Implementing Kempe Chain Neighborhood Structure in Harmony Search for Solving Curriculum Based Course Timetabling

Juliana Wahid
School of Computing
Universiti Utara Malaysia
Kedah, Malaysia
w.juliana@uum.edu.my

Naimah Mohd Hussin
Faculty of Computer Science and Mathematical
Universiti Teknologi MARA
Perlis, Malaysia
naimahmh@perlis.uitm.edu.my

Abstract— **An essential aspect that contributes to the success of meta-heuristic algorithm over a curriculum-based course timetabling problem is determined by the neighborhood structure used. The basic neighborhood structures such as move and swap between lectures has no method for escaping from local minima or optimum that restricts the improvement of current solutions. The aim of this paper is to implement Kempe chain neighborhood structure together with the other neighborhood structures in harmony search algorithm for solving curriculum-based course timetabling. The result shows significant improvements in the solution quality.**

Keywords-component; Harmony Search Algorithm, Kempe Chain, Curriculum Based Course Timetabling

I. INTRODUCTION

The implementations of meta-heuristic algorithm for solving curriculum-based course timetabling (CBCTT) has been emerging from the second International Timetabling Competition (ITC2007) in the year 2007. The meta-heuristics include tabu search [1-2], great deluge [3], simulated annealing [4-5], and ant bee colony [6-7]. All these meta-heuristics have the common aspect of implementation which is the neighborhood structure. Even though the term of neighborhood structure in each meta-heuristic slightly differs with each other, the function remains the same. The basic neighborhood structures applied in the previous works [8] were move lectures to other timeslot and swap lectures between timeslots. These basic neighborhood structures have no method for escaping from local minima or optimum. The solution can easily stuck and no improving neighbors are available [9]. The kempe chain has been proved to make uphill and downhill moves away from the stuck solution to improve the solution quality [10].

Harmony search algorithm (HSA) is a meta-heuristic 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 [11].

The performance of HSA was studied for solving CBCTT [8], in which two basic neighborhood structures, i.e. move and swap were used in the pitch adjustment operator. Unfortunately, the result of the solution was slowly converged from the initial cost. The objective of this paper is to implement the Kempe chain neighborhood structure together with the other two basic neighborhood structures in the pitch adjustment operator of HSA. Next, the experimentation will be executed using the same setting and data instances used in [8].

The rest of this paper is organized as follows; section II provides the detailed description of CBCTT problem. The approach in tackling the problem is described in Section III, followed by presenting the results in Section IV. Finally, the conclusions are presented in Section V.

II. CURRICULUM BASED COURSE TIMETABLING

The curriculum-based course timetabling (CBCTT) problem is an alternative of a university timetabling problem which creating a weekly timetable by allocating lectures for several university courses to a certain number of rooms and time periods based on curricula. The available constraints for this variant of university timetabling problem are solely based on the definition of the curricula [12]. Opposite to post-enrollment based course timetabling problems in which students specifically register for courses they wish to attend. The CBCTT problem definition consists of the basic entities shown in Table I.

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

There are four hard constraints in this problem:

- Lectures: All lectures of a course must be scheduled, and they must be assigned to distinct periods.
- Conflicts: Lectures of courses in the same curriculum or taught by the same teacher must be all scheduled in different periods.

- RoomOccupancy: Two lectures cannot take place in the same room in the same period.
- 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 for that period.

There are four soft constraints in this problem:

- RoomCapacity: For each lecture, the number of students that attend the course must be less than or equal the number of seats of all the rooms that host its lectures.
- RoomStability: All lectures of a course should be given in the same room.
- MinWorkingDays: The lectures of each course must be spread into a given minimum number of days.
- Curriculum compactness: For a given curriculum, a violation is counted if there is one lecture not adjacent to any other lecture belonging to the same curriculum within the same day.

