khazanah informatika
Jurnal Ilmu Komputer dan Informatika

Automated Course Timetabling Optimization Using Tabu-Simulated Annealing Hyper-Heuristics Algorithm

Ahmad Muklason[1*], Ahsanul Marom[1], I Gusti Agung Premananda[1]
[1]Department of Information Systems,
Institut Teknologi Sepuluh Nopember (ITS),
Surabaya, East Java, Indonesia
*mukhlason@is.its.ac.id

Abstract-The topic of solving Timetabling Problems is an interesting area of study. These problems are commonly encountered in many institutions, particularly in the educational sector, including universities. One of the challenges faced by universities is the Course Timetabling Problem, which needs to be addressed regularly in every semester, taking into consideration the available resources. Solving this problem requires a significant amount of time and resources to create the optimal schedule that adheres to the predefined constraints, including both hard and soft constraints. As a problem of computational complexity, University Course Timetabling is NP-hard, meaning that there are no exact conventional algorithms that can solve it in polynomial time. Several methods and algorithms have been proposed to optimize course timetabling in order to achieve the optimal results. In this study, a new hybrid algorithm based on Hyper-Heuristics is developed to solve the course timetabling problem using the Socha Dataset. This algorithm combines the strengths of Simulated Annealing and Tabu Search to balance the exploitation and exploration phases and streamline the search process. The results show that the developed algorithm is competitive, ranking second out of ten previous algorithms, and finding the best solution in six datasets.

Keywords: Course Timetabling Problem, Tabu Search Algorithm, Simulated Annealing Algorithm, Hyper-Heuristics

Article info: *submitted March 30, 2023, revised September 18, 2023, accepted October 16, 2023*

1. Introduction

The Timetabling problem is a problem to efficiently allocate time and resources towards meetings with the aim of minimizing constraint violations [1], [2]. This issue is widespread across multiple fields, however, it receives particular attention in the area of course timetabling. This specific challenge involves scheduling academic courses for lecturers, time slots, and classrooms in a manner that enhances the quality of education [3]. Despite its importance, manually resolving the timetabling problem is time-consuming and often yields suboptimal results. As a result, current research efforts have shifted towards automating the solution to this problem, particularly in the context of large-scale case studies [4].

The Timetabling problem is regarded as NP-hard, making it difficult for exact algorithms to solve it in polynomial time [5]. As a result, non-deterministic algorithms, such as metaheuristic and hyper-heuristic algorithms, have been developed to generate solutions that are close to the global optimum in polynomial time [6]. In response to this problem, the present study has proposed a new hybrid hyper-heuristic algorithm. This hybrid approach combines the benefits of two algorithms, and the use of a hyper-heuristic is motivated by the generalization advantage it provides, eliminating the need for parameter tuning for each dataset [7].

The hybridization was developed through the integration of two algorithms: Simulated Annealing and Tabu Search. The Simulated Annealing algorithm, a metaheuristic that simulates the cooling process of heated steel, has the advantage of escaping local optima through its diversification process and accepting worst solutions [8]–[11]. On the other hand, Tabu Search is a meta-heuristic algorithm that uses memory objects to achieve both economic exploitation and exploration in the search space. The tabu list is used to prevent the search from revisiting previously visited solutions by adding the recently visited solutions to the list [12]. The main advantage of the Tabu Search algorithm is its implementation of the tabu list, which helps the search move away from previously visited areas and perform more extensive exploration in the search space [13]. By combining these two hybridized algorithms, it is expected that an optimal solution for the automated course timetabling problem can be obtained. The hybridization was performed due to several previous studies that showed that hybrid algorithms produce more optimum solutions.

The Socha dataset was utilized as the test dataset in this study. It is a popular dataset among researchers and has become a benchmark for evaluating the performance of developed algorithms [14]–[16]. This dataset encompasses a range of course timetabling problems, from small to large in size.

