2012 International Conference on Innovation, Management and Technology Research (ICIMTR2012), Malacca, Malaysia : 21-22 May, 2012

Constructing Population of Initial Solutions for Curriculum-Based Course Timetabling Problem

Juliana Wahid
Universiti Utara Malaysia
06010 UUM Sintok
Kedah, Malaysia
w.juliana@uum.edu.my

Naimah Mohd Hussin
Universiti Teknologi MARA (Perlis)
02600 Arau,
Perlis, Malaysia
naimahmh@perlis.uitm.edu.my

Abstract **- This paper presents an investigation of a combination of graph coloring heuristics in construction approach in University course timetabling problem (UCTP) to produce a population of initial solutions. The graph coloring heuristics were set as individual, combination of two heuristics and combination of three heuristics. In addition, several steps of courses assignment were applied to all the settings. All settings of heuristics are then tested on the same curriculum-based problem instances and are compared with each other in terms of number of population produced. The results can be used for improvement phase which are the second phase of UCTP. This approach allows generalization over a set of problems instead of producing feasible timetables for some of the problems only. Future work will use the best settings of heuristic to the improvement phase in population-based improvement algorithm.**

Keywords **– Curriculum-Based Timetabling, Graph Coloring Heuristics, Construction Phase**

I. INTRODUCTION

The university course timetabling is the tasks that assign the set of courses offered by the university to particular rooms and time slots. The assigning process needs to satisfy several constraints so that timetable can actually be carried out i.e. feasible. There are two categories of constraints in timetabling such as hard and soft constraints [1]. Hard constraints are constraints that cannot be violated, e.g. courses must be allocated into different time slots in order to avoid students and staffs clashes. While soft constraints are constraints that reflect the preferences and teaching/learning comfort expectations of the students and staffs which are not necessarily vital. Some examples of soft constraints are avoiding students having to attend three or more courses in successive time slots, and preventing students from having only one course in any day.

Similar to many combinatorial optimization problems, the solution for university course timetabling usually goes through in two stages: the *construction* and the *improvement* phases [2]. In the construction phase, the algorithm starts with an empty timetable and gradually adding one element i.e. an exam or a course into the timetable at each step of the algorithm. The initial timetable is feasible i.e. with no hard constraints but usually may have many soft constraints violations. The improvement algorithm afterward starts from the initial timetable and then tries to gradually improve it, with respect to the objective function. In each step of the improvement algorithm, some elements of the timetable may be changed hoping to achieve a better timetable.

This paper investigates the first approach of solving the curriculum based university course timetabling problem i.e. the *construction* in which the combination of graph coloring heuristics to produce population of initial solutions to the curriculum based university course timetabling problem is use. In addition to the approach, several steps of inserting courses into the timetabling are implemented to ensure that all elements in the population are feasible.

This kind of study is important to the improvement approach which is the second approach of solving the curriculum based university course timetabling problem especially for improvement approach that uses set of population of initial solutions such as genetic algorithm (GA) [3], ant colony optimization (ACO) [4], and harmony search algorithm (HSA) [5].

This study emphasizes the combination of graph coloring heuristics to identify the best sequence of heuristics for a particular problem instead of using a particular heuristic that produces feasible timetables for a few problems. This allows generalization over a set of problems instead of producing feasible timetables for some of the problems only.

This paper is organized as follows; section two provides the detailed description of CB-CTT in terms of hard and soft constraints; details of graph colouring heuristics are presented in section three; the methodology in which the combination of graph colouring heuristics was implemented are discussed in section four. Section five shows the computational results and comparison of several different combinations of graph colouring heuristics. The final section presents the conclusion and future direction.

II. THE CURRICULUM-BASED COURSE TIMETABLING

University course timetabling problem (UCTP) is a non deterministic polynomial (NP) problem that can be formulated as a combinatorial optimization problem (COP) [6-7], in which, the larger the number of lectures to be scheduled and the diversity of constraints that need to be considered, the harder the problem to be solved.

