Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 13 (2): 262-269, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.262.269

A Hybrid Simulated Annealing with Solutions Memory for Curriculum-based Course Timetabling Problem

[1]H.Y. Tarawneh, [1]Masri Ayob and [2]Zulkifli Ahmad
[1]Data Mining and Optimization Research Group, Center of Artificial Intelligence Technology, Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
[2]School of Linguistics and Language Studies, Faculty of Social Sciences and Humanities, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

Abstract: This study proposes a hybrid Simulated Annealing with solutions memory (SAM) to solve university course timetable problems. Simulated Annealing (SA) is one of the popular meta-heuristic algorithms for solving combinatorial optimization problems. However, SA could get trapped in local optimum, especially when the temperature becomes very low. In order to escape from this local optimum, this hybrid work tried to jump to another promising region using not accepted solutions saved in the memory .The computational results tested on ITC 2007 course timetabling benchmark datasets showed that SAM, can consistently produce good quality solutions, which are comparable to other approaches tested on ITC 2007 datasets.

Key words: Simulated annealing, memory strategy, course timetabling problem, heuristic, optimization, scheduling, educational timetabling

INTRODUCTION

Timetabling is defined as ''the allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way as to satisfy as nearly as possible a set of desirable objectives'' (Wren, 1996). This work focus on university course timetabling problem that assign a set of courses within a given number of rooms and time periods (McCollum and Ireland, 2006). Generating a course timetable manually often requires a lot of time, effort and usually it is hard to satisfy all constraints. Thus, automating the generation of course timetables is still a challenging task (McCollum and Ireland, 2006).

Many researchers have focused on solving this problem using meta-heuristics. The "Meta-heuristics are typically high-level strategies which guide an underlying, more problem specific heuristic, to increase their performance" (Blum and Roli, 2003). Some example of common mela-heuristics approaches are: ant colony optimization (Eley, 2007), genetic algorithms (Ueda *et al*., 2004), tabu search (Alvarez-Valdds *et al*., 2002), evolutionary search (Beligiannis *et al*., 2008) and Simulated Annealing (SA) (Abramson *et al*., 1999).

SA is one of the popular meta-heuristic algorithms due to its ease of implementation and ability to solve many combinatorial optimization problems. However, the disadvantages are that, it could still get trapped in local optimum and take longer time to find good quality solutions (Xinchao, 2011). Several researchers have tried to overcome this drawback. For examples, Azizi *et al*. (2009) proposed a hybrid simulated annealing with evolution-based diversification approach called SAMED to solve job shop scheduling problem. SAMED hybridized SA, three types of memories and GA based crossover component. The first two types of memories are short term memories (tabu list and seed memory list), while the third type is long term memory. SAMED used short term memory (tabu list) to temporarily save the accepted solutions to avoid the recycling; whilst the second short term memory (seed memory) is to keep track of the best solutions with lowest objectives functions for further improvement. The differences between our study with SAMED is in the memory part. We save the unaccepted solutions to jump to other promising region when the search got trapped in local optimum, whilst the SAMED save the best accepted solutions. Gao *et al*. (2006) presented a hybrid meta-heuristic algorithm by combined

Corresponding Author: Masri Ayob, Data Mining and Optimization Research Group, Center of Artificial Intelligence Technology, Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
Tel: +603 8921 6741 Fax: +603 8921 6184

the characteristic of simulated annealing, genetic algorithm and chaos strategy to solve TSP problem. The experimental results showed that the hybrid meta-heuristic algorithm (CASAGA) is quite flexible with satisfactory results.

