Modern Applied Science; Vol. 12, No. 11; 2018
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Hybrid Simulated Annealing with Meta-Heuristic Methods to Solve UCT Problem

Issam AlHadid[1], Khalid Kaabneh[2] & Hassan Tarawneh[3]

[1] Faculty of Computer Information Systems, The University of Jordan, Aqaba, Jordan

[2] College of Computer Science and Informatics, Amman Arab University, Amman, Jordan

[3] College of Information Technology, Aqaba University of Technology, Aqaba, Jordan

Correspondence: Issam AlHadid, Faculty of Computer Information Systems, The University of Jordan, Aqaba, Jordan. Tel: 962-79-9282636. E-mail: issamfe@hotmail.com

Received: April 10, 2018 Accepted: September 20, 2018 Online Published: October 29, 2018

doi:10.5539/mas.v12n11p385 URL: https://doi.org/10.5539/mas.v12n11p385

Abstract

Simulated Annealing (SA) is a common meta-heuristic algorithm that has been widely used to solve complex optimization problems. This work proposes a hybrid SA with EMC to divert the search effectively to another promising region. Moreover, a Tabu list memory applied to avoid cycling. Experimental results showed that the solution quality has enhanced using SA-EMCQ by escaping the search space from local optimum to another promising region space. In addition, the results showed that our proposed technique has outperformed the standard SA and gave comparable results to other approaches in the literature when tested on ITC2007-Track3 university course timetabling datasets.

Keywords: Simulated Annealing, Tabu List, Meta-heuristic, EMC, Monte Carlo, University Course Timetabling (UCT)

1. Introduction

Simulated Annealing, also known as (SA) is a stochastic optimization technique inspired by Metropolis algorithm for statistical mechanics (Metropolis et al., 1953; Van Laarhoven & Aarts, 1987). Annealing is the cooling process of materials in a heat case or the molten crystalline solids. The target of this cooling process is the alignment of atoms in the most regular possible crystalline structure. Kirkpatrick et al., (1983) showed that the solids annealing model, which proposed by Metropolis et al. (1953) could be used for optimization problems, where the (minimizing or maximizing) the objective function corresponds to the metal status energy. SA starts once with an initial solution (Sol) generated randomly or using constructive techniques. At each iteration, SA generates a neighbor solution, Sol' of Sol. The Sol' is accepted if it improves the cost value f(Sol) of Sol (where Δ= f(Sol') – f(sol) ≤ 0); Otherwise Sol is accepted if, p = e-Δ/T > generated random number between 0 and 1. T is the current temperature. At the beginning of the search, T is high and decreased during the search following using a cooling scheme. SA stops when the T value is close to zero (frozen temperature), where there are no more moves to be accepted. As in many metaheuristics techniques, SA has the drawback of revisiting the same solution (recycling) and trapped in local optimum. This will lead to a longer time to find reasonable good solution. This problem will cost an extra computational time without any improvement. Therefore, using memory in SA is an effective way to overcome the problem of recycling as proposed in (Méndez-Díaz, Zabala, & Miranda-Bront, 2016). Moreover, the temperature reheating could divert the solutions to escape from local optimum when the temperature becomes very cold. Moreover, this paper presented enhanced exponential Monte Carlo algorithm with non-improvement counter, where the basic Monte Carlo (MC) method is defined by Woller (1996) as:

"Monte Carlo (MC) methods are stochastic techniques--meaning they are based on the use of random numbers and probability statistics to investigate problems".

Monte Carlo (MC) is the simplest approximate algorithm. Ayob & Kendall (2003) introduced three probabilistic approaches to accept the worse solution or non-improving moves.

1) Linear Monte Carlo (LMC) uses a linear probability of accepting to the worst solution.
2) Exponential Monte Carlo (EMC) has the same behaviour as LMC but it uses an exponential ratio.

3) Exponential Monte Carlo with counter (EMCQ) has extend the EMC acceptance criterion to accept the worse solution depending on the solution quality, and a consecutive non-improving iterations number as a counter to adaptively accept the worst solution.

Abdullah et al. (2005) presented a modified variable neighborhood search approach with EMC acceptance criterion for course timetabling problem (called VNS-EMC). The experimental results showed that the VNS-EMC is better than the standard VNS and it comparable with other approaches in the literature. Saber et al. (2009) hybridized the EMCQ with Tabu list. The main contribution of this hybrid approach is to save the moves during the search, in order to avoid cyclic move, by keeping the accepted move in a Tabu list for a certain number of iterations. Therefore, this work presents a hybrid solution using SA with EMC-counter and Tabu list memory to addresses the limitations of SA trapped in a local optimum by escaping from local optimum solution.