The university course timetabling as described in International Timetabling Competition 2007 (ITC-2007) [www.cs.qub.ac.uk/itc2007/] fall into two versions. The first version is the post enrolment course timetabling (PE-CTT), which the timetable is constructed based on student enrolments i.e. after students have selected which lectures they wish to attend. The second version is the curriculum-based course timetabling (CB-CTT) which recently been proposed as new track in the ITC-2007. This version will construct the timetable according to curricula published by the university.

The CB-CTT has:

- A set of N courses, $e = \{e_1,\ldots,e_N\}$, each course is composed of L same lectures to be scheduled and associated to a teacher.
- A set of P periods, $T=\{t_1,\ldots t_p\}$.
- A set of M rooms, $R=\{R_1,\ldots R_M\}$.

The CB-CTT problems deal with the assignment of a set of lectures of courses to a set of rooms and timeslots on a weekly basis, in accordance with a given set of constraints. A period p is a pair composed of a day and a timeslot. Thus, the total number of scheduling periods is the product of the number of days and the number of timeslots per day. In addition, there is a set of q curricula $CU = \{cu_1, cu_2, \ldots, cu_q\}$ where each curriculum cu_i corresponds to a group of courses such that any pair of courses in the group have students in common.

A timetable is considered feasible once all lectures of courses have been assigned to timeslots and rooms with respect to the hard constraints. In addition, a feasible timetable satisfying the four hard constraints gives a penalty cost for the violations of the four soft constraints. The main objective of the CB-CTT problem is to minimize the number of soft constraint violations in a feasible solution.

The four hard constraints and four soft constraints considered as in [8] are outlined below:

Hard Constraints

H1- Lectures: All lectures of a course must be assigned to a distinct period and a room.

H2 - Room Occupancy: Two lectures cannot be scheduled to the same room and the same period.

H3 - Conflicts: Lectures of courses in the same curriculum or taught by the same teacher must be scheduled to different periods.

H4 - Availability: If the teacher of a course is not available at a given period, then no lectures of the course can be allocated to that period.

Soft Constraints

S1 - Room Capacity: 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.

S2 - Minimum Working Days: The lectures of each course should be spread across a given number of days.

S3 - Curriculum Compactness: Lectures of courses belonging to the same curriculum should be in consecutive periods (i.e., adjacent to each other).

S4 - Room Stability: All lectures of a particular course should be assigned to the same room; otherwise, the number of occupied rooms should be less.

The quality of solution is calculated as the total penalties of the soft constraints: S1 + S2 + S3 + S4.

III. Graph Colouring Heuristics

Graph heuristics are widely studied methods which were used in sequential (or constructive) solution methods to order the events that are not yet scheduled according to the difficulties of scheduling them into a feasible timeslot (without violating any hard constraints) [9]. The difficulties are represented by the degrees of the vertices in the graph, which model the timetabling problem by representing the events as vertices and conflicts by edges.

From five level of heuristics described as in [9], the work presented in this paper investigates only the following three level heuristics, with individual heuristic, combination of two heuristic and combination of three heuristic:

- Largest degree first. Events that have a large number of conflicts with other events (i.e., a large degree) are scheduled early. The rationale is that the events with a large number of conflicts are more difficult to schedule and so should be tackled first.
- Largest weighted degree. This is a modification of the largest degree first which weights each conflict by the number of students involved in the conflict.
- Saturation degree. Events that have the least number of valid periods available are scheduled early.

IV. Methodology

This study focuses on two main sections in producing the population of initial solutions such as the order of courses and the assignment of the courses. In terms of the assignment of courses, the term lectures and courses will be used interchangeable as the courses consist of several lectures.