Kolonko (1999) investigated the cooling schedules for a wide range of examination timetabling problems and proposed that, very slow cooling schedule should be used. Thompson and Dowsland (1995) and Swarnkar and Tiwari (2004) hybridized SA with tabu search to avoid revisiting solutions. Jeon and Kim (2004) proposed UMOSA algorithm that use a strategy called the criterion scalarizing approach since the probability to accept new solution must take into account the distance between the old and the new move. Elmohamed *et al.* (1998) employed a simulated annealing with different cooling schedules (geometric, adaptive and adaptive with reheating). The experiment results showed that using SA with adaptive cooling schedules with reheating is able to generate competitive results comparing with other state-of-the-art techniques and outperformed other SA approaches. Therefore, the use of reheating function might be good to escape from local optimum, whilst the search could revisit the same local optimum again after several iterations. Nevertheless, The use of memory to save the not accepted solutions and reuse one of them when the search trapped in local optimum could avoid to re-trapped in the same local optimum again and escape from local optimum.

This study aims to solve the problem of trapping in local optimum especially when the temperature in SA are very low. The low temperature leads SA to accept only improved solutions. This means more chance to get trap in local optimum. The question is:

- How to escape from local optimum, if the search got trapped in it?

Hybridize the simulated annealing with other techniques may be a reasonable good answer for the above question. However, this might not be efficient to deal with a bad solution structure that leads the search to get trapped in bad local optimum (very hard to escape). Therefore, this study aims to deal with this problem by saving several unaccepted solutions; and to reuse them when the search got trapped in the local optimum, in order to jump to other promising region.

The contribution of this work is to enhance the performance of SA by escaping from local optimum using solutions memory by saving non-accepted solutions and reuse it when the search got trapped in local optimum.

PROBLEM DESCRIPTION

The Curriculum-based Course Timetabling problem for the ITC-2007 is about the scheduling all lectures for a set of courses into a weekly timetable. Each lecture of a course must be assigned to periods and rooms in accordance to a given set of constraints. The employed method of determining the schedule must be able to satisfy the hard constraints and to minimize the soft constraints violations. This problem has four hard constraints H1-H4 and four soft constraints S1-S4 as follows (Gaspero *et al.*, 2007):

Hard constraints:

H1: Lectures. All lectures of a course must be scheduled to a distinct periods
H2: Room Occupancy. Any two lectures cannot be assigned to the same period and the same room
H3: Conflicts. Lectures of courses in the same curriculum or taught by the same teacher cannot be schedule in the same period
H4: Availability. If the teacher of a course is not available at a given period, then no lectures of the course can be assigned to that period

Soft constraints:

S1: Room Capacity. For each lecture, the number of students attending the course should not be greater than the capacity of the room hosting the lecture
S2: Minimum Working Days. The lectures of a course should be spread into the given minimum number of days
S3: Room Stability. All lectures of a course should be scheduled at the same room. If this is impossible, the number of occupied rooms should be as few as possible
S4: 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, which means the agenda of students should be as compact as possible

ALGORITHM

This work is divided into two stages. The first stage is to construct initial solution using constructive algorithms. At this stage, the feasible initial solution is constructed by satisfying all hard constraints (H1-H4) using sequential greedy heuristic as in Lu and Hao (2010). There is no guarantee that this greedy heuristic can always find feasible solution. Therefore, we used steepest

decent to search for a feasible solution if the solution still not feasible. The second stage is to minimize soft constraints violations. At this stage, we used the enhanced Simulated Annealing with memory (SAM).

Simulated annealing with memory: SA has been widely used to solve combinatorial optimization problems. It accepts the new solution when the objective value is lower than or equals to the current one. There is a probability to accept worse quality solutions using probability acceptance criteria:

$$p(X)=e^{\frac{-\Delta f}{T_1}}$$

This probability function controls the acceptance of new solution, where $\Delta f = f(s^*)-f(s)$, the difference between the quality of new solution $f(s^*)$ and the current solution $f(s)$. The current temperature, T_1 is iteratively reduced according to the cooling schedule with a given cooling rate α for each iteration or level until this temperature reaches final minimum temperature, which is close to Zero, T_{min}.

When the SA temperature becomes very low, the SA behaves like descent heuristic (accept only improve solution). Thus, the search might easily be trapped in local optimum. Therefore, this work will save several non-accepted solutions (Sol_m) in the memory during the search (before the non- improvement) and randomly employs one of these solutions when local optimum is met. The memory size is fixed and is updated by replacing the worst solutions in memory with new ones (Fig. 1).