2. Related Work

2.1 SA Components

Generally, when applying SA algorithm, a number of common components must be set in order to obtain an effective SA algorithm, which are: (i) Initial temperature, (ii) Cooling schedule, (iii) Neighborhood structure.

2.1.1 Initial Temperature

The initial temperature leads SA to walk randomly over the landscape. Thus, high or infinity initial temperature would be the best choice (Triki et al., 2005). On other hand, the high or infinity initial temperature leads SA to take a longer time to cool down (Triki et al., 2005). However, Poupaert & Deville (2000) estimated the temperature during the search process and made the control parameter using the current acceptance probability. The value of the initial temperature is a fixed value selected according to the acceptance probability criteria during the search process.

Table 1. Summary of the presented initial temperature from the literature

Author	Mechanism	Disadvantages
Tarawneh et al. (2013a)	Dynamic mechanism to initialize the initial temperatures according to some solutions for each instance. Given the feasible initial solution, the SA starts several iterations and calculates the deviations average. Where the mechanism will decide the initial moderate temperature according to the SA acceptance criterion ratio.	The moderated temperature leads the solution to local optimum and minimize the worst accepted solutions.
Zhang et al. (2010)	Initialize the initial temperature at the first stage of the algorithm	The temperature is still high especially after the first part of the process, which consumed more computational time
Aycan & Ayav (2009	Select the initial temperature before the algorithm start, comparing the bad transition ratio for the first current solutions with a given value	the temperature is still high after the first part of the search process
Poupaert & Deville (2000)	Estimate the temperature during the search process	The temperature is not control parameter anymore but the acceptance probability
Kirkpatrick et al. (1983)	Starting with high initial temperature value, and then estimate it by comparing the accepted transitions ratio applied for several bad transition with a given value	The temperature is not a control parameter anymore and the SA could trap in local optimum very fast

According to the above literature, many researchers selected the initial temperature as their starting point in order to avoid consuming more computational time and to avoid quickly trapping in local optima. Poupaert & Deville (2000) initialized the temperature based on the initial acceptance ratio $\chi 0$, as described in equation 1.

$$\chi_0 = \chi\ (\{\delta_1, \delta_n, \delta_{n+1}, \dots, \delta_m\}, T_0)$$

$$\chi_0 = \frac{1}{m}\sum_{i=1}^{n} exp\ (-\delta_i / T_0) + \frac{m-n}{m} \quad (1)$$

Where χ_0 is the starting acceptance probability between (60% to 80%), T0 is the starting temperature; $\delta i = f(si) - f(s0)$, s0 is the initial solution, s_i is the new neighbour of s0, f(si) is the objective function for si, and m is the neighbors solutions space size. They repeat the above procedure until the acceptance ratio exceeds $\chi 0$ (i.e $\chi 0 >$

0.8).

2.1.2 Cooling Schedule

The Cooling schedule is the process of decreasing the temperature value in SA search process. Romeo and Sangiovanni-Vincentelli (1991) presented a procedure in order to design the cooling schedule in the following steps:

1) Start with an initial temperature T0, leading to a satisfactory approximation of the steady distribution DT0.
2) Reduce T0 by small increment α(T) such that DT0 is a good starting point in order to approximate DT0 - α(T).
3) Repeat the above process until no more improvement.

The cooling schedule has two categories:

4) Static schedule specifies at the beginning (before the algorithm starts). Example of static cooling schedule is geometric Cooling Schedule (GCS) which proposed by Kirkpatrick et al. (1983), is the most popular cooling schedule, in term of the simplicity to obtain. In GCS, the temperature is reduced as in equation (2).

$$T_{i+1}=\alpha\ . \tag{2}$$

Where ti+1 is temperature after i+1 iteration; α (0< α <1) is the cooling rate or factor.

5) Adaptive schedule is the process of adjusting the temperature decrement during the algorithm process in reference to the information obtained (e.g. use the objective values for each level to decide the decrement amount). Namely, the amount of temperature decrement in the next iteration obtained based on the run history.

