Solving Curriculum Based Course Timetabling by Hybridizing Local Search Based Method within Harmony Search Algorithm

Juliana Wahid[1] and Naimah Mohd Hussin[2]

[1] School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010 UUM Sintok Kedah, Malaysia
w.juliana@uum.edu.my
[2] Faculty of Computer Science and Mathematical, Universiti Teknologi MARA (Perlis), 02600 Arau Perlis, Malaysia
naimahmh@perlis.uitm.edu.my

Abstract. The curriculum-based university course timetabling which has been established as non-deterministic polynomial problem involves the allocation of timeslots and rooms for a set of courses depend on the hard or soft constraints that are listed by the university. To solve the problem, firstly a set of hard constraints were fulfilled in order to obtain a feasible solution. Secondly, the soft constraints were fulfilled as much as possible. In this paper we focused to satisfy the soft constraints using a hybridization of harmony search with a great deluge. Harmony search comprised of two main operators such as memory consideration and random consideration operator. The hybridization consisted three setups based on the application of great deluge on the operators of the harmony search. The great deluge was applied either on the memory consideration operator, or random consideration operator or both operators together. In addition, several harmony memory consideration rates were applied on those setups. The algorithms of all setups were tested on curriculum-based datasets taken from the International Timetabling Competition, ITC2007. The results demonstrated that our approach was able to produce comparable solutions (with lower penalties on several data instances) when compared to other techniques from the literature.

Keywords: Harmony Search, Great Deluge, Curriculum Based Course Timetabling

1 Introduction

Curriculum-based course timetabling (CBCTT) is considered as non-deterministic polynomial (NP) problem that is intractable, i.e. there is no efficient algorithm that is guaranteed to find an optimal solution for such problems [1]. This is parallel to the theory of no-free-lunch theorem [2] which states that if any prior assumptions cannot be made about the optimization problem we are trying to solve, no algorithm can be expected to perform better than any other algorithm on that problem. The design of new methods and techniques to solve CBCTT problem is a very active area of

M.W. Berry et al. (Eds.): SCDS 2015, CCIS 545, pp. 141–153, 2015.
DOI: 10.1007/978-981-287-936-3_14

research. A very promising research area is the hybridization of metaheuristics techniques [3]. This paper focuses on the hybridization of metaheuristics between the population-based method (harmony search) and local search based method (great deluge). The aim of hybridizing a local search based with 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 [4], [5].

2 Curriculum Based Course Timetabling Problem

The CBCTT problem is to create a weekly timetable of courses with meeting time by allocating the lectures to a certain number of rooms and timeslots based on the curricula [6]. The CBCTT problem definition consists of the basic entities shown in Table 1. The entities are described in the second column.

Table 1. Basic entities in curriculum based course timetabling.

Entity	Definition
Days (d)	Number of teaching days in the week (typically 5 or 6).
Timeslots (ts)	Each day is split into a fixed number of timeslots, which is equal for all days.
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,…T_P}.
Courses and Teachers	A set of *N* courses, C = {C_1,…,C_N}, each course is composed of the number of lectures (*L)*, to be scheduled and each lecture is associated to a teacher.
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,…R_M}.
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, ..., Cu_Q}

A feasible timetable is when all lectures have been scheduled for a time slot and a room, so that the hard constraints are satisfied. On the other hand, soft constraints may be violated. Then the objective of CBCTT problem is to minimize the number of soft constraint violations in a feasible solution in order to improve the solution quality.

A previous study [7] has divided the constraint into two sets: the hard constraints and the soft constraints. (1) Hard constraints: H1 - Lectures: All lectures of a course must be scheduled, and they must be 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 take place 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. (2) 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 spread 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. The quality of solution is calculated as the total penalties of the soft constraints: S1 + S2 + S3 + S5.

3 The Algorithm

The algorithm consisted of construction and improvement algorithms.

3.1 Construction Algorithm

Starting from an empty timetable, each of the lectures which were sequenced by saturation degree heuristic, followed by largest degree heuristic and went through a lecture assignment process. A conflict matrix was produced in the pre-processing phase to identify the conflict for each lecture. Sometimes lectures did not have a conflict (in certain data instances) because they existed on their own in the curricula. These lectures were assigned later, after all the lectures with conflict had already been assigned.