In order to develop the approach, firstly, the courses in CB-CTT were sequenced in order (in different settings) by the following combination(s):

1. L – largest degree only
2. W – weighted degree only
3. S – saturation degree only
4. LW - largest degree and weighted degree
5. LS - largest degree and saturation degree
6. WL - weighted degree and largest degree
7. WS - weighted degree and saturation degree
8. SL - saturation degree and largest degree
9. SW - saturation degree and weighted degree
10. LWS - largest degree, weighted degree, and saturation degree
11. LSW - largest degree, saturation degree, and weighted degree

12. WLS - weighted degree, largest degree, and saturation degree
13. WSL - weighted degree, saturation degree, and largest degree
14. SLW - saturation degree, largest degree, and weighted degree
15. SWL - saturation degree, weighted degree, and largest degree

These settings were tested with the same problem instances to find out which settings can produced the highest number of population for all problem instances.

In addition, these settings then will be getting through with the same technique of lectures assignment. On top of this approach, the courses with no conflict will firstly be identified in which it will be assigned after all other courses had already assigned. For courses with conflicts, it will pass such as the following step:

Main step: Assigned the lectures to the randomly available slots (suitable rooms and compliance to unavailability constraints) that empty (-1) in the timetable, while maintain the checking of conflicts to other lectures that scheduled in the same slot but different room.

If the main step was unable to assign all the lectures, the following steps will be executed:

Step1: Assign the unassigned lectures to the available slots (suitable rooms and compliance to unavailability constraints) that empty (-1) in the timetable, while maintain the checking of conflicts to other lectures that scheduled in the same slot but different room.

Step2: Assign the unassigned lectures to the available slots (suitable rooms and compliance to unavailability constraints) that empty (-1) in the timetable, in which the conflict to other lectures that scheduled in the same slot but different room equal to 1. The scheduled lecture will be assigned to its available slots (suitable rooms, compliance to unavailability constraints, and not conflict to other lectures) that empty (-1) in the timetable. The conflict slot then will be replaced by -1 that makes the conflict of unassigned lecture to 0.

Step 3: Assign the unassigned lectures to the available slots (suitable rooms and compliance to unavailability constraints) that not empty (there were lectures scheduled to it) in the timetable, in which the conflict to other lectures that scheduled in the same slot but different room equal to 0. The scheduled lecture will be assigned to its available slots (suitable rooms, compliance to unavailability constraints, and not conflict to other lectures) that empty (-1) in the timetable. The scheduled slot then will be replaced by the unassigned lectures.

Step 4: Assign the unassigned lectures to the available slots (suitable rooms and compliance to unavailability constraints) that not empty (there were lectures scheduled to it) in the timetable, in which the conflict to other lectures that scheduled in the same slot but different room equal to 1. The lecture that makes the conflict will be assigned to its available slots (suitable rooms, compliance to unavailability constraints, and not conflict to other lectures) that empty (-1) in the timetable and replacing the left slot with -1 that makes the conflict equal to 0. The available slots with scheduled lecture will be assigned to the empty slot and the scheduled slot then will be replaced by the unassigned lectures.

Step 5: Assign the unassigned lectures to the available slots (suitable rooms but not compliance to unavailability constraints) that empty (-1) in the timetable, while maintain the checking of conflicts to other lectures that scheduled in the same slot but different room.

Step 6: Assign the unassigned lectures to the available slots (suitable rooms but not compliance to unavailability constraints) that empty (-1) in the timetable, in which the conflict to other lectures that scheduled in the same slot but different room equal to 1. The scheduled lecture will be assigned to its available slots (suitable rooms, not compliance to unavailability constraints, and not conflict to other lectures) that empty (-1) in the timetable. The conflict slot then will be replaced by -1 that makes the conflict of unassigned lecture to 0.

Step 7: Assign the unassigned lectures to the available slots (suitable rooms but not compliance to unavailability constraints) that not empty (there were lectures scheduled to it) in the timetable, in which the conflict to other lectures that scheduled in the same slot but different room equal to 0. The scheduled lecture will be assigned to its available slots (suitable rooms, not compliance to unavailability constraints, and not conflict to other lectures) that empty (-1) in the timetable. The scheduled slot then will be replaced by the unassigned lectures.