Our SA algorithm involves: neighborhood structure, temperature, cooling schedule, termination criteria and memory.

The algorithm terminates when one of the three cases occurs:

Case I: The minimum temperature T_{min} closed to zero (frozen stage) (T_{min} = 0.0001).

Case II: Number of iterations.

Case III: Timeout based on ITC 2007 course timetabling track 3 stopping condition.

Let:

- Sol be an initial solution,
- f (Sol) as the quality of Sol,
- Sol* as the best found solution so far;
- Iter as the total iterations number,

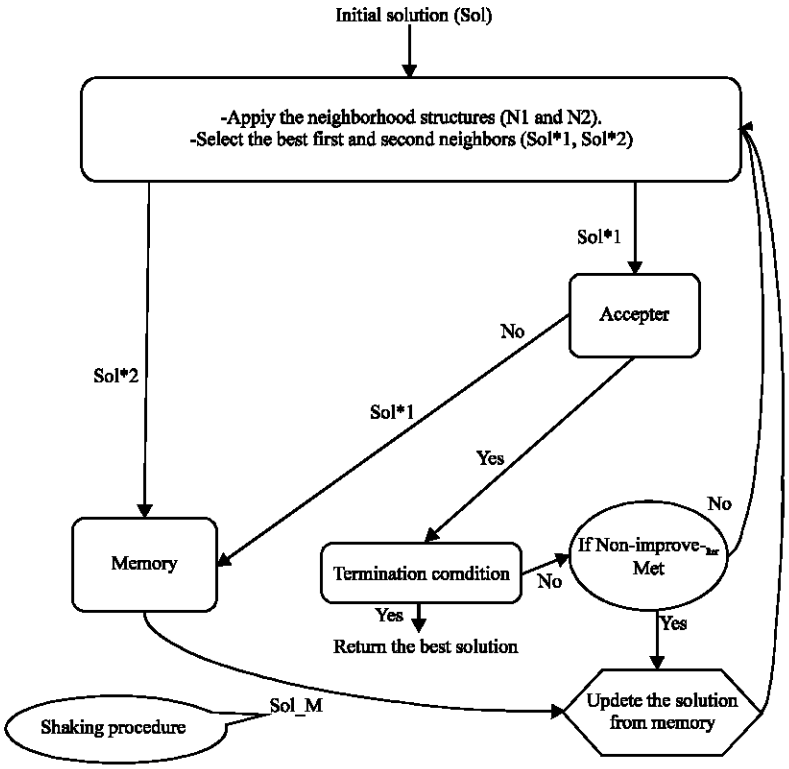

Fig.1: Proposed schematic representation of SA with memory

```
do while (termination criteria does not met)
        Generate k neighbors from N neighborhood structures and calculate the objective function for each neighbor;
        assign the best neighbor to Sol1 and the second best to Sol2;
        Add Sol2 to Mem;
        Calculate Δ: = f (Sol1)-f (Sol)
        If Δ≤0 then // improved solution
                Sol |Sol1;
                If f(Sol1)< [f (Sol]*)//Sol1 has better quality than Sol*
                Sol* | Sol1;
        End if
else
                Non_Improve_Iter = Non_Improve_Iter + 1
                Generate a random number called RAND between [0, 1];
                if ( e^(-Δ/T) >RAND then
                        Sol←Sol1;
                        Non_Improve_Iter = 0
else
Add Sol1 to Mem;
```

Fig. 2: A pseudo-code for SAM

- C_{iter} as the current iteration,
- T_0 as the initial temperature,
- T_C as the current temperature,
- $None_{improve_{Iter}}$ as the non-improvement consecutive iterations,
- Mem as the memory that save the not accepted solution with length μ.