The quality of the solution is calculated as the total penalties of the soft constraints: RoomCapcity + RoomStability + MinWorkingDays + Curriculum compactness

TABLE I. 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 day.
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={$t_1, \ldots t_p$}.
Courses and Teachers	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.
Rooms	Each room has a capacity, expressed in terms of the number of available seats, and a location expressed as an integer value representing a separate building. Some rooms may not be suitable for some courses (because they miss some equipment). A set of *M* rooms, R={$R_1, \ldots R_M$}.
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. A set of q curricula CU = {cu_1, cu_2, ..., cu_q}

III. METHODS

The method used in this paper is based on the HSA procedures with Kempe chain neighborhood structure mixed with technique of using the next best solution after no improvement iteration as shown in Fig 1. In addition this study used the best improvement acceptance, i.e. only equal or better solution is accepted for any move of events.

A. Initialize the problem and HSA parameters

The CBCTT problem is formulated and initialized in several data structures as follows:

- Course Correlation matrix: is a matrix with size C x C where the element > 0 if courses in the same curriculum and if more than one course taught by the same teacher or 0 otherwise.
- Course period matrix: is a binary matrix with size C x P where the element contains either 1 if and only if course C not available in period P or 0 otherwise.
- Course room matrix: is a binary matrix with size C x R where the element contains either 1 if and only if course C and room R is compatible with both aspects of size and features or 0 otherwise.
- Courseperiodavail matrix C x (P xR) contains either 1 if and only if course C available in period P in room R or 0 otherwise.

The following parameters of the HSA that identified by Geem [13] which required to solve the optimization problem are also specified in this step:

- Harmony memory size (HMS) - number of solution vectors concurrently handled.
- Harmony memory considering rate (HMCR) – ($0 \leq$ HMCR ≤ 1) where HSA picks one value randomly from memory.
- Pitch adjusting rate (PAR) - ($0 \leq$ PAR ≤ 1) is the rate where HSA fine-tunes the value which was originally picked from memory.
- Maximum improvisation (MI) - number of iterations or stopping criterion.

The harmony memory (HM) is a memory location where all the solution vectors (sets of decision variables) and corresponding objective function values are stored. The function values are used to evaluate the quality of solution vectors. In CBCTT, the HM consists of sets of timetable solution and corresponding of costs of soft constraints, i.e. room capacity, room stability, minimum working days and curriculum compactness. As surveyed by Geem [14], HMS of value range from 30 to 100 was frequently used in literature.

```
Step 1: Initialize the HSA and and its parameters (HMS,HMCR, PAR, MI)
Step 2: Initialize-HM{x1,…, xHMS}
while not termination criterion specified by MI do
  Select the best harmony xbest ∈(x1,…, xHMS ).
  Step 3: Improvisation
  for j=1,…, N do // N is the number of decision variables
    if U(0,1) ≤ HMCR then accept event in the best harmony
        (memory consideration)
        PAR=1 (pitch adjustment)t
          Move-timeslot: 0< U(0 ,1 ) ≤ 0.30xPAR
          Swap-timeslot: 0.30xPAR<U(0,1)≤ 0.60xPAR
          Kempe chain : 0.60xPAR< U(0 ,1 ) ≤ 1
    else //random consideration
      Select randomly courses that available and feasible at j
    end if
  end for
  Step 4: Update the new harmony
    if f(x NEW) ≤ f(x worst) then
        x worst= x new
    end if
  Step 5: Consider other harmony in the population
      if f(x NEW) >= f(x best) then (no improvement after 1 MI iteration)
        Select next best harmony for next MI iteration
      end if
end while (check the stop criterion)
```

Figure 1. Framework of Algorithm

The HSA parameter setting used in this study are as follows:

- Harmony memory size (HMS) - 50
- Harmony memory considering rate (HMCR) – 0.9
- Pitch adjusting rate (PAR) – 1.0, with multi pitch with three procedures such as follows:
  - Move-timeslot: 0< U(0 ,1) ≤ 0.30xPAR
  - Swap-timeslot: 0.30xPAR<U(0,1)≤ 0.60xPAR
  - Kempe chain : 0.60xPAR< U(0 ,1) ≤ 1
- Maximum improvisation (MI) – first phase: 10, Second phase: 50

B. Initialize the harmony memory

In this step, the harmony memory (HM) matrix is filled with as many randomly generated solution vectors as the HMS. In the CBCTT model, the randomly generated solution vectors undergo a validation process to verify that they feasible, i.e. not violated the hard constraints (lectures, conflicts, availability and room occupation).