Step 8: Assign the unassigned lectures to the available slots (suitable rooms and not compliance to unavailability constraints) that not empty (there were lectures scheduled to it) in the timetable, in which the conflict to other lectures that scheduled in the same slot but different room equal to 1. The lecture that makes the conflict will be assigned to its available slots (suitable rooms, not compliance to unavailability constraints, and not conflict to other lectures) that empty (-1) in the timetable and replacing the left slot with -1 that makes the conflict equal to 0. The available slots with scheduled lecture will be assigned to the empty slot and the scheduled slot then will be replaced by the unassigned lectures.

If first round of Step 1 to Step 8 were unable to assign all lectures, the step of *swaplocation* together with Step 1 to Step 8 will be iteratively executed. For this study, 20 iterations were sufficient. The *swaplocation* changes scheduled slots to empty slots while maintaining the checking of conflicts.

For the courses that have no conflict, it will be assigned iteratively to the timetable by the following step:

Step 9: Assign the lectures to the available slots (suitable rooms and compliance to unavailability constraints) that empty (-1) in the timetable, while maintain the checking of other same lectures that scheduled in the same slot but different room.

Step 10: Assign the lectures to the available slots (suitable rooms and not compliance to unavailability

constraints) that empty (-1) in the timetable, while maintain the checking of other same lectures that scheduled in the same slot but different room.

If first round of Step 9 to Step 10 were unable to assign all lectures that have no conflict, the step of *swaplocation* together with Step 9 to Step 10 will be iteratively executed. For this part of iterations, 10 iterations were sufficient.

V. Computational Results And Discussion

In this section, the performance of each setting of graph coloring heuristics was evaluated using 21 data instances generated in ITC-2007. The details of the problem can be found in http://tabu.diegm.uniud.it/ctt/index.php.

The approach was implemented using C++ programming in visual studio 2008 and all tests were run on an Intel Core 2 2.2 GHz processor with XP operating system. 50 runs, each with a different random number generator seed, were performed for each problem.

Table 1 shows the results of individual setting of largest degree (L), weighted degree (W), and saturation degree (S); TOTAL represents the total of feasible initial solutions that produced over 50 runs, while MIN and MAX represents the minimum cost and maximum cost of soft constraints that produced.

Table 1: Results of L, W and S

	L			W			S		
Problem Instances	Total	Min	Max	Total	Min	Max	Total	Min	Max
Comp01	50	346	646	50	365	900	50	272	525
Comp02	50	781	1409	50	757	1586	50	728	1482
Comp03	49	682	1193	50	705	1277	50	724	1157
Comp04	50	708	843	50	696	987	50	702	871
Comp05	2	1625	2027	3	1405	1769	2	1526	1999
Comp06	50	957	1276	50	952	1595	50	961	1360
Comp07	50	1080	1297	50	1051	1401	50	1092	1263
Comp08	50	780	922	50	773	962	50	787	919
Comp09	50	839	1053	50	849	1041	50	852	1032
Comp10	50	916	1071	50	884	1126	50	936	1100
Comp11	50	203	308	50	288	712	50	220	329
Comp12	47	1552	2057	50	1610	2281	46	1547	1909
Comp13	50	810	973	50	802	984	50	816	963
Comp14	50	710	913	50	733	901	50	723	889
Comp15	49	682	1193	50	705	1277	50	724	1157
Comp16	50	937	1127	50	968	1354	50	924	1150
Comp17	48	908	1315	48	887	1323	48	902	1077
Comp18	50	608	782	50	581	795	50	636	764
Comp19	50	707	1107	50	715	1246	50	706	1158
Comp20	50	1011	1280	50	1260	2495	50	1040	1437
Comp21	50	942	1285	50	952	1330	50	918	1452

In Table 1, for all settings i.e. L, W, and S, it was obvious that problem instances of COMP05, COMP12, and COMP17 were not able to produce total of 50. This would happen because of the complexity of the instances. In this paper, for the purpose of comparison, these three instances will become focus of discussion to determine the best setting, while the minimum cost and maximum cost of soft constraints will not be analysed as they were likely on average rate for each settings.