Each of the lectures which have already sequenced according to the above heuristics setting was randomly assigned to empty slots in an iterative process. During this process, the feasibility of all hard constraints in each iteration step was enforced. In the case where a lecture cannot be assigned to any slots due to non-available slots, it was then included in the unassigned lectures list.

The unassigned lecture list was further assigned to the timetable using nine procedures that were carried out in sequence. If a feasible timetable is found at any level of procedures, the algorithm keeps the solution and start new assignment procedures using a different random seed. Otherwise, all other procedures are repeated with a maximum of 50 iterations. After 50 iterations of unassigned lecture assignment procedures, if a feasible solution is not found, the current timetable will be discarded and the algorithm start a new assignment procedure using a different random seed. For lectures without a conflict, it was scheduled in the timetable using the same approaches. However, only two procedures were used.

The whole process was repeated 50 times to produce a population of initial solutions as the improvement algorithm used is a population-based method. Only feasible timetable (i.e. with no hard constraints) was included in the population of initial solution.

3.2 Improvement Algorithm

Harmony search. The harmony search algorithm (HSA) is a population based on metaheuristic algorithm that impersonates the musical improvisation process in which a group of musicians improvise their instruments' pitch by searching for a perfect state of harmony according to audio-aesthetic standard [8]. The HSA requires six steps as follows:

1. Determine the algorithm parameter setting - the HSA parameters required to solve the CBCTT problem are 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)
2. Memory initialization – the process of constructing the population of initial solution which is called harmony memory (HM). The number of initial solution is determined by the value of HMS stated in the first step.
3. 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) - replaced the lecture assigned by memory consideration (MC) operator.
4. Update memory with the solution found - the new solution that is better than previous solution is included in HM.
5. Determine the termination criteria - the termination criteria used is the number of the iteration process, i.e. maximum improvisations (MI) that has been defined in the first step.
6. Cadenza (musical terminology) - return the best harmony

Great Deluge. The great deluge (GD) algorithm is proposed by [9] and is motivated by the behaviour of the hiker who seeks the peak of the ground when the water level rises up during rainy season. GD is a variance of simulated annealing (SA) technique in which the differences such as GD involves fewer parameters and decrease the objective function in its acceptance rule of solutions [10].

Hybridization of Harmony Search and Great Deluge. The hybridization of GD within HSA consists of three types: (1) hybridization of HSA and GD in RC operator (NGD), (2) HSA with GD in MC operator (GDN) and (3) HSA with GD in MC and RC operator (GDGD). The N in the NGD and GDN consists of an algorithm as in the original HSA described above.

Figure 1 shows the general pseudo code of the hybridization of HSA and GD, while Table 2 shows the respective MC, RC, and UPDATE acceptance formula related to the hybridization. The NN hybridization is the original HSA implementation without local search based method hybridization.

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: $0 \leq U(0,1) \leq 0.2xPAR$
Swap timeslot: $0.2xPAR < U(0,1) \leq 0.4xPAR$
Move room: $0.4xPAR < U(0,1) \leq 0.6xPAR$
Swap room: $0.6xPAR < U(0,1) \leq 0.8\ xPAR$
Kempe chain move: $0.8xPAR < U(0,1) \leq 1\ xPAR$
MC Acceptance Formula
end if (end of memory consideration)
else (random consideration)
Move or Swap (timeslot and room)
RC Acceptance Formula
end if (end of random consideration)
end for
Step 4*: Update the new harmony in the HM*
UPDATE Acceptance Formula
end while *(****Step 5:*** *Performing termination)*
Step 6: *Cadenza (returns the best harmony ever found*

Fig. 1. General Pseudo Code of HSA and GD Hybridization

In the pseudo code, in step 1, the values of HSA parameters were initialized by the following values: HMS = 50, HMCR = 0.2, 0.5, and 0.8, PAR = 1.0, MI = 1000. In step 2, the initialization of harmony memory was the construction algorithm which already described in section 3.1.

During the improvement stage, i.e. step 3, the best solution, x^{BEST} from the harmony memory (HM) was selected and assigned to current variable, $x^{CURRBEST}$. The cost of the best solution $f(x^{BEST})$ was assigned to B which was the initial water level.