The algorithm begins with a given initial solution. First, it generates n neighbors from N_i neighborhood structures. The best (Sol^1) and second (Sol^2) (based on the quality of solutions) are selected. The algorithm accepts (Sol^1) if its objective value is less than or equals the current solution (Sol), whilse (Sol^2) will be saved in the memory (if the memory (Mem) is full replace the worst solution in Mem with (Sol^2). In case (Sol^1) if (Sol) (i.e., the quality of Sol^1 is worse than Sol), the acceptance criteria ($e^{(-\Delta/T)}$) is applied. If $e^{\frac{-\Delta}{T}} >$ generated random number is between 0 and 1, the Sol^1 is accepted. Similarly, the memory Mem is updated with (Sol^2). If the algorithm does not accept (Sol^1), Mem will be updated by (Sol^1)(Fig. 2).

When the non-improvement consecutive iterations are met, one solution is randomly selected from the Mem and a shaking procedure is applied on it to produce Sol_m. This will divert the search to other solution space, by randomly swap the highest penalties lecture with other lectures (free clash), when the difference Δ between the solution Sol_m and the current solution Sol is less than $Cost_r$ (Eq. 1):

$$Cost_r = f(sol) * \frac{C_{Iter}}{Iter} * 0.1 \quad (1)$$

where, Iter is total iterations number, C_{Iter} is the current iterations, f (Sol) the current cost and $Cost_r$ is the cost range to limit the cost value of Sol_m. Lu and Hao (2010) presented strategy called penalty guided perturbation to improve the best solution if the search cannot improve it. They employed the perturbation operator to reconstruct the obtained local optimum by selecting a number of the highly-penalized lectures randomly and applies swap or move. The difference between the proposed shaking procedure and (Lu and Hao, 2010) is in the ranking penalties of soft constraint violation. Lu and Hao (2010) ranked the lectures in non-increasing order according to their total penalties of soft constraint violations (for all soft constraints penalties). Whilst, the proposed shaking procedure assign the lectures in non-increasing way according to each soft constraint penalty independently, so that almost all lectures with high penalty will be involved in this shaking procedure during the search space.

Neighborhood structure: One of the important features of local search algorithm is the definition of its neighborhood (Lu and Hao, 2010). This study use three neighborhood structures denoted by NS_1, NS_2 and NS_3. (Note: - all neighborhoods consider free clash):

NS_1: Move one lecture from the current period to another free position period.

NS_2: Randomly swaps two different lectures from different time slots and rooms.

NS_3: Normally the selection mechanism in neighborhood structure happens randomly (e.g:- select two lectures randomly which belongs to two different rooms and slots). Thus, the search may need more time to reach good solution. Hence, this study proposed neighborhood

Table 1: Soft constraint violation for each period and time slot

	Room 1					Room 2					Room 3					
DT	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5	Total penalty
1	0	5	6	30	0	100	0	1	4	2	10	20	0	1	160	339
2	6	15	10	0	0	2	4	2	30	100	1	0	60	1	03	234
3	10	0	80	10	0	1	20	1	1	1	3	0	0	0	0	127
4	10	1	4	5	2	0	10	0	10	20	6	4	150	1	10	233
5	1	2	8	10	0	20	30	15	0	1	2	2	6	0	9	106
6	2	3	6	1	8	120	30	40	0	4	0	0	9	0	1	224

Table 2: Average penalty costs and CPU time for NS_1, NS_2 and NS_3 neighborhood structures and their combinations over 35 independent runs

	N1		N2		N3		N1¬N2		N1¬N3		N2∪N3		N1¬N2∪N3	
Comp	Cost	Time (sec)	Cost	Time (sec)	Cost	Time (sec)	Cost	Time (sec)	Cost	Time (sec)	Cost	Time (sec)	Cost	Time (sec)
01	38	40.10	29	40.19	21	100.00	17	70.17	16	88.22	18	90.34	12	82.12
02	187	50.08	175	83.51	140	115.18	129	96.31	127	101.29	135	99.28	90	96.34
03	233	45.20	200	49.29	180	66.15	170	70.14	171	78.31	179	63.37	130	71.33
04	167	47.09	159	51.21	139	72.23	118	66.31	117	61.05	122	68.12	89	60.26
05	700	35.31	650	45.21	523	59.22	511	42.51	516	54.19	512	55.23	360	51.23
06	185	45.38	174	48.26	157	61.04	153	43.06	156	58.25	155	59.45	107	50.75
07	170	31.16	156	43.14	131	43.28	129	39.40	121	36.12	130	43.16	89	41.19
08	173	38.22	162	46.27	129	50.31	122	43.17	123	46.18	129	47.01	93	(43.89

structure to minimize the random selection. In NS_3, the total soft constraints violations for each timeslot and sum them up for all days in the week is calculated (Table 1) and the highest penalty timeslot is swapped with randomly selected timeslot.