This step constructs maximum feasible timetabling solutions for CBCTT as determined by HMS. HM is filled with these solutions and sorted according to the minimum total cost of soft constraints. The technique used to generate random HM solutions is assigning the courses by using the combination of graph heuristic that produced maximum numbers of solutions proposed by [15] which combines saturation degree with largest degree. In this technique, the course with the least available period and largest number soft conflicting students is scheduled first.

C. Improvise a new harmony (memory consideration, random consideration, and pitch adjustment)

a) Memory consideration

The memory consideration selects feasible locations of the courses to be scheduled in the new harmony solution, $\boldsymbol{x} = (x1, x2, \ldots, xN)$, from the solutions stored in HM with the probability of HMCR. In memory consideration, the value of the course number for new solution was selected from all other course numbers located in the same column of the HM. At first, this research used the highest occurrence of the course number to be scheduled in a new harmony with probability of HMCR. In certain situations, if there are same numbers of highest occurrence for any course number, the least available period and highest conflicts for that course were used to choose between the course numbers. This technique however cannot find a feasible timetable (this happens in some medium and large data instances). To overcome this limitation, this study implements methods from [16] which select the best harmony stored in the harmony memory. In another word, the best solution from the population is selected to be pitched in the pitch adjustment operator. In addition, the next best solution will be selected if the current best solution did not improve in terms of their quality solution after one or several MI iterations.

b) Random consideration

The remaining events that have not been scheduled by memory consideration will select randomly any courses that available and feasible to be scheduled in the new harmony solution with probability (1-HMCR).

c) Pitch adjustment

In CBCTT the pitch adjustment operator works similar to neighborhood structures in local search-based methods. There are three procedures used in this study such as follows:

- The *move-timeslot.* An event that meets the probability of 0%×PAR and 30%×PAR is randomly *moved* to any feasible timeslot where the room is not changed.
- The *Swap-timeslot.* An event that meets the probability between 30%×PAR and 60%×PAR is swapped with the timeslot of another event, while the rooms of both events are not changed.
- The *Kempe chain move.* An event that meets the probability between 60%×PAR and 100%×PAR is moved using Kempe chain.

Each course scheduled out of memory consideration is examined as to whether it should be pitch adjusted with probability by the above three procedures.

A Kempe chain is defined as a set of lectures that form a connected component because of conflict in the subset of lectures that belong to two distinct periods. Let K be a Kempe chain with respect to two periods ti and tj and Li (Lj) be the set of lectures in period ti (tj), a Kempe chain interchange produces an assignment by replacing Li with $(Li \setminus K) \cup (Lj \cap K)$ and Lj with $(Lj \setminus K) \cup (Li \cap K)$[17]. The condition of Kempe chain to be moved is at least three lectures are involved, i.e., $|K| \geq 3$. Example of Kempe chain in Fig 2 below is $K = \{8, 11, 26\}$ because course 26 have conflict with course 8 and 11.

This Kempe chain can be moved because an empty timeslot (-1) available to support the move. The output for the move is course 8 will be swapped with course 26, while course 11 will be swapped with the empty timeslot (-1).

D. Update the harmony memory

If the new harmony vector, $\boldsymbol{x}= (x1, x2, \ldots, xN)$, is better than the worst harmony stored in HM in terms of the objective function value, the new harmony vector is included in the HM, and the worst harmony vector is excluded from the HM.

E. Select the next best harmony

For the purpose of next MI iteration, If the new harmony vector, $\boldsymbol{x}= (x1, x2, \ldots, xN)$, is equal or higher (no improvement) than the best harmony so far stored in HM in terms of the objective function value, the next MI iteration will use next best harmony in the HM. This step enforces that all the other harmony in the HM was actually being considered in the improvisation process rather than using only the best harmony all the time.

F. Check the stop criterion

Steps 3, 4 and 5 of the framework are repeated until the number of maximum improvisation (MI) is met. For the purpose of this study two cycles of iterations, i.e. 10 and 50 were used to observe whether the solutions are improving or not.