For N iterations, the improvisation step in MC operator and RC operator in the hybridization of HSA with GD algorithm constituted several neighborhood structures as follows:

- The move timeslot. With the probability between 0%×PAR and 20%×PAR, the lecture is randomly moved to any feasible timeslot in the same room.
- The swap timeslot. With the 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 the probability between 40%×PAR and 60%×PAR, the lecture is randomly moved to any feasible timeslot in a different room.
- The swap room. With the probability between 60%×PAR and 80%×PAR, the lecture is swapped with the timeslot of another lecture located in a different room.
- The kempe chain move. With the probability between 80%×PAR and 100%×PAR, the lecture is moved using a Kempe chain. A Kempe chain is defined as a set of lectures that form a connected component because the conflict in the subset of lectures belongs to two distinct periods.

Table 2. Different Acceptance formula in hybridization of HSA and GD.

Acceptance Formula			Hybridization's name
MC	**RC**	**UPDATE**	
if $f(x^{NEW}) <= f(x_{CURBEST})$ $x^{CURBEST} = x^{NEW}$ ***end if***	***if*** $f(x^{NEW}) <= f(x_{CURBEST})$ $x^{CURBEST} = x^{NEW}$ ***end if***	***if*** $f(x^{NEW}) <= f(x^{WORST})$ $x^{WORST} = x^{NEW}$ ***end if***	*NN*
if $f(x^{NEW}) <= f(x_{CURBEST})$ $x^{CURBEST} = x^{NEW}$ ***end if***	***If*** $f(x^{NEW}) <= B$ ***or*** $f(x^{NEW}) <= f(x_{CURBEST})$ $x^{CURBEST} = x^{NEW}$ ***end if***	***If*** $f(x^{NEW}) <= B$ ***or*** $f(x^{NEW}) <= f(x_{WORST})$ $x^{WORST} = x^{NEW}$ ***end if***	*NGD*
If $f(x^{NEW}) <= B$ ***or*** $f(x^{NEW}) <= f(x_{CURBEST})$ $x^{CURBEST} = x^{NEW}$ ***end if***	***if*** $f(x^{NEW}) <= f(x_{CURBEST})$ $x^{CURBEST} = x^{NEW}$ ***end if***	***if*** $f(x^{NEW}) <= B$ ***or*** $f(x^{NEW}) <= f(x_{WORST})$ $x^{WORST} = x^{NEW}$ ***end if***	*GDN*
If $f(x^{NEW}) <= B$ ***or*** $f(x^{NEW}) <= f(x_{CURBEST})$ $x^{CURBEST} = x^{NEW}$ ***end if***	***If*** $f(x^{NEW}) <= B$ ***or*** $f(x^{NEW}) <= f(x_{CURBEST})$ $x^{CURBEST} = x^{NEW}$ ***end if***	***if*** $f(x^{NEW}) <= B$ ***or*** $f(x^{NEW}) <= f(x_{WORST})$ $x^{WORST} = x^{NEW}$ ***end if***	*GDGD*

The RC operator which was selected based on 1 - HMCR probability moved or swapped the lecture at j randomly to other timeslots (whether in the same room and timeslot or different room and time slot) that were available and feasible.

In NN and NGD, for each movement of selected neighborhood structure on the MC operator, the quality of the new solution $f(x^{NEW})$ was calculated and compared with 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 cannot be accepted. In other hand, in GDN and GDGD, the better solution ($f(x^{NEW})$ whether it is less or equal to $f(x^{CURRBEST})$) is always accepted while the worse solution in which x^{NEW} is more than the best solution $x^{CURRBEST}$, the new solution is accepted if it is less or equal to the value of current water level B.

In these proposed hybridization of HSA with GD algorithms, the water level B did not use any decay rate, instead the value of B was set to the value of the updated best solution in the HM at every MI iteration. In other word, the same water level B was used within N variables iteration.