Cooling schedule: The performance of SA depends heavily on the cooling schedule (Blum and Roli, 2003). This work used cooling scheme that was proposed by Lewis (2006) as follow:

$$\lambda_0 = 1\text{-}\beta$$

$$\lambda_{i} + 1 = \lambda_{1+\frac{\beta+\beta}{M}}$$

$$T_{i+1} = T_i\text{-}\lambda_{i+1}\left(\frac{T_0}{M}\right)$$

where, β represents a parameter to control a value for λ. The resulting value for λ is used to influence the amount of concavity or convexity present in the cooling schedule. M represents the number of steps taken by temperature T_0 to decrease to a value close to zero. In this experiment, the parameters are set based on the preliminary experiments, where T_0 = 1500 and β- -0.99 as in Lewis (2006) study.

Experiment and result: This study test on 3 datasets instances categories. The first 4 instances (test 1-4) were previously used for the old version of the Curriculum-based Course Timetabling (CB-CTT) in the literature (Di Gaspero and Schaerf, 2006). The second 7 instances (DDS1-DDS7) proposed by Bonutti *et al.* (2008) and the third 21 competition instances from The Second International Timetabling Competition ITC 2007 track 3 (Gaspero *et al.*, 2007). This work is programmed in vb.net 2010 on a PC Windows Vista platform with 2.10 GHz CPU and 4 GB RAM.

The experiment was first carried out to analyze the performance of the proposed neighborhood structure (NS_3). Table 2 shows the results of the experiments that were carried out on the three neighborhoods and their different combinations. For comparison purpose, this experiment applied Steepest Descent (SD) with for the first 8 competition instances (Abramson *et al.*, 1999). The average penalty cost and CPU time over 35 independent runs is reported in Table 2. From Table 2 one clearly finds that obtain much better quality solution. According to the results, starting from NS_1 will decrees the costs and minimizes the CPU time more than from NS_2. This result had encouraged this study to use this combination in the enhance SA.

$$NS_1|\ NS|\ NS$$

Firstly, to make a fair comparison, the basic SA applied with geometric cooling schedule (Abramson *et al.*,1999) and compare it with the SA with the adopted cooling schema proposed by Lewis (2006) in Table 3. The SA with the adopted cooling schema performed better than the geometric cooling schedule almost in all instances, whereas geometric cooling schedule performed better in Comp 5 and Comp 10 but ties with the adopted cooling schema.

Table 3: Comparable results between SAs (adopted cooling schema and geometric cooling schema) average penalties costs over 30 indepe ndent runs

	Adopted cooling schema	Geometric cooling schema
Comp01	6.7	7.1
Comp02	56.2	59.3
Comp03	98.3	99.0
Comp04	67.9	73.2
Comp05	340.6	339.4
Comp06	74.0	76.4
Comp07	32.8	33.0
Comp08	66.9	68.6
Comp09	121.5	123.0
Comp10	24.2	24.1
Comp11	0.4	0.8

Table 4: Comparison between different numbers of non-Improved iteration (average value of penalty costs over 25 runs)

	10 iterations		20 iterations		30 iterations		40 iterations		50 iterations	
$Non_{Improve_{Iter}}$										
Dataset	Ave	Best	Ave	Best	Ave	Best	Ave	Best	Ave	Best
Comp 01	11.31	10	9.56	8	7.10	6	5.0	5	5.0	5
Comp 11	9.91	7	6.37	4	5.01	4	0.00	0	0.00	0
Comp 05	352.47	345	332.05	325	309.67	302	300.53	293	305.49	294