The purpose of this adaptive cooling schedule is to maintain the search solutions to be close with each other. Another adaptive cooling schedule presented by Lewis et al. (2007) (Eq. 3 – 5).

$$\lambda_0 = 1-\beta \tag{3}$$

$$\lambda_{i+1} = \lambda_i + \frac{\beta+\beta}{M} \tag{4}$$

$$T_{i+1} = T_i - \lambda_{i+1}(\frac{T_0}{M}) \tag{5}$$

Where β represents the parameter that helps to determine the value for λ in each step. λ is used to influence the amount of concavity or convexity present in the cooling schedule; M is the number of decreasing steps that takes the temperature T0 to be close to zero Tmin. Lewis et al. (2007) used this cooling schedule to get the initial feasible solution and set M is equal 100; β = - 0.99.

Table 2. Summary of the presented cooling schedules and investigations from the literature

Author	Mechanism	Advantages	Disadvantages
Lewis et al. (2007)	They presented formulas to adjust the temperature decrement amount during the search process, in order to generate the feasible solution	It adjust the temperature decrement amount during the search process properly	It take to much computational time if the value of the initial temperature is high, therefore, the initial temperature need to be selected carefully (not high)
Azizi & Zolfahg (2004)	Adaptive cooling schedule that update or manage the temperature dynamically based on a number of consecutive improving moves	Adjust the temperature based on the search path profile	It need to tune the consecutive improving iterations carefully
Triki et al. (2005)	Discussed and investigated different cooling schedules. Furthermore, they formulated a new practical annealing schedule for a given objective function	They showed that the presented logarithmic cooling schedule better than any the others to ensure the SA convergence	Need to compare the obtained cooling schedules with other adaptive, in order to make a fair comparisons

Kirkpatrick et al.1983)	Proposed the geometric cooling schedule Ti+1=α . Ti , where α is the cooling rate between zero and one, and Tk is the current temperature value	Most popular cooling schedule, in term of the simplicity to be applied	Static cooling schedule, must be specified before the search starts and it need to be carefully tuned

Based on the literature reviews above, it has been concluded that the adaptive cooling schedule have more strengths than the static cooling schedule. However, we will apply adaptive cooling schedule as proposed in the Lewis et al. (2007).

2.2 Introduction to Timetable

Wren (1995) considered timetabling as a unique case of scheduling problems and describes it as follows:

"*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".*

Another definition of the timetable presented by Burke et al., (2004) as follows*:"a timetabling problem is a problem with four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of meetings; and C, a finite set of constraints. The problem is to assign time and resources to the meetings so as to satisfy the constraints as far as possible".*

Many timetabling problems have been researched such as - railway timetabling (Climent et al., 2008; Cacchiani, & Toth, 2012), sports timetabling (Aggoun & Vazacopoulos, 2004), university timetabling (Tarawneh & Ayob, 2011; Burke et al., 2013; Tarawneh & Ayob, 2013; Tarawneh, et al., 2013b; Babaei et al., 2015) and school timetabling (Veenstra et al., 2016).

Building the academic timetable is a typical real world-scheduling problem and became a very challenging work in every academic institution each year. In educational filed, there are three deferent timetabling problems: school timetable, university course timetable and university examination timetable. The three categories may have the same basic characteristic but still have some differences. For example, the class size (student's enrolment) for all school courses usually are very similar and the same group of students is associated with a set of courses, whilst the university courses different in the class size.

2.2.1 University Course Timetabling Problem

"*The university course timetabling problems (UCT) is the task of allocating courses and lecturers to the rooms and timeslots, in such a way as to satisfy the hard constraints and minimize the soft constraints"* (McCollum & Ireland, 2006).

There are two commons rules, that the timetable should follows, which are:

1) Hard constraints, where this constraint needs to be satisfied e.g. two or more courses should schedules at the different room and time-period.
2) Soft constraints, where this constraint needs to be satisfied if possible e.g. female lecturers is preferred to teach at the morning timeslots.

Generally, the timetable quality is measured or calculated by the satisfaction degree of the soft constraints, whilst, for the hard constraints, it is measured by the complete fulfillment degree. That is, it is measured by penalized each violation of soft constraints (penalties cost). Small penalty values indicate good quality solution. This evaluationfunction is called the objective function or penalties cost function. "*The problem of university course timetable consists of a set courses* $C = \{c_1, c_2,...,c_n\}$ *to be scheduled in a set of p periods,* $T = \{ t_1, t_2, ..., t_p \}$ *and a set of m rooms* $R = \{r_1,r_2, ... ,r_m\}$*. For each course* (c_i) *there are* l_i *lectures should be scheduled. Each courses group that share common students called curriculum* Cr_k*,* $CR = \{Cr_1, Cr_2, ... , Cr_s\}$*"*(Lü and Hao 2010).