With the 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. In NN and GDN, the execution of these moves and swap were calculated and compared with the quality of the best solution, $f(x^{CURRBEST})$. If there is an improvement in which $f(x^{NEW})$ is less or equal to $f(x^{CURRBEST})$, the new solution x^{NEW} is accepted and $x^{CURRBEST}$ is set to a new solution x^{NEW}. The worse solution cannot be accepted. In NGD and GDGD, the execution of the RC operator was calculated and compared using the GD acceptance formula. The improved solution ($f(x^{NEW})$ whether it is less or equal to $f(x^{CURRBEST})$), it 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, the new solution x^{NEW} was updated to HM if the cost of new solution $f(x^{NEW})$ is less or equal to the worst solution in the HM, $f(x^{WORST})$. For HSA and GD hybridization (NGD, GDN, and GDGD), if the new solution x^{NEW} is worse than the worst solution in HM, the new solution is accepted if it less or equal to the value of current water level B. At the beginning of the next MI iteration, the water level B is set to the best solution found so far.

The whole procedure was repeated until the termination criteria (number of MI) were met and the best solution found was returned to the end of this algorithm.

4 Experimental Results

The proposed methods were coded using C++ in Microsoft Visual 2008 under Windows 7 on an Intel Machine with Core TM i7 4770 CPU and a 3.1GHz processor and 3GB RAM. Twenty-one data of instances were categorized as ITC-2007 which were available at CBCTT website (http://tabu.diegm.uniud.it/ctt). The data were used to compare the performance of NN, NGD, GDN, and GDGD. For each proposed algorithm, three HMCR, i.e. 0.2, 0.5, and 0.8 were executed 10 times for each data, by imposing 1000 iterations as the stopping condition.

Table 3 shows the penalties obtained from the best results of 10 run which were obtained by NN, NGD, GDN, and GDGD algorithm with different HMCR setting. The total penalties from Table 3 are highlighted in a bar chart as shown in Figure 2. It is apparent from Figure 2 that the lowest total penalties were obtained from NGD with HMCR 0.5 setting.

Table 3. Results of NN, NGD, GDN and GDGD with different HMCR

	NN			NGD			GDN			GDGD		
HMCR / **Data Instances**	**0.2**	**0.5**	**0.8**	**0.2**	**0.5**	**0.8**	**0.2**	**0.5**	**0.8**	**0.2**	**0.5**	**0.8**
Comp01	6	5	6	7	5	6	8	7	6	9	7	10
Comp02	151	145	143	116	81	111	144	127	127	129	125	125
Comp03	162	146	151	116	108	103	150	136	147	141	144	139
Comp04	74	73	75	81	62	57	90	88	100	105	107	105
Comp05	479	524	514	381	365	396	441	412	391	361	365	358
Comp06	118	110	113	116	100	81	127	138	152	161	149	157
Comp07	112	102	100	111	73	61	123	124	154	156	161	176
Comp08	87	81	88	91	76	58	100	115	130	129	131	128
Comp09	181	165	164	149	136	142	177	166	180	180	178	185
Comp10	82	82	85	77	52	53	102	111	119	128	132	137
Comp11	0	0	0	0	0	0	0	0	0	1	1	0
Comp12	584	557	584	385	392	454	480	482	412	424	406	416
Comp13	124	120	110	113	103	95	128	139	153	153	152	153
Comp14	108	110	116	94	81	82	120	114	118	122	118	122
Comp15	153	167	158	111	116	124	153	142	137	143	147	148
Comp16	108	89	100	108	81	68	129	129	140	153	147	160
Comp17	142	149	137	140	117	113	162	180	176	174	172	173
Comp18	113	115	115	104	106	117	115	105	100	101	106	104
Comp19	182	154	137	107	104	107	139	129	138	126	135	136
Comp20	107	117	130	90	82	86	136	135	148	143	135	140
Comp21	205	197	206	167	143	169	206	207	205	209	211	214
TOTAL	3278	3208	3232	2664	**2383**	2483	3230	3186	3233	3248	3229	3286

It can be concluded that the hybridization of HSA with GD, with the range of 50% normal acceptance formula (the movement is accepted if the cost is less or equal to the current cost) in MC operator and 50% of the GD acceptance formula in RC operator (NGD) outperformed those obtained by other HSA and GD hybridization algorithms with the lowest total penalties, i.e. 2383. NGD with HMCR 0.8 gave a better performance compared to NGD with HMCR 0.2. These results demonstrated that the use of less than 50% of GD acceptance formula in the RC operator produces better performance. In addition, these results indicated that the MC operator which used different neighborhood structures with normal acceptance formula contributed faster convergence rather than employing the neighborhood structures with the GD acceptance formula. This was verified with the result provided by GDN and GDGD, in which both of them applied GD acceptance formula in the MC operator.