Table 5: Computational statistics results of the SAM algorithm over 30 independent runs under the ITC 2007 competition stop conditions

Dataset	Best	Mean	Mediam	SD	Dataset	Best	Mean	Mediam	SD
Comp 01	5	5.0	5.0	0.0	Comp 19	62	70.7	71	5.9
Comp 02	35	42.2	41.5	5.7	Comp 20	14	17.8	17.5	2.9
Comp 03	77	81.3	80.0	3.5	Comp 21	81	87.8	88.5	3.7
Comp 04	43	50.7	51.5	3.6	Test1	226	231.5	231.0	4.6
Comp 05	293	301.3	301.0	4.9	Test2	19	23.6	24.0	3.6
Comp 06	51	56.7	55.5	4.6	Test3	72	81.0	82.0	5.6
Comp 07	15	21.1	21.5	4.1	Test4	80	88.4	89.5	5.0
Comp 08	46	52.4	51.0	4.2	DDS1	58	73.6	75.0	7.8
Comp 09	99	106.3	106.0	5.2	DDS2	0	0.0	0.0	0.0
Comp 10	6	11.1	11.5	3.6	DDS3	0	0.0	0.0	0.0
Comp 11	0	0.0	0.0	0.0	DDS4	29	34.7	34.0	3.8
Comp 12	307	314.1	313.5	5.2	DDS5	0	0.3	0.0	0.5
Comp 13	71	76.3	76.0	3.9	DDS6	0	0.6	0.5	0.7
Comp 14	55	64.6	64.5	6.6	DDS7	0	0.8	1.0	0.9
Comp 15	68	75.5	75.0	5.2	Toy	0	0.0	0.0	0.0
Comp 16	32	42.5	43.0	6.2					
Comp 17	61	71.3	71.0	6.8					
Comp 18	70	79.9	78.5	6.5					

Next, this study made preliminary experiments to find good number non improvement consecutive iterations, $Non_{Improve_{Iter}}$. $Non_{Improve_{Iter}}$ is the maximum number of consecutive iterations that the search is allowed to local optimum. When this number is met, the search will randomly select a solution from the memory (Mem). For this experiment, Comp 1 and Comp 11 are used to investigate this parameter. Table 4 showed that in 25 runs the best $Non_{Improve_{Iter}}$ is 40.

Results under ITC 2007 timeout condition: Table 5 illustrates the computational statistic of the proposed SAM algorithm based to the following performance indicators:- the best penalty cost (Best), the average penalty costs (Mean), median result (Median) and the standard deviation over 30 runs.

Table 6: Best results and comparison with the best known results under ITC 2007 timeout condition over 30 independent runs

	The proposed approach			Best known	
	Best	average	time (sec)	Results	Methods
Comp 01	**5**	5	160	**5**	Tabu search
Comp 02	35	42.2	450	**24**	SAT-modulo-theory
Comp 03	77	81.3	430	**66**	Local search
Comp 04	43	50.7	455	**35**	Local search
Comp 05	293	301.3	400	**290**	Simulated annealing
Comp 06	51	56.7	390	**27**	SAT-modulo-theory
Comp 07	15	21.1	400	**6**	SAT-modulo-theory
Comp 08	46	52.4	300	**37**	other
Comp 09	99	106.3	420	**96**	Tabu search
Comp 10	6	11.1	440	**4**	SAT-modulo-theory
Comp 11	**0**	0	120	**0**	Tabu search
Comp 12	307	314.1	450	**300**	Simulated annealing
Comp 13	71	76.3	424	**59**	Tabu search
Comp 14	55	64.6	399	**51**	Mathematicalprogrammirg
Comp 15	68	75.5	443	**66**	Tabu search
Comp 16	32	42.5	330	**18**	SAT-modulo-theory
Comp 17	61	71.3	350	**56**	SAT-modulo-theory
Comp 18	70	79.9	410	**62**	Hybrid method
Comp 19	62	70.7	461	**57**	Local search
Comp 20	14	17.8	423	**4**	SAT-modulo-theory
Comp 21	81	87.8	350	**75**	Simulated annealing
Test 1	226	231.5	460	**224**	Tabu search
Test 2	19	23.6	390	**16**	Tabu search
Test 3	72	81.0	350	**67**	Other
Test 4	80	88.4	390	**73**	Tabu search
DDS1	58	73.6	400	**48**	SAT-modulo-theory
DDS2	**0**	0	118	**0**	Tabu search
DDS3	**0**	0	126	**0**	Tabu search
DDS4	29	34.7	450	**17**	Hybrid methods
DDS5	**0**	0.3	400	**0**	Tabu search
DDS6	**0**	0.6	442	**0**	SAT-modulo-theory
DDS7	**0**	0.8	350	**0**	Tabu search
Toy	**0**	0	0.4	**0**	Simulated annealing