Table 2 to 7 shows the results of combination setting of two heuristics. SL in Table 6 produced the highest number of COMP05 which is 13 while other settings produced less than 4. For the instances of COMP12 and COMP17, most settings produced average rate that nearly to 50 except for WL in Table 4 for COMP17 that produced 38. From these results, setting of SL which consists of saturation degree and largest degree is the best so far compared to other settings.

Table 2: Results of LW

Problem Instances	Total	Min	Max
Comp01	50	364	843
Comp02	50	768	1474
Comp03	50	692	1062
Comp04	50	686	856
Comp05	4	1202	1910
Comp06	50	921	1378
Comp07	50	1059	1258
Comp08	50	795	966
Comp09	50	851	1017
Comp10	50	905	1065
Comp11	50	195	310
Comp12	48	1484	2037
Comp13	50	796	937
Comp14	50	729	905
Comp15	50	692	1062
Comp16	50	929	1138
Comp17	48	881	1291
Comp18	50	589	750
Comp19	50	722	955
Comp20	50	1052	1327
Comp21	50	914	1316

Table 3: Results of LS

Problem Instances	Total	Min	Max
Comp01	50	330	569
Comp02	50	769	1345
Comp03	50	702	881
Comp04	50	694	831
Comp05	4	1466	1890
Comp06	50	947	1448
Comp07	50	1043	1310
Comp08	50	790	929
Comp09	50	847	1064
Comp10	50	898	1074
Comp11	50	230	312
Comp12	46	1498	2044
Comp13	50	793	969
Comp14	50	745	903
Comp15	50	702	881
Comp16	50	949	1117
Comp17	49	902	1270
Comp18	50	583	773
Comp19	50	637	1225
Comp20	50	1042	1282
Comp21	50	928	1260

Table 4: Results of WL

Problem Instances	Total	Min	Max
Comp01	50	355	731
Comp02	50	754	1574
Comp03	50	637	1223
Comp04	50	693	904
Comp05	2	1368	1803
Comp06	50	966	1611
Comp07	50	1056	1235
Comp08	50	775	928
Comp09	50	814	1027
Comp10	50	908	1106
Comp11	50	273	678
Comp12	50	1477	2269
Comp13	50	808	979
Comp14	50	695	877
Comp15	50	687	1223
Comp16	50	964	1340
Comp17	38	920	1320
Comp18	50	586	782
Comp19	50	699	1246
Comp20	50	1181	2191
Comp21	50	927	1293

Table 5: Results of WS

Problem Instances	Total	Min	Max
Comp01	50	326	690
Comp02	50	762	1424
Comp03	50	701	1209
Comp04	50	705	934
Comp05	2	1313	1750
Comp06	50	964	1473
Comp07	50	1077	1255
Comp08	50	793	934
Comp09	50	854	1040
Comp10	50	898	1112
Comp11	50	264	660
Comp12	50	1504	2236
Comp13	50	803	943
Comp14	50	724	904
Comp15	50	701	1209
Comp16	50	936	1328
Comp17	43	901	1288
Comp18	50	636	775
Comp19	50	675	935
Comp20	50	1158	2177
Comp21	50	913	1310

Table 6: Results of SL

Problem Instances	Total	Min	Max
Comp01	50	323	489
Comp02	50	747	1356
Comp03	50	715	1129
Comp04	50	692	862
Comp05	13	1297	2173
Comp06	50	982	1273
Comp07	50	1063	1270
Comp08	50	788	944
Comp09	50	849	999
Comp10	50	920	1104
Comp11	50	215	327
Comp12	49	1542	1919
Comp13	50	818	990
Comp14	50	720	898
Comp15	50	715	1129
Comp16	50	965	1163
Comp17	48	910	1198
Comp18	50	627	776
Comp19	50	668	1047
Comp20	50	1036	1552
Comp21	50	906	1177