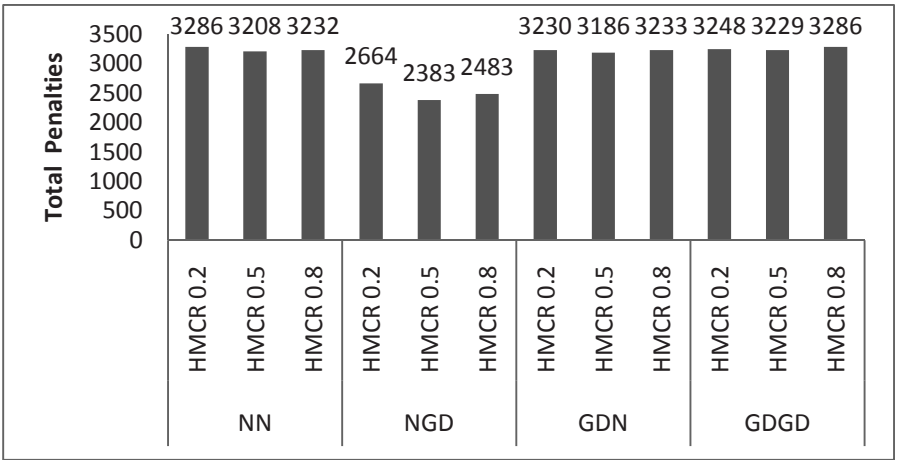

Fig. 2. Total Penalties of HSA and GD Hybridization

4.1 Experiments with Higher Number of Iterations

After further analysis on the performance graph of the NGD algorithm, it was found that the improvements are still being made towards the end of the maximum number of iterations, i.e. 1000. Therefore, the execution of NGD was extended until no further improvement for the last 1000 iterations. This is to study whether much better solutions can be found.

The execution of NGD with higher iteration was applied to all problem instances except for Comp01 and Comp05, as both of the problem instances were already at the state of optimal solution.

Table 4 shows the percentage improvement of NGD with higher iterations compared to NGD with 1000 iterations. The NGD with higher iterations was able to improve the solution for most of the problem instances except Comp05. The NGD with higher iterations was able to improve the solution by more than 10% for eleven problem instances.

4.2 Comparing with Other Approaches in Literature

Table 5 shows the comparison of NGD with higher iterations with other approaches in the literature. The objective here was to show that the hybridization of HSA and GD is able to produce a good quality and feasible solutions for CBCTT problems even though they may not produce the best results. The approaches considered include repair-based heuristic using propositional satisfiability (SAT) by [11], dynamic tabu search by [7], combination of hill climbing (HC), great deluge (GD) and simulated annealing (SA) by [12], great deluge (GD) with kempe chain neighborhood by [13], adaptive tabu search (ATS) by [14], adaptive tabu search (ATS) with numerous combination of neighborhoods by [15], integer programming by [16], threshold accepting by [17], combination of an electromagnetic-like mechanism (EM) and great

deluge (GD) by [18], propositional satisfiability (SAT) solvers and optimizers by [19], combination of simulated annealing (SA) and dynamic tabu search (DTS) by [20], and hybridization between simulated annealing and non-accepted solutions memory (SAM) by [21].

Table 4. Improvement Percentage of NGD with Higher Iterations

	NGD HMCR 0.5 1000 iterations	**NGD HMCR 0.5 Higher iterations**		
Problem instances	**Penalty**	**Penalty**	**No. of iterations**	**Improvement Percentage**
Comp01	5	5	Optimal	Optimal
Comp02	81	66	2177	18.52
Comp03	108	98	3984	9.26
Comp04	62	43	5259	30.65
Comp05	365	365	441	0.00
Comp06	100	78	5445	22.00
Comp07	73	30	7544	58.90
Comp08	76	50	4295	34.21
Comp09	136	126	2912	7.35
Comp10	52	36	5013	30.77
Comp11	0	0	Optimal	Optimal
Comp12	392	390	1893	0.51
Comp13	103	89	4882	13.59
Comp14	81	69	2309	14.81
Comp15	116	110	3495	5.17
Comp16	81	47	4544	41.98
Comp17	117	98	2456	16.24
Comp18	106	103	2333	2.83
Comp19	104	94	3719	9.62
Comp20	82	68	3611	17.07
Comp21	143	132	1981	7.69
TOTAL	**2383**	**2097**		12.00