2.2.2 Constructive Heuristic for University Course Timetabling Problem

In order to generate the initial solution, the lectures placed in the periods and rooms without any hard constraints violations. Generally, the heuristic begins with an empty timetable and list of unscheduled events U. The procedure starts by taking the events one by one form U and place them into a feasible places in a timetable. The procedure is continued until U= Ɵ.

Burke et al. (2007) presented common constructive timetabling heuristics with free conflict:

1) Largest Degree First (LDF): Schedule the courses with the greatest number of conflicting courses first.
2) Largest Colour Degree First (LCDF): Schedule the courses with the largest number of conflicting courses

that already been scheduled. In order to break ties.

3) Least Saturation Degree First (LSDF): Schedule the course with the least number of valid periods currently available.

Arntzen & Løkketangen (2005) designed a mechanism called a sequential assignment of events to places, as follow:

1) Initialize list L that contains all unassigned events E.
2) Select E with fewest possible places from L.
   - If there is no unique *E* that has a fewest possible places,
   - Then select *E* randomly among the events with fewest possible places.
3) Find a place P for E.
4) Insert E to the chosen place P. Then, update the information of the possible places and fitting events. Furthermore, remove E from L. Then return to step II if L is not empty. Otherwise, generate a feasible solution.
5) Finally, if the above process failed to generate a feasible solution when some event is unassigned, then restarts the process again with new different random seed.

The next section discusses our proposed method to optimize the University Course Timetable (UTC) problem using simulated annealing and Tabu search with temperature reheating function to divert the solution when the algorithm trapped on local optimum solution.

2.3 Problem Statement and Discussion

Based on literature reviewed above, we found that each algorithm has strengths and weakness in terms of the search ability and the solution quality. Tabu Search (TS) performed effectively when the neighborhood structure is small and the landscape is fat, by escaping from the local optimum.

For the SA algorithm, many researchers tried to overcome it weakness, such as the longer computational time to reach a good quality solutions, and to solve the problem of trapping in the local optimum at the end of the search space. For example, using Variable Neighborhood Search (VNS) technique with SA will enhance the SA performance, by improving the search ability and solution quality, a hybrid SA with TS to avoid revisiting solutions and it will guide the SA diversification part in order to explore more of the search space. The main aim is to capitalize on both SA and TS strengths.

Based on the recommendations from the literature, we have been motivated to enhance the SA performance with the following challenges in mind:

1) How to avoid trapping in local optimum when the initial temperature reaches the very cold one?
2) How to escape from local optimum if the search solution trapped in a local optimum and divert the solution to other good promising region?

3. Proposed Algorithm

Figure (1 & 2) shows our proposed algorithm. The algorithm begins with a given initial solution (*Sol*) and generates *n* neighbours from the neighbourhood structure. The best solution (Sol^l) selected based on the quality of solutions. The algorithm checks the solution movements using the Tabu list, then the solution (Sol^l) is accepted if its quality is better than or equal (objective value is less than or equal) to the current solution (*Sol*). Otherwise, (i.e. the quality of (Sol^l) is worse than *Sol*), the acceptance criteria $p(X) = e^{-\Delta f / T_i}$ is applied or $p(X) = e^{-\Delta f / Tanh(Non\text{-}improving\ iterations}$. If $p(X)>$ generated random number between 0 and 1, the Sol^l is accepted.

Given an initial solution Sol; set the total iteration number Iter;

Set the non-improvement iterations*Non_improve*;

Set the Tabu list memory μ;

Set the size of the non-improvement iteration ω;

Do while (termination criteria does not met)

Generate *k* neighbors from *N* neighborhood structures;

Update the μ *// update the Tabu list*

Assign the best neighbor to Sol^l

Set the initial temperature T_0 using EQ 2.

Calculate $\Delta = f(Sol^1) - f(Sol)$

If $\Delta \leq 0$ then // Improved solution.

$Sol \leftarrow Sol^1$ // Update the current solution

If $f(Sol^1) < f(Sol^*)$ // Sol^1 has better quality than Sol^*.