IV. Experimental Results

The performance of HSA for solving the CBCTT problem was evaluated using 14 data instances that available at the Curriculum-based Course Timetabling website (http://tabu.diegm.uniud.it/ctt). The instances have different number of features, as shown in Table II, such as the instance name, 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), min and max lectures per day per curriculum (MML), number of unavailability constraints (UC) and room constraints (RC).

The proposed method is coded using C++ in Microsoft Visual 2008 under Windows Vista on an Intel Machine with Core TM and a 2.16GHz processor and 1GB RAM. Table III shows the experimental results of the HSA algorithm with basic neighborhood structures in [8] for 10 iterations and 50 iterations (in column 3 and 5) , while at column 4 and 6, it shows the result of an HSA algorithm with the addition of Kempe chain as neighborhood structures.

The implementation of Kempe chain in HSA shows better improvement from the initial solution even at the early 10th iteration. At the 50th iteration, the solution quality improves better. For example, Comp11 started with 215 penalties in its initial solution. The basic neighborhood structures in HSA improve it to 179 penalties at 50th iterations which show a 17 % improvement. While the addition of Kempe chain in HSA improves Comp11 by 93% to become 14 penalties at 50th iterations. Figure 3 and 4 show the performance of HSA with or without the Kempe chain for comp01 and comp11 problem instances respectively.

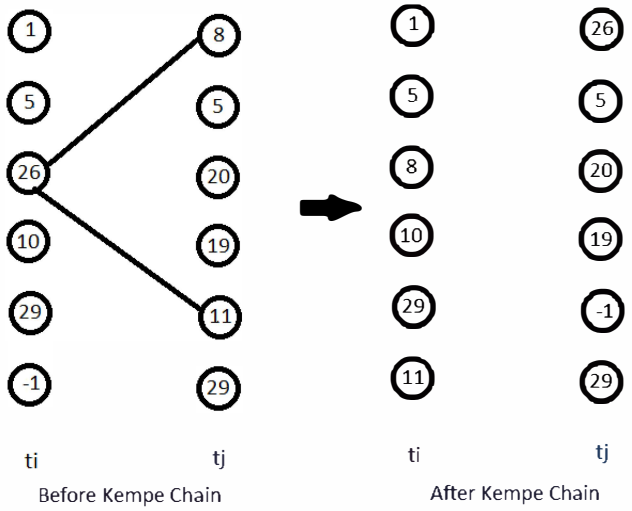

Figure 2. Kempe Chain Move

TABLE II. Features of Curriculum Based Course Timetabling Data Instances

Instance	C	L	R	PpD	D	Cu	MML	UC	RC
comp01	30	160	6	6	5	14	2–5	53	23
comp02	82	283	16	5	5	70	2–4	513	167
comp03	72	251	16	5	5	68	2–4	382	149
comp04	79	286	18	5	5	57	2–4	396	177
comp05	54	152	9	6	6	139	2–4	771	73
comp06	108	361	18	5	5	70	2–4	632	234
comp07	131	434	20	5	5	77	2–4	667	308
comp08	86	324	18	5	5	61	2–4	478	197
comp09	76	279	18	5	5	75	2–4	405	170
comp10	115	370	18	5	5	67	2–4	694	257
comp11	30	162	5	9	5	13	2–6	94	17
comp12	88	218	11	6	6	150	2–4	1368	72
comp13	82	308	19	5	5	66	2–3	468	116
comp14	85	275	17	5	5	60	2–4	486	105

TABLE III. Experimental Result Of HSA For Curriculum Based Course Timetabling

Problem instances	Initial Solution	HSA (10 iterations)	HSA with Kempe chain (10 iterations)	HSA (50 iterations)	HSA with Kempe chain (50 iterations)
	Penalty	Penalty	**Penalty**	Penalty	**Penalty**
Comp01	323	323	**141**	322	**78**
Comp02	747	747	**374**	732	**263**
Comp03	715	701	**380**	665	**280**
Comp04	692	617	**345**	577	**198**
Comp05	1297	1297	**946**	1297	**761**
Comp06	982	934	**460**	879	**387**
Comp07	1063	930	**559**	930	**519**
Comp08	788	663	**343**	645	**199**
Comp09	849	730	**599**	685	**599**
Comp10	920	868	**453**	816	**241**
Comp11	215	208	**73**	179	**14**
Comp12	1542	1398	**949**	1398	**779**
Comp13	818	731	**441**	694	**251**
Comp14	720	707	**425**	702	**260**

