figure 4 in Step 3, for each movement of selected neighborhood structure on the PA operator, the quality of the new solution $f(x^{NEW})$ is calculated and compared to the quality of the best solution, $f(x^{CURRBEST})$. If there is an improvement, where $f(x^{NEW})$ is less or equal to $f(x^{CURRBEST})$, the new solution x^{NEW} is accepted and $x^{CURRBEST}$ is set to the new solution x^{NEW}. The worse solution is not accepted.

In this proposed hybridization of HSA with GD algorithms, the water level B is not using any decay rate; instead, the value of B is set to the value of the updated best solution in the HM at every MI iteration. In other words, the same water level B is used within an N variables iteration.

With a probability of 1-HMCR, the RC operator randomly *moved* the lecture to any feasible timeslot or *swapped* the lecture with another lecture located in the same or different rooms or periods. The execution of the RC operator is calculated and compared using GD acceptance formula. The improved solution ($f(x^{NEW})$ is less or equal with $f(x^{CURRBEST})$) is always accepted while the worse solution is accepted if it is less or equal to the value of current water level B.

At the end of N variables iteration, in Step 4, the new solution x^{NEW} will be updated to the HM if the cost of new solution $f(x^{NEW})$ is less or equal to the worst solution in the HM, $f(x^{WORST})$. If the new solution x^{NEW} is worse than the worst solution in the HM, the new solution is accepted if it is less or equal to the value of current water level B. At the beginning of next MI iteration, the water level B is set to the best solution found so far. Step 3 and 4 in Figure 4 are repeated until the termination criteria (number of MI) are met and the best solution found is returned at the end of this algorithm.

Step 1: HSA parameters settings (HMS,HMCR, PAR, MI)
Step 2: Initialize HM{x_1,..., x_{HMS}}
  while not termination criterion specified by MI **do**
Step 3: Harmony Improvisation
  Select the best harmony $x^{BEST} \in (x_1,..., x_{HMS})$.
  Set Current Best harmony, $x^{CURBEST} = x^{BEST}$
  Set water level, B= $f(x^{BEST})$
  for j =1,..., N **do** (N is the number of decision variables)
  if $U(0,1) \leq$ HMCR ***(memory consideration)***
  (pitch adjustment)
  Move timeslot: $0 \leq U(0,1) \leq 0.2$xPAR
  Swap timeslot: 0.2xPAR < $U(0,1) \leq 0.4$xPAR
  Move room: 0.4xPAR < $U(0,1) \leq 0.6$xPAR
  Swap room: 0.6xPAR < $U(0,1) \leq 0.8$ xPAR
  Kempe chain move: 0.8xPAR < $U(0,1) \leq 1$ xPAR
  if $f(x^{NEW}) <= f(x^{CURBEST})$
  $x^{CURBEST} = x^{NEW}$
  end if (*end of memory consideration*)
  else (*random consideration)*
  Move or Swap (timeslot and room)
  if $f(x^{NEW}) <= B$ **or** $f(x^{NEW}) <= f(x^{CURBEST})$
  $x^{CURBEST} = x^{NEW}$
  end if (*end of random consideration*)
  end for
Step 4: Update the new harmony in the HM
  if $f(x^{NEW}) <= B$ **or** $f(x^{NEW}) <= f(x^{WORST})$
  $x^{WORST} = x^{NEW}$
  end if
  end while (**Step 5:** Performing termination)
Step 6: Cadenza (returns the best harmony ever found)

Figure 4: Hybridization of HSA with GD Pseudocode

IV. Experimental Result

The above UUMCAS A131 data set is now processed using the proposed algorithm. The UUMCAS A131 data set will be processed using the construction algorithm approach(which fulfill step 2 of HSA) from Juliana. The number of population of feasible initial solution is set to 10. Feasible initial solution means that the timetable does not violate any hard constraints, but may violate the soft constraints. The total number of feasible solutions (with no hard constraints) is easily obtained over 10 iterations. The penalty of soft constraints for the 10 feasible solutions are shown in Table 4.

Table 4
Soft Constraint Cost of 10 Feasible Solution of UUMCAS A131 Data Set

Iteration	S1	S2	S3	S4	Total Cost
1	19574	0	3106	1654	24334
2	19697	0	3034	1657	24388
3	19939	0	2934	1672	24545
4	20135	0	3116	1665	24916
5	20399	0	2996	1660	25055
6	20295	0	3134	1662	25091
7	20832	0	2914	1650	25396
8	20792	5	3014	1663	25474
9	20621	0	3286	1654	25561
10	21071	0	3022	1664	25757

In the improvement phase, the above population of 10 feasible initial solutions of UUMCAS A131 data set is improved using the hybridization of HSA with GDwith different harmony memory consideration rates (HMCR) (0.2, 0.5 and 0.8) with 1000 iterations and executed with 10 runs for each HMCR. Tables 5 to 7 show the result of UUMCAS A131 data set after being improved for 10 runs for each HMCR respectively. In brief, the results in Table 5 to 7 are highlighted in terms of best, average and worst result as shown in Table 8.