$Sol^* \leftarrow Sol^1$ // Update the best solution so far.

End if

If $\Delta = 0$ then

$Non\_improve = Non\_improve + 1$ //Update the non-improvement iterations.

End if

Else

$Non\_improve = Non\_improve + 1$ //Update the non-improvement iterations.

Generate a random number η between [0, 1];

If ($e^{-\Delta f / Ti} > \eta$ or $e^{-\Delta f /} Tanh(Non\_improve)$ then // The solution Sol^1 is accepted

$Sol \leftarrow So$

Update μ // Add the solution movement Tabu list memory

End if

End if

Update the temperature; //Adaptive cooling schedule (see Eq 3)

End while;

Output the final solution;

Figure 1. A pseudo-code for SA-EMCQ.

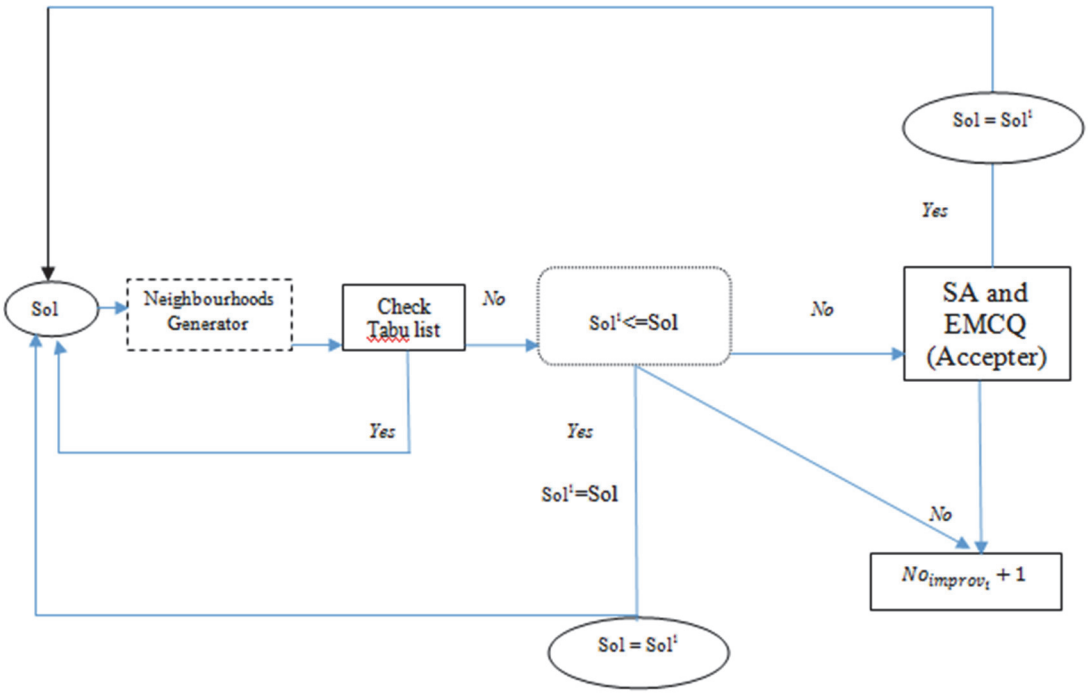

Figure 2. SA-EMCQ

4. Experimental and Results

This section presents our experiment results to test the performances of our EMCQ in comparison to others as

describes in the literature review. The initial parameters were set in our work to:

1) Tabu list: 7
2) Temperature: dynamic (see equation 2.)
3) Cooling schedule: adaptive (see equations 3-5)
4) Neighbourhood structures: Move and Swap.
5) Termination condition: time = 460 seconds.

4.1 Comparing the SA-EMCQ and SA on ITC-2007 Track3 Dataset

In Table 3, we have compared SA-EMCQ with standard SA under the ITC2007 with track3 timeout condition of 460 seconds in our machine, in order to analyze the search capability for the proposed method against SA. For this comparison, we have used the standard SA as described in (Van Laarhoven & Aarts, 1987).