Table 7: Results of SW

Problem Instances	Total	Min	Max
Comp01	50	366	559
Comp02	50	779	1377
Comp03	50	690	1125
Comp04	50	717	872
Comp05	1	1296	-
Comp06	50	953	1368
Comp07	50	1059	1243
Comp08	50	780	948
Comp09	50	811	1067
Comp10	50	943	1099
Comp11	50	221	339
Comp12	49	1426	1897
Comp13	50	793	960
Comp14	50	732	888
Comp15	50	690	1125
Comp16	50	942	1129
Comp17	47	867	1098
Comp18	50	604	796
Comp19	50	670	1025
Comp20	50	1009	1560
Comp21	50	956	1470

Table 8 to 13 shows the results of combination of three heuristics. Hoping to get better total number of

population for instances COMP05 for three heuristics settings; none were higher than value 13. The highest was only 11 in Table 12 for SLW. The other settings manage to produce less than 3. For instances COMP12 and COMP17, the population that produced were merely in the average rate.

Table 8: Results of LWS

PROBLEM INSTANCES	TOTAL	MIN	MAX
COMP01	50	364	843
COMP02	50	731	1449
COMP03	50	692	1091
COMP04	50	699	859
COMP05	3	1217	1677
COMP06	50	947	1463
COMP07	50	1053	1280
COMP08	50	775	948
COMP09	50	826	1040
COMP10	50	909	1103
COMP11	50	209	339
COMP12	48	1533	2042
COMP13	50	798	981
COMP14	50	726	911
COMP15	50	692	1091
COMP16	50	935	1185
COMP17	47	878	1322
COMP18	50	607	752
COMP19	50	685	1025
COMP20	50	1054	1324
COMP21	50	944	1395

Table 9: Results of LSW

PROBLEM INSTANCES	TOTAL	MIN	MAX
COMP01	50	333	595
COMP02	50	761	1388
COMP03	50	714	970
COMP04	50	715	879
COMP05	3	1374	1484
COMP06	50	922	1398
COMP07	50	1093	1258
COMP08	50	794	945
COMP09	50	847	1029
COMP10	50	894	1075
COMP11	50	226	316
COMP12	30	1542	2005
COMP13	50	797	989
COMP14	50	710	890
COMP15	50	714	970
COMP16	50	958	1153
COMP17	49	906	1300
COMP18	50	600	779
COMP19	50	695	1222
COMP20	50	1047	1259
COMP21	50	926	1261

Table 10: Results of WLS

PROBLEM INSTANCES	TOTAL	MIN	MAX
COMP01	50	355	731
COMP02	50	772	1430
COMP03	50	682	1197
COMP04	50	715	968
COMP05	3	1338	1854
COMP06	50	954	1659
COMP07	50	1060	1267
COMP08	50	785	972
COMP09	50	825	1044
COMP10	50	919	1120
COMP11	50	264	660
COMP12	48	1499	2152
COMP13	50	800	973
COMP14	50	727	921
COMP15	50	682	1197
COMP16	50	963	1299
COMP17	37	914	1401
COMP18	50	627	781
COMP19	50	678	975
COMP20	50	1219	2222
COMP21	50	944	1284

Table 11: Results of WSL

PROBLEM INSTANCES	TOTAL	MIN	MAX
COMP01	50	325	834
COMP02	50	754	1523
COMP03	49	698	1151
COMP04	50	678	993
COMP05	2	1313	1750
COMP06	50	962	1509
COMP07	50	1085	1302
COMP08	50	775	927
COMP09	50	860	1033
COMP10	50	913	1142
COMP11	50	264	660
COMP12	50	1499	2234
COMP13	50	804	960
COMP14	50	740	934
COMP15	49	698	1151
COMP16	50	984	1342
COMP17	43	904	1310
COMP18	50	636	775
COMP19	50	655	942
COMP20	50	1313	2095
COMP21	50	956	1314

Table 12: Results of SLW

PROBLEM INSTANCES	TOTAL	MIN	MAX
COMP01	50	306	469
COMP02	50	747	1410
COMP03	50	718	1068
COMP04	50	734	909
COMP05	11	1298	2088
COMP06	50	938	1371
COMP07	50	1073	1256
COMP08	50	791	951
COMP09	50	861	1000
COMP10	50	919	1255
COMP11	50	222	331
COMP12	49	1525	1900
COMP13	50	801	997
COMP14	50	727	920
COMP15	50	718	1068
COMP16	50	950	1140
COMP17	47	890	1166
COMP18	50	618	818
COMP19	50	661	1258
COMP20	50	1018	1522
COMP21	50	941	1373