Best results are bold

Results in Table 5 showed that SAM standard deviation is quite small and the mean penalty results is small as well.

Table 6 shows the comparison results of SAM with best known results under ITC 2007 timeout condition (http://tabu.diegm.uniud.it/ct/). The first column indicates the instances. Column 2-4 indicate the penalty of best, average results and average runs time in seconds over 30 runs on each instance for the SAM. Column 5-6 shows the penalty cost of the best known results and the approaches that obtained the best known results.

As shown in Table 6, SAM obtained competitive results with all best known approaches. For instance Comp 1, Comp 11, DDS2, DDS3, DDS5, DDS6, DDS7 and Toy, the optimal solutions are found (under the ITC 2007 timeout condition). Therefore, this study can conclude that the SAM algorithm is generally able to produce high quality solutions when compared to the best known results.

In the next experimentation, the experiments evaluate the performance of SAM with other algorithms in literature (Lu and Hao, 2010; Cesco *et al*., 2008; Muller, 2008; Lach and Lubbecke, 2008; Geiger, 2008; Clark *et al*., 2008), (M2 to M6) consecutively (Table 6). Best solutions cost is written in bold.

CONCLUSION

This study presented a new hybridization between simulated annealing with non-accepted solutions memory. The main idea of the hybrid SAM is to enhance the ability of escaping from local optimum. SA could be trapped in local optimum especially when the temperature is very low. When the search trapped in local optimum for a number of consecutive non-improvement solutions, SAM randomly select one solution from the memory which was generated and not-accepted in some recently visited iterations, to be the current solution . Thus, the search can jump to other promising region and escape from local optimum. Whilst, the memory and the proposed shaking procedure leads the search to avoid recycling and trapping in the same local optimum.

In this work, the SAM algorithm is evaluated on curriculum-based course timetabling problem track 3 of the Second International Timetabling Competition. The computational results showed that the SAM algorithm competes very well with reference algorithms and the best known results in the literature. Thus, the proposed SAM enhanced the SA performance by enhancing the ability of escaping from local optimum, which might lead to a good promising region.

In future works, the next study plan to use tabu list mechanism to save both not accepted and not visited solutions in order to avoid recycling and will test it on in real world course timetable dataset from Universiti Kebangsaan Malaysia (Faculty of Engineering case study).

ACKNOWLEDGMENT

The authors wish to thank Ministry of Higher Education for supporting this work under the FRGS Research Grant Scheme (FRGS/1/2012/SG05/UKM/02/11).

REFERENCES

Abramson, A., H. Dang and M. Krisnamoorthy, 1999. Simulated annealing cooling schedules for the school timetabling problem. Asia-Pacific J. Oper. Res., 16: 1-22.

Alvarez-Valdds, R., F. Parredo and J.M. Tarnarit, 2002. A tabu search algorithm for assigning teachers to courses. Top, 10: 239-259.

Azizi, N., S. Zolfaghari and M. Liang, 2009. Hybrid simulated annealing with memory: An evolution-based diversification approach. Int. J. Prod. Res., 47: 1-26.