Table 3. Results of SA-EMCQ and SA over 31 runs on ITC2007 track3 datasets

	(SA-EMCQ)				(SA)			
	Best	Worst	Mean	STD	Best	Worst	Mean	STD
Comp01	5	5	5.00	.000	5	7	5.28	.581
Comp02	44	68	53.12	6.371	47	74	59.34	8.170
Comp03	83	103	93.65	6.831	93	123	106.78	9.540
Comp04	52	73	60.12	5.645	62	84	74.47	7.427
Comp05	301	319	310.18	4.908	315	340	323.97	6.597
Comp06	65	89	75.12	6.089	70	90	78.94	6.185
Comp07	24	55	33.03	7.880	26	52	35.47	7.858
Comp08	48	69	56.47	6.964	60	91	75.06	9.308
Comp09	108	127	115.76	5.737	120	155	134.06	10.336
Comp10	18	34	25.62	4.899	23	56	34.81	9.972
Comp11	0	0	.00	.000	0	3	.69	.896
Comp12	321	342	333.44	6.061	332	359	342.38	8.304
Comp13	71	90	80.50	5.848	92	121	104.72	8.113
Comp14	60	82	69.29	6.987	72	96	81.59	6.380
Comp15	69	89	77.00	5.598	82	113	97.87	8.206
Comp16	32	51	41.09	5.288	39	66	48.62	6.445
Comp17	62	82	69.85	5.533	77	102	88.19	7.399
Comp18	70	89	79.03	5.931	90	113	100.94	7.020
Comp19	71	92	80.71	6.417	96	119	108.06	6.801
Comp20	14	26	19.29	3.555	32	56	42.56	7.242
Comp21	85	99	91.62	4.452	99	130	110.25	10.737

Table 3, shows that EMCQ outperformed the standard SA across all instances, for all (best, worst, mean, and the standard deviation). Table 4 presents the t-test and P-Value of SAEMCQ against Standard SA. Results tested using Wilcoxon Signed-Rank Test in order to support our Alternative Hypothesis stated above.

Table 4. Statistical analysis for SA-EMCQ against Standard SA on ITC 2007 track3 datasets

Instances	Asymp. Sig. (2-tailed) P-value
comp01	.014
comp02	.002
comp03	.000
comp04	.000
comp05	.000
comp06	.003
comp07	.308
comp08	.000
comp09	.000
comp10	.001
comp11	.000
comp12	.000

comp13	.000
comp14	.000
comp15	.000
comp16	.000
comp17	.000
comp18	.000
comp19	.000
comp20	.000
comp21	.000

Table (4) shows clearly that there are statistically significant differences between SA-EMCQ performance and the standard SA where SA-EMCQ improved the solution quality almost in every case. Note: the significance level in this test is equal 0.05, meaning, if the P-Value is less than 0.05 then we have statistically significant results, which achieved in our work for the majority-tested instances. In addition, the significance value in table (4) evidently supports the alternative hypothesis that SA-EMCQ will enhance the SA performance (the solution quality) (which means that the alternative hypothesis is accepted).

Table 5. Average and best results of SA-EMCQ in comparison with the top five competitors and best-known results from ITC 2007 Track3. (Http: //Tabu.diegm.uniud.it/ct /)

	SA-EMCQ		T1		T2		T3		T4		T5		Best Known*
Rank			5		6		3		4		1		-
	best	mean	best	mean	best	mean	best	mean	best	mean	best	mean	Best*
comp01	5	5.00	5	6.7	10	27	5	5.0	5	5.1	5	5.0	5
comp02	44	53.12	55	142.7	111	131.1	55	61.2	50	65.6	51	61.3	24
comp03	83	93.65	71	160.3	119	138.4	71	84.5	82	89.1	84	94.8	66
comp04	52	60.12	43	82	72	90.2	43	46.9	35	39.2	37	42.8	35
comp05	301	310.18	309	525.4	426	811.5	309	326	312	334.5	330	343.5	290
comp06	65	75.12	53	110.8	130	149.3	53	69.4	69	74.1	48	56.8	27
comp07	24	33.03	28	76.6	110	153.4	28	41.5	42	49.8	20	33.9	6
comp08	48	56.47	49	81.7	83	96.5	49	52.6	40	46	41	46.5	37
comp09	108	115.76	105	164.1	139	148.9	105	116.5	110	113.3	109	113.1	96
comp10	18	25.62	21	81.3	85	101.3	21	34.8	27	36.9	16	21.3	4
comp11	0	.00	0	0.3	3	5.7	0	0.0	0	0.0	0	0.0	0
comp12	321	333.44	343	485.1	408	445.3	343	360.1	351	361.6	333	351.6	300
comp13	71	80.50	73	110.4	113	122.9	73	79.2	68	76.1	66	73.9	59
comp14	60	69.29	57	99	84	105.9	57	65.9	59	62.3	59	61.8	51
comp15	69	77.00	71	160.3	119	138	71	84.5	82	89.1	84	94.8	66
comp16	32	41.09	39	92.6	84	107.3	39	49.1	40	50.2	34	41.2	18
comp17	62	69.85	91	143.4	152	166.6	91	100.7	102	107.3	83	86.6	56
comp18	70	79.03	69	129.4	110	126.8	69	80.7	68	73.3	83	91.7	62
comp19	71	80.71	65	132.8	111	125.4	65	69.5	75	79.6	62	68.8	57
comp20	14	19.29	47	97.5	144	179.3	47	60.9	61	65	27	34.3	4
comp21	85	91.62	106	185.3	169	185.8	106	124.7	123	138.1	103	108	75