Table 13: Results of SWL

PROBLEM INSTANCES	TOTAL	MIN	MAX
COMP01	50	365	513
COMP02	50	781	1364
COMP03	49	743	1099
COMP04	50	703	861
COMP05	1	1296	-
COMP06	50	972	1413
COMP07	50	1060	1252
COMP08	50	778	957
COMP09	50	807	1051
COMP10	50	952	1135
COMP11	50	222	343
COMP12	48	1464	1920
COMP13	50	805	1001
COMP14	50	726	923
COMP15	49	743	1099
COMP16	50	947	1141
COMP17	48	889	1100
COMP18	50	604	796
COMP19	50	680	1021
COMP20	50	975	1489
COMP21	50	929	1269

The population of problem instances such as COMP05, COMP12 and COMP17, would perhaps be increased by adding several steps of lectures assignment that consider conflicts more than 1, which merely the same with step 2 and 4 in section four that only consider conflict equal to 1. However this step will take longer time as the checking of the conflicts will increase.

The best setting i.e. the SL that consists of courses ordered by saturation degree followed by largest degree have the maximum number of elements in population of initial solution. This result can afterward used in improvement phase for algorithm that uses population-based initial solutions such as GA, ACO and HSA.

VI. CONCLUSIONS

The overall goal of this paper was to investigate the use of combinations of graph coloring heuristics in UCTP for producing population of initial solutions. Result shows that combination of graph coloring heuristics i.e. saturation degree followed by largest degree, produced highest elements of population rather than the use of individual graph coloring heuristics. Next step is to apply the result of the population of initial solution to the improvement phase so that the solution will be optimize to the nearest optimal point i.e. with lowest number of soft constraints.

The finding of this study can be also generalized over other different scheduling problem such as nurse scheduling, job shop scheduling, school timetabling, etc.

ACKNOWLEDGMENT

This study was funded by Excellence Fund, Research Management Institute, Universiti Teknologi MARA, Malaysia and Universiti Utara Malaysia.

REFERENCES

[1] E. K. Burke, *et al.*, "Automated university timetabling: The state of the art.," *The Computer Journal, 40(9), 565–571.*, 1997.

[2] S. Petrovic, "Towards the Benchmarks for Scheduling," in *The International Conference on Automated Planning and Scheduling, ICAPS07,* , Rhode Island, USA 2007.

[3] R. Lewis and B. Paechter, "Application of the Grouping Genetic Algorithm to University Course Timetabling," in *Evolutionary Computation in Combinatorial Optimization.* vol. 3448, G. Raidl and J. Gottlieb, Eds., ed: Springer Berlin / Heidelberg, 2005, pp. 144-153-153.

[4] K. Socha, *et al.*, "A max-min ant system for the university course timetabling problem," *In Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002).Lecture Notes in Computer Science, Vol. 2463. Springer-Verlag, Berlin, pp. 1-13,* 2002.

[5] M. Al-Betar and A. T. Khader, "A harmony search algorithm for university course timetabling," *Annals of Operations Research,* pp. 1-29, 2010.

[6] A. Colorni, *et al.*, "Heuristics from nature for hard combinatorial optimization problems," *International Transactions in Operational Research, 3(1): 1-21, January 1996,* 1996.

[7] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization:Overview and conceptual comparison," *ACM Computing Surveys, 35(3):268-308, 2003,* 2003.

[8] L. Di Gaspero, *et al.*, "The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3) " (Technical Report QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0/1). School of Electronics, Electrical Engineering and Computer Science, Queens

University, Belfast (UK), August 2007. ITC-2007 site: http://www.cs.qub.ac.uk/itc2007/.2007.

[9] E. K. Burke, *et al.*, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research 176 (2007) 177–192*, 2007.