The dashes (-) sign in Table 5 indicates the problem instances that were not experimented by the authors. The best results were highlighted in each cell. The first comparison resulted against [11] who had applied repair-based heuristic using propositional satisfiability (SAT). The NGD results were better than the repair-based heuristic in all problem instances. The NGD results were better than [7] for three problem instances (Comp02, Comp04, and Comp07), better than [16] for two problem instances (Comp01 and Comp12), better than [19] for four problem instances (Comp03, Comp05, Comp12, and Comp15), and has equal result with [21] for one problem instance (Comp04).

In Table 5, the comparisons were also made with the results obtained from the best-known solution (last column) available from CBCTT website (http://satt.diegm.uniud.

it/ctt). The NGD algorithm obtained the optimal solution for Comp01 and Comp11, while achieved competitive results with the best-known solution for the rest of the problem instances.

Table 5. Comparison of Results Obtained by the NGD (HMCR 0.5 higher iteration) with Other Approaches

Table 5. Comparison of Results Obtained by the NGD (HMCR 0.5 higher iteration) with Other Approaches

Problem Instances	NGD Penalty	[11] Penalty	[7] Penalty	[12] Penalty	[13] Penalty	[14] Penalty	[15] Penalty	[16] Penalty	[17] Penalty	[18] Penalty	[19] Penalty	[20] Penalty	[21] Penalty	Best Known Solution (Until 01/07/2015) Penalty
Comp01	5	9	5	5	5	5	5	13	5	5	5	5	5	5
Comp02	66	103	75	43	60	34	40	43	91	39	24	41	35	24
Comp03	98	101	93	72	81	70	71	76	108	76	111	66	77	64
Comp04	43	55	45	35	39	38	39	38	53	35	35	35	43	35
Comp05	365	370	326	298	321	298	298	314	359	315	1343	301	293	284
Comp06	78	112	62	41	45	47	47	41	79	50	27	43	51	27
Comp07	30	97	38	14	21	19	21	19	36	12	6	18	15	6
Comp08	50	72	50	39	41	43	43	43	63	37	37	39	46	37
Comp09	126	132	119	103	102	99	101	102	128	104	171	96	99	96
Comp10	36	74	27	9	17	16	18	14	49	10	4	15	6	4
Comp11	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Comp12	390	393	358	331	349	320	320	405	389	337	977	320	307	294
Comp13	89	97	77	66	73	65	65	68	91	61	59	64	71	59
Comp14	69	87	59	53	59	52	55	54	81	53	51	53	55	51
Comp15	110	119	87	84	82	69	-	-	-	73	111	66	68	62
Comp16	47	84	47	34	49	38	-	-	-	32	18	28	32	18
Comp17	98	152	86	83	81	80	-	-	-	72	56	71	61	56
Comp18	103	110	71	83	79	67	-	-	-	77	83	69	70	61
Comp19	94	111	74	62	67	59	-	-	-	60	57	60	62	57
Comp20	68	144	54	27	30	35	-	-	-	22	4	29	14	4
Comp21	132	169	117	103	110	105	-	-	-	95	86	89	81	74

5 Conclusion and Future Work

The overall goal of this paper was to investigate the HSA hybridization with great deluge for solving the CBCTT problem. The hybridization produced three proposed algorithms such as hybridization of GD in RC operator of HSA (NGD), hybridization of GD in MC operator of HSA (GDN), and hybridization of GD in MC and RC operator of HSA (GDGD). In addition, each proposed algorithm was executed using three different harmony memory consideration rate (HMCR) such as 0.2, 0.5, and 0.8. The performance of each proposed algorithm with different HMCR was compared to each other based on the lowest total penalties obtained. The NGD with HMCR 0.5 produced the lowest total penalties compared to all other proposed algorithms. A further execution of NGD with HMCR 0.5 using higher number of iterations was carried out to find whether the solution of each problem instances can be improved. The results showed more than 10% of improvement for half of the problem instances. The result of NGD with HMCR 0.5 (with higher number of iterations) was compared to other approaches in the literature which applied the same domain and the best-known solution available in the CBCTT website. The approach produced solutions that were better than one published results of all problem instances, while it was better on certain problem instances on certain published results. Moreover, this approach was able to obtain the optimal penalty cost for two problem instances. In the future, this proposed approach can be applied to real data CBCTT problem.