Note:

- Mmis the average quality solutions results; b is the best solution.
- T5: Hybrid approach (Müller, 2009).
- T3: TS (Lau & Zhao, 2010).
- T4: Atsuta et al. (2008).
- T2: Threshold accepting local search (Geiger, 2008).
- T1: Repair-based Local Search (Clark et al., 2008).
- Best*: Best Known results under ITC2007 track3 timeout condition (http: //Tabu.diegm.uniud.it/ct /).

Table 5 showed that the hybrid SA-EMCQ outperformed some approaches that tested in ITC 2007 Track3, under the same time out condition. In addition, SA-EMCQ outperformed threshold accepting local search presented by Geiger (2008) and also outperformed repair-based Local Search presented by Clark et al. (2008) (for all

instances). Moreover, the SA-EMCQ approach also gave results better than TS in (Lau & Zhao, 2008) and the Hybrid approach in Müller (2009) (i.e. comp 1, 2, 5, 9, 12, 15, 16, 20 and 21) and comparable with other instances. Furthermore, our proposed algorithm gave best-known results in comparison to all techniques for instances Comp1 and Comp11. According to the average results in table 5, SA-EMCQ ranked second place among the others.

5. Conclusion and Future Work

This paper focused on reviewing several heuristics and meta-heuristics algorithms that implemented in the literature to solve the combinatorial optimization problems, especially university course timetabling problem. Mainly, we have focused on SA algorithm and identifying its strengths and weakness such as consuming more computational time and trapped in local optimum, especially at the end of the search process when the temperature closed to zero. Thus, this revision motivated us to improve SA features and components using the most effective hybrid approach between SA with other EMCQ algorithms.

Test results showed that our proposed algorithm gave best-known results in comparison to all techniques for some instances and the average results of SA-EMCQ ranked second place among the others. Furthermore, we can gladly conclude that our proposed algorithm is generally able to produce good and comparable solutions when compared to the best known and some other approaches in the literature. As a future work, this proposed algorithm can be modified using another metaheuristics algorithms such as genetic algorithm and practical fish swarm, moreover this hybrid algorithm will be applied on real word case study.

References

Abdullah, S., Burke, E. K., & Mccollum, B. (2005). An investigation of variable neighbourhood search for university course timetabling. Paper presented at the *The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA)*.

Aggoun, A., & Vazacopoulos, A. (2004). Solving sports scheduling and timetabling problems with constraint programming. In *Economics, Management and Optimization in Sports* (pp. 243-264). Springer, Berlin, Heidelberg.

Arntzen, H., & Løkketangen, A. (2005). A tabu search heuristic for a university timetabling problem. In *Metaheuristics: Progress as Real Problem Solvers* (pp. 65-85). Springer, Boston, MA.

Atsuta, M., Nonobe, K., & Ibaraki, T. (2008). ITC-2007 Track2: An approach using general CSP solver.

Aycan, E., & Ayav, T. (2009). Solving the course scheduling problem using simulated annealing. Paper presented at the *Advance Computing Conference, 2009. IACC 2009. IEEE International*.

Ayob, M., & Kendall, G. (2003). A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. Paper presented at the *Proceedings of the International Conference on Intelligent Technologies, InTech*.

Azizi, N., & Zolfaghari, S. (2004). Adaptive temperature control for simulated annealing: a comparative study. *Computers & Operations Research, 31*(14), 2439-2451.

Babaei, H., Karimpour, J., & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, *86*, 43-59.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, *64*(12), 1695-1724.