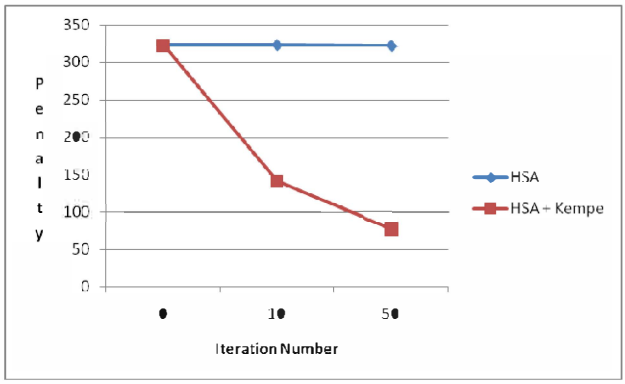

Figure 3. Performance of Kempe Chain Move in HSA for comp01 problem instance

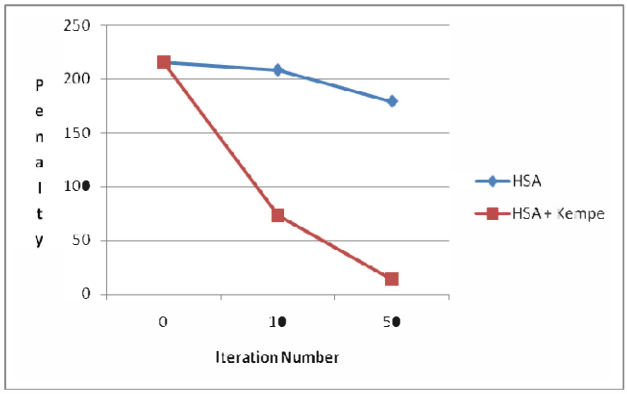

Figure 4. Performance of Kempe Chain Move in HSA for comp11 problem instance

In this section we carried out higher number of iterations to illustrate the performance of the Kempe chain for all the problem instances. The number of iterations used was 300. This number of iterations were estimated as sufficient because comp01 and comp11 problem instances (which has the smallest number of unavailability constraints (UC) and room constraints (RC) as shown in Table I) produces no improvement solution in average after 250 iterations. Using this as a point of reference, all other problem instances were executed in 300 iterations. Table IV shows the results for the CBCTT using HSA with Kempe Chain compared to the best known solution available at http://tabu.diegm.uniud.it/ctt. Even though not comparable with the best known solution, the competitive result shows that HSA with Kempe chain would yield positive results. The enhancing of this method would be in terms of changing the acceptance method of solution, i.e. such as used in simulated annealing and perhaps incorporating the HSA with local search based meta-hueristic such as done in [16] .

TABLE IV. EXPERIMENTAL RESULT OF HSA FOR CURRICULUM BASED COURSE TIMETABLING FOR HIGHER ITERATIONS

Problem instances	**Initial Solution**	**HSA with Kempe chain**	**Best Known Solution (Until 10/6/2013)**	
	Penalty	**Penalty**	**Penalty**	**Method Used**
Comp01	323	**35**	5	Tabu Search
Comp02	747	**175**	24	SAT-based
Comp03	715	**214**	66	Local Search
Comp04	692	**95**	35	Local Search
Comp05	1297	**694**	290	SA
Comp06	982	**191**	27	SAT-based
Comp07	1063	**304**	6	SAT-based
Comp08	788	**128**	37	Other
Comp09	849	**261**	96	Tabu Search
Comp10	920	**207**	4	SAT-based
Comp11	215	**6**	0	Tabu Search
Comp12	1542	**694**	300	SA
Comp13	818	**170**	59	Tabu Search
Comp14	720	**164**	51	Mathematical Programming

V. CONCLUSIONS

This paper presented the harmony search algorithm with the addition of Kempe chain neighborhood as one of the neighborhood structures for solving the CBCTT problems. Results from the experiments have shown the algorithm were capable of improving the quality of solution. Further higher iterations of the algorithm also produce results that competitive to the best known solution.