References

1. Gomes, C.P., Williams, R.: Approximation Algorithms. In: Kendall, G., Burke, E.K. (eds.) Search Methodologies Introductory Tutorials in Optimization and Decision Support Techniques. Springer Science-i-Business Media, LLC (2005)
2. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1(1), 67–82 (1997), doi:10.1109/4235.585893
3. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization:Overview and conceptual comparison. ACM Comput. Surv. 35(3), 268–308 (2003)
4. Al-Betar, M.A., Khader, A.T., Zaman, M.: University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm. IEEE Trans. on Syst. Man, Cybern. Part C Appl. Rev. 42(5), 664–681 (2012)
5. Jaradat, G.M., Ayob, M.: A Comparison between Hybrid Population-based Approaches for solving Post-Enrolment Course Timetabling Problems. IJCSNS Int. J. Comput. Sci. Netw. Security 11(11), 116 (2011)
6. Geiger, M.: Multi-criteria Curriculum-Based Course Timetabling—A Comparison of a Weighted Sum and a Reference Point Based Approach. In: Ehrgott, M., Fonseca, C., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) Evolutionary Multi-Criterion Optimization, vol. LNCS, vol. 5467, pp. 290–304. Springer, Heidelberg (2009)
7. De Cesco, F., Di Gaspero, L., Schaerf, A.: Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, and results. In: Proceedings of the Seventh PATAT Conference (2008). `http://tabu.diegm.uniud.it/ctt/DDS2008.pdf`
8. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. Simulation 76, 60–68 (2001)
9. Dueck, G.: New Optimization Heuristic: The Great Deluge Algorithm and the Recordto-record Travel. J. Comput. Phys. 104, 86–92 (1993)
10. Dreo, J., Siarry, P., Petrowski, A., Taillard, E.: Metaheuristics for Hard Optimization, pp. 153–176. Springer, Heidelberg (2006)
11. Clark, M., Henz, M., Love, B.: QuikFix: A repair-based timetable solver. In: Proceedings of the Seventh PATAT Conference (2008). `http://www.comp.nus.edu.sg/~henz/publications/ps/PATAT2008.pdf`
12. Müller, T.: ITC2007 solver description: a hybrid approach. Ann. Oper. Res. 172(1), 429–446 (2009)
13. Shaker, K., Abdullah, S.: Incorporating great deluge approach with kempe chain neighbourhood structure for curriculum-based course timetabling problems. In: 2nd Conference on Data Mining and Optimization, DMO 2009, October 27-28, pp. 149–153 (2009)
14. Lü, Z., Hao, J.-K.: Adaptive Tabu Search for course timetabling. Eur. J. Oper. Res. 200(1), 235–244 (2010)
15. Lü, Z., Hao, J.-K., Glover, F.: Neighborhood analysis: a case study on curriculum-based course timetabling. J. Heuristics, 1–22 (2010)
16. Lach, G., Lübbecke, M.: Curriculum based course timetabling: new solutions to Udine benchmark instances. Ann. Oper. Res., 1–18 (2010)
17. Geiger, M.: Applying the threshold accepting metaheuristic to curriculum based course timetabling. Ann. Oper. Res., 1–14 (2010)
18. Abdullah, S., Turabieh, H., McCollum, B., McMullan, P.: A hybrid metaheuristic approach to the university course timetabling problem (2010)

19. Asín Achá, R., Nieuwenhuis, R.: Curriculum-based course timetabling with SAT and MaxSAT. Ann. Oper. Res., 1–21 (2012)
20. Bellio, R., Di Gaspero, L., Schaerf, A.: Design and statistical analysis of a hybrid local search algorithm for course timetabling. J. Sched. 15(1), 49–61 (2012)
21. Tarawneh, H.Y., Ayob, M., Ahmad, Z.: A Hybrid Simulated Annealing with Solutions Memory for Curriculum-based Course Timetabling Problem. J. Appl. Sci. 13, 262–269 (2013), doi:10.3923/jas.2013.262.269; ISSN 1812-5654