Beligiannis, G.N., C.N. Moschopoulos, G.P. Kaperonis and S.D. Likothanassia, 2008. Applying evolutionary computation to the school timetabling problem: The Greek case. Comput. Oper. Res., 35: 1265-1280.

Blum, C. and A. Roli, 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surveys, 35: 268-308.

Bonutti, A., F.D. Cesco, L.D. Gaspero and A. Schaerf, 2008. Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization and results. Ann. Oper. Res., 173: 1-12.

Cesco, F.D., L.D. Gaspero and A. Schaerf, 2008. Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation and results. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, August 18-22, 2008, Montreal, Canada pp: 1-11.

Clark, M., M. Henz and B. Love, 2008. A repair-based timetable solver. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, August 18-22, 2008, Montreal, Canada, pp: 1-18.

Di Gaspero, L. and A. Schaerf, 2006. Neighborhood portfolio approach for local search applied to timetabling problems. J. Math. Model. Algorith., 5: 65-89.

Eley, E., 2007. Ant algorithms for the exam timetabling problem. Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling, August 30-September 1, 2006, Brno, Czech Republic, pp: 364-382.

Elmohamed, M.A.S., P. Coddington and G. Fox, 1998. A Comparison of Annealing Techniques for Academic Course Scheduling. In: The Practice and Theory of Automated Timetabling II, Burke, E. and M. Carter (Eds.). Springer-Verlag, Berlin Heidelberg, pp: 92-112.

Gao, H., B. Feng, Y. Hou, B. Guo and L. Zhu, 2006. Adaptive SAGA based on mutative scale chaos optimization strategy. Inform. Technol. J., 5: 524-528.

Gaspero, L.D., B. McCollum and A. Schaerf, 2007. The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (Track 3). Technical Report. http://www.cs.qub.ac.uk/itc2007/curriculmcourse/report/curriculumtechreport.pdf.

Geiger, M.J., 2008. An application of the threshold accepting metaheuristic for curriculum based course timetabling. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, August 18-22, 2008, Montreal, Canada, pp: 1-12.

Jeon, Y.J. and J.C. Kim, 2004. Application of simulated annealing and tabu search for loss minimization in distributed systems. Int. J. Elect. Power Energy Syst., 26: 9-18.

Kolonko, M., 1999. Some new results on simulated annealing applied to the job shop scheduling problem. Eur. J. Operat. Res., 113: 123-136.

Lach, G. and M.E. Lubbecke, 2008. Curriculum based course timetabling: Optimal solutions to the udine benchmark instances. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, August 18-22, 2008, Montreal, Canada.

Lewis, R., 2006. Metaheuristics for university course timetabling. Ph.D. Thesis, School of Computing, Napier University, Edinburgh.

Lu, Z. and J.K. Hao, 2010. Adaptive tabu search for course timetabling. Eur. J. Oper. Res., 200: 235-244.

McCollum, B. and N. Ireland, 2006. University timetabling: Bridging the gap between research and practice. Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling, August 30-September 1, 2006, Brno, Czech Republic.

Muller, T., 2008. ITC 2007 solver description: A hybrid approach. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, August 18-22, 2008, Montreal, Canada.

Swarnkar, R. and M.K. Tiwari, 2004. Modeling machine loading problem of FMSs and its solution methodology using a hybrid tabu search and simulated annealing-based heuristic approach. Rob. Comp. Integr. Manuf., 20: 199-209.

Thompson, J. and K. Dowsland, 1995. General cooling schedules for a simulated annealing based timetabling system. Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling, August 29-September 1, 1995, Edinburgh, UK., pp: 345-363.

Ueda, H., D. Ouchi, K. Takahashi and T. Miyahara, 2004. Comparisons of genetic algorithms for timetabling problems. Syst. Comput. Japan, 35: 1-12.

Wren, A., 1996. Scheduling, timetabling and rostering-A special relationship? Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling, August 29-September 1, 1995, Edinburgh, UK., pp: 46-76.

Xinchao, Z., 2011. Simulated annealing algorithm with adaptive neighborhood. Applied Soft Comput., 11: 1827-1836.