REFERENCES

[1] F. D. Cesco, *et al.*, "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>. 2008.

[2] Z. Lü and J.-K. Hao, "Adaptive Tabu Search for course timetabling," *European Journal of Operational Research, Volume 200, Issue 1, 1 January 2010, Pages 235-244,* 2010.

[3] 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. DMO '09. 27-28 Oct. 2009*, 2009, pp. 149-153.

[4] R. Bellio, *et al.*, "Design and statistical analysis of a hybrid local search algorithm for course timetabling," *Journal of Scheduling,* vol. 15, pp. 49-61, 2012/02/01 2012.

[5] H. Y. Tarawneh, *et al.*, "A Hybrid Simulated Annealing with Solutions Memory for Curriculum-based Course Timetabling Problem," *Journal of Applied Sciences 13 (2): 262-269, 2013 ISSN 1812-5654 / DOI: 10.3923/jas.2013.262.269,* 2013.

[6] A. L. Bolaji, *et al.*, "Artificial Bee Colony Algorithm for Curriculum-Based Course Timetabling Problem,"

ICIT 2011 The 5th International Conference on Information Technology, 2011.

[7] A. L. Bolaji*, et al.*, "An Improved Artificial Bee Colony for Course Timetabling," in *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference on*, 2011, pp. 9-14.

[8] W. Juliana and M. H. Naimah, "Harmony Search Algorithm for Curriculum-Based Course Timetabling Problem," in *2012 International Conference on Soft Computing and Software Engineering. Editor: Mehdi Bahrami, JSCSE-eISSN:2251-7545, San Francisco, CA, USA, 2013. Doi: 10.7321/jscse.v3.n4.55.*, San Francisco, California, USA, 2013.

[9] E. K. Burke*, et al.*, "Hybrid variable neighbourhood approaches to university exam timetabling. ," *Technical Report NOTTCS-TR-2006-2, University of Nottingham, School of CSiT, 2006.,* 2006.

[10] J. M. Thompson and K. A. Dowsland, "A robust simulated annealing based examination timetabling system," *Computers & Operations Research,* vol. 25, pp. 637-648, 1998.

[11] Z. W. Geem*, et al.* (2001) A new heuristic optimization algorithm: harmony search. . *Simulation 76, 60–68 (2001)*. Available: http://sim.sagepub.com/content/76/2/60.abstract

[12] M. Geiger, "Multi-criteria Curriculum-Based Course Timetabling—A Comparison of a Weighted Sum and a Reference Point Based Approach," in *Evolutionary Multi-Criterion Optimization*. vol. 5467, M. Ehrgott*, et al.*, Eds., ed: Springer Berlin / Heidelberg, 2009, pp. 290-304.

[13] Z. W. Geem, "State-of-the-Art in the Structure of Harmony Search Algorithm," in *Recent Advances In Harmony Search Algorithm*. vol. 270, Z. Geem, Ed., ed: Springer Berlin / Heidelberg, 2010, pp. 1-10.

[14] Z. W. Geem, "Optimal cost design of water distribution networks using harmony search.," *Engineering Optimization 38, 259–280 (2006),* 2006.

[15] W. Juliana and M. H. Naimah, "Constructing Population of Initial Solutions for Curriculum-Based Course Timetabling Problem," in *International Conference on Innovation, Management and Technology Research (ICIMTR2012), 21-22 May 2012*, Melaka, Malaysia, 2012.

[16] M. Al-Betar and A. T. Khader, "A hybrid harmony search for university course timetabling," in *Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2009) 10-12 August 2009, Dublin, Ireland*, 2009.

[17] M. Chiarandini*, et al.*, "An effective hybrid algorithm for university course timetabling," *Springer Science + Business Media, LLC 2006,* 2006.