Table 5
Result of Improvement with HMCR 0.2

Run	S1	S2	S3	S4	Total Cost
1	1	0	180	1105	1286
2	0	0	226	1146	1372
3	2	0	78	945	1025
4	0	0	38	932	970
5	0	0	16	869	885
6	0	0	20	860	880
7	0	0	194	1144	1338
8	0	0	225	1147	1372
9	2	0	77	946	1025
10	1	0	179	1106	1286

Table 6
Result of Improvement with HMCR 0.5

Run	S1	S2	S3	S4	Total Cost
1	0	0	130	981	1111
2	0	0	104	995	1099
3	1	0	110	963	1074
4	0	0	112	980	1092
5	0	0	102	933	1035
6	0	0	74	961	1035
7	0	0	122	1010	1132
8	0	0	86	995	1081
9	0	0	124	994	1118
10	0	0	128	979	1107

Table 7
Result of Improvement with HMCR 0.8

Run	S1	S2	S3	S4	Total Cost
1	0	0	26	716	742
2	0	0	28	706	734
3	0	0	46	708	754
4	0	0	30	739	769
5	0	0	34	674	708
6	0	0	34	719	753
7	0	0	50	698	748
8	0	0	32	681	713
9	0	0	34	702	736
10	0	0	30	705	735

Table 8
Best, Average and Worst Results

	Best	Average	Worst
HMCR 0.2	880	1144.9	1372
HMCR 0.5	1035	1088.4	1132
HMCR 0.8	708	739.2	769

Table 9
Result between Timetable produced by ready-made software package with Timetable produced by the Proposed Algorithm

	Hard Constraints				Soft Constraints				
	H1	H2	H3	H4	S1	S2	S3	S4	Total Cost
Actual Timetable	0	0	0	1178	0	0	0	52	1230
Timetable by proposed algorithm	0	0	0	0	0	0	34	674	708

The hybridization of HSA with GD with HMCR 0.8 shows the best result over other HMCRs. The total cost produced by the best result of NGD with HMCR 0.8 is 708, which consists of cost of IsolatedLectures (S3) - 34 and cost of RoomStabililty (S4) - 674. The proposed algorithms show better result of cost quality compared to the result produced by the ready-made software package as shown in Table 9. The proposed algorithm produces zero violation of hard constraints compared to the timetable that is produced by the ready-made software package, which contain H4 (Conflicts). With no conflicts existing in the timetable, the students have more flexibility in choosing courses. In terms of soft constraints cost comparison, the actual UUMCAS A131 timetable scores zero penalties in S3 (IsolatedLectures) as the timeslot (1 hour, 1.5 hours, 2 hours or 3 hours) is not separated to half-hour assignment. The half-hour lectures assignment which is implemented in the proposed algorithm almost certainly will schedule the courses in a scattered manner, which contributes to penalties. This is also the same reason for S4 (RoomStability) for the proposed algorithm.

V. CONCLUSIONS

This paper has presented the hybridization of HSA with GD to solve CBCTT problem. The proposed approach was applied to solve the UUM CAS course timetabling problem. The quality cost of the UUM CAS course timetabling produced by the proposed algorithms is better compared to the quality cost calculated on the course timetable by the ready-made software package.

ACKNOWLEDGMENT

The UUMCAS A131 course timetable has been provided by Academic Affairs Department, UUM and this work was funded by Ministry of Higher Education of Malaysia.

REFERENCES

[1] H. Babaei, J. Karimpour, and A. Hadidi, "A Survey of Approaches for University Course Timetabling Problem," Comput. Ind. Eng. 2014, 86, 43–59.

[2] F. De Cesco, L. Di Gaspero, and A. Schaerf, "Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, and results," in Proceedings of the Seventh PATAT Conference, 2008., 2008.

[3] Z. Lu and J.-K. Hao, "Adaptive Tabu Search for course timetabling," Eur. J. Oper. Res. 2010, 200(1), 235-244.

[4] K. Shaker and S. Abdullah, "Incorporating great deluge approach with kempe chain neighbourhood structure for curriculum-based course timetabling problems," in 2nd Conference on Data Mining and Optimization, 2009, 27-28 October 149–153; DMO.

[5] R. Bellio, L. Di Gaspero, and A. Schaerf, "Design and statistical analysis of a hybrid local search algorithm for course timetabling," J. Sched., 2012, 15 (1), 49–61.

[6] H. Y. Tarawneh, M. Ayob, and Z. Ahmad, "A Hybrid Simulated Annealing with Solutions Memory for Curriculum-based Course Timetabling Problem," J. Appl. Sci. 2013, 13(2), 262–269.

[7] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "Artificial Bee Colony Algorithm for Curriculum-Based Course Timetabling Problem," ICIT 2011 5th Int. Conf. Inf. Technol., 2011.

[8] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "An Improved Artificial Bee Colony for Course Timetabling," in Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference, 2011, 9–14.

[9] Z. W. Geem, J. H. Kim, and G. V Loganathan, "A new heuristic optimization algorithm: harmony search.," Simulation 76 (2001) 60–68.

[10] G. Dueck, "New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel," J. Comput. Phys. 1993, 104(1), 86–92.

[11] M. A. Al-Betar, A. T. Khader, and Zaman. (2012) "University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm," IEEE Trans. Syst. Man, Cybern. Part C Appl. Rev., 42(5), 664–681.

[12] G. M. Jaradat and M. Ayob, "A Comparison between Hybrid Population-based Approaches for solving Post-Enrolment Course Timetabling Problems," IJCSNS Int. J. Comput. Sci. Netw. S 116 ecurity 2011, 11 (11).

[13] J. Wahid and N. M. Hussin. "Constructing population of initial solutions for curriculum-based course timetabling problem," in Innovation Management and Technology Research (ICIMTR), 2012 International Conference,2012, 585–590.