Burke, E. K., Kendall, G., Mısır, M., Özcan, E., Burke, E., Kendall, G., … Mısır, M. (2004). *Applications to timetabling.* Paper presented at the Handbook of Graph Theory, chapter 5.6.

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research, 176*(1), 177-192.

Cacchiani, V., & Toth, P. (2012). Nominal and robust train timetabling problems. *European Journal of Operational Research*, *219*(3), 727-737.

Clark, M., Henz, M., & Love, B. (2008). QuikFix—A Repair-based Timetable Solver. Paper presented at the *Proceedings of the Seventh International Conference on the Practice and Theory of Automated Timetabling*. Retrieved from http://www. comp. nus. edu. sg/~ henz/publications/ps/PATAT2008. pdf

Climent, L., Barber, F., Salido, M., & Ingolotti, L. (2008). Robustness and capacity in scheduling: Application to railway timetabling. In *Workshop on Planning, Scheduling and Constraint Satisfaction* (pp. 87-96).

Geiger, M. J. (2008). Randomised variable neighbourhood search for multi objective optimisation. *arXiv preprint arXiv:0809.0271*.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simmulated annealing. *Science, 220*(4598), 671-680.

Lau, H. Y., & Zhao, Y. (2008). Integrated scheduling of handling equipment at automated container terminals. *International Journal of Production Economics*, *112*(2), 665-682.

Lewis, R., Paechter, B., & Rossi-Doria, O. (2007). Metaheuristics for university course timetabling. *Evolutionary Scheduling* (pp. 237-272): Springer.

McCollum, B., & Ireland, N. (2006). University timetabling: Bridging the gap between research and practice. *E Burke, HR, ed.: PATAT*, 15-35.

Méndez-Díaz, I., Zabala, P., & Miranda-Bront, J. J. (2016). An ILP based heuristic for a generalization of the post-enrollment course timetabling problem. *Computers & Operations Research, 76*, 195-207.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*(6), 1087-1092.

Müller, T. (2009). ITC2007 solver description: A hybrid approach. *Annals of Operations Research, 172*(1), 429-446.

Poupaert, E., & Deville, Y. (2000). Simulated Annealing with estimated temperature. *AI Communications, 13*(1), 19-26.

Romeo, F., & Sangiovanni-Vincentelli, A. (1991). A theoretical framework for simulated annealing. *Algorithmica, 6*(1-6), 302-345.

Sabar, N. R., Ayob, M., & Kendall, G. (2009, April). Tabu exponential Monte-Carlo with counter heuristic for examination timetabling. In *Computational Intelligence in Scheduling, 2009. CI-Sched'09. IEEE Symposium on* (pp. 90-94). IEEE.

Tarawneh, H. Y., & Ayob, M. (2011). Using Tabu search with multi-neighborhood structures to solve University Course Timetable UKM case study (faculty of engineering). In *Data Mining and Optimization (DMO), 2011 3rd Conference on* (pp. 208-212). IEEE.

Tarawneh, H. Y., Ayob, M., & Ahmad, Z. (2013a). Simulated annealing with dynamic initial temperatures for university course timetable problem. *Journal of Engineering and Applied Sciences, 8*(2), 58-63.

Tarawneh, H., & Ayob, M. (2013). Adaptive neighbourhoods structure selection mechanism in simulated annealing for solving university course timetabling problems. *Journal of Applied Sciences, 13*(7), 1087.

Tarawneh, H., Ayob, M., & Ahmad, Z. (2013b). A hybrid simulated annealing with solutions memory for curriculum-based course timetabling problem. *Journal of Applied Sciences, 13*(2), 262.

Triki, E., Collette, Y., & Siarry, P. (2005). A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *European Journal of Operational Research, 166*(1), 77-92.

Van Laarhoven, P. J., & Aarts, E. H. (1987). *Simulated Annealing: Theory and Applications* (pp. 7-15): Springer.

Veenstra, M., & Vis, I. F. (2016). School timetabling problem under disturbances. *Computers & Industrial Engineering, 95*, 175-186.

Woller, J. (1996). The basics of Monte Carlo simulations. *Physical Chemistry Lab, Spring: University of Nebraska–Lincoln*.

Wren, A. (1995). Scheduling, timetabling and rostering—a special relationship?. Paper presented at the *International Conference on the Practice and Theory of Automated Timetabling*.

Zhang, R., & Wu, C. (2010). A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing*, *10*(1), 79-89.