The structure of this paper is as follows: Section 2 provides an overview of the related literature and research that supports the study. Section 3 explains the implementation process of the Tabu-Simulated Annealing Hyper-Heuristics Algorithm for the Socha dataset. The results and analysis of the implementation of the Tabu-Simulated Annealing based Hyper-Heuristics Algorithm are presented in Section 4. Section 5 compares the results obtained from the Tabu-Simulated Annealing Hyper-Heuristics Algorithm with the benchmark solution from previous studies. In the final section, 6, the conclusion and future prospects of this research are discussed.

2. Related Works

a. Timetabling

The Timetabling problem is a combinatorial optimization problem that involves scheduling a set of events with specific characteristics onto limited resources while satisfying predefined constraints [17]. This problem is prevalent in various domains, such as transportation, sports, health and education [3]. Due to its computational complexity, the Timetabling problem is considered to be NP-hard, meaning that conventional algorithms cannot solve the problem in polynomial time [4], [5], [18].

b. Socha Dataset

Socha dataset is a dataset that introduced by Kryzysztof Socha and developed by Ben Paechter [19]. Socha dataset consists of 11 instances, which are divided into 5 small instances, 5 medium instances, and 1 large instance. Table 1 show detail socha dataset. Each instance has various time limits. The time limit for the small instance is 90 seconds. Meanwhile, for the medium instance has a time limit of 900 seconds and for large instance is 9000 seconds. This time limit has been determined by Socha's research [20].

The available timeslot for Socha dataset is 45 timeslots with 9 timeslots in 5 days per week. The number of events is the sum of all available course. The number of features is a facility used for each scheduled course. The number of students is the sum of student in a semester. Meanwhile, the number of rooms is the available rooms in a semester [20].

Table 1. Statistic of Socha Dataset

Characteristic	Small	Medium	Large
Event	100	400	400
Rooms	5	10	10
Features	5	5	10
Student	80	200	400
Approx. features per rooms	3	3	5
Percent feature use	70	80	90
Max events per student	20	20	20
Max student per event	20	50	100

c. Constraint of Socha Dataset

The hard constraints of Socha dataset are [16]:

1) No Student can be assigned more than one course at the same time.
2) The rooms must satisfy the features required by course, including enough for all student taking course in that room.
2) No more than course is allowed at a timeslot in each room.
4) Only one course is allowed in each room at a time.

The soft constraints of Socha dataset are [16]:

1) Student should not have a single course on a day.
2) Student should not have more than two courses in a row on a day.
3) Student should not have a course scheduled in the last timeslot of a day.

d. Hyper-Heuristics

Hyper-heuristics is an approach to develop more general non-deterministic algorithms. This approach has four types: (1) exploration of heuristics combination for solution perturbation, (2) exploration of heuristics combination for solution construction, (3) generating heuristics for solution perturbation, and (4) generating heuristics for solution construction [21]. Structurally, hyper-heuristics are divided into three main components: (1) move acceptance to decide whether the new solution result is used in the next iteration or not, (2) heuristic selection to choose some heuristics used to modify the solution, and (3) a set of heuristics [22].

3. Methods

a. Generate Initial Solution

Initial Solution is a solution that is used as an initial schedule in optimization. This initial solution contains the initial timeslot and rooms before the optimization is run. Greedy Algorithm is used to form the initial solution, where the first order in a list of subjects is placed in the first available slot, so that all courses are scheduled.

b. Implementation of Tabu-Simulated Annealing Algorithm

Tabu-Simulated Annealing based Hyper-heuristics algorithm is implemented after the initial solution is generated. The first step of implementation is making low level heuristics. This research uses two types of low-level heuristics, there are “swap” and “move”. “Swap” is the low-level heuristic that exchanging timeslot for two or more selected timeslot. While the “move” is moving the one or more selected timeslot to the random timeslot. The implementation of the algorithm starts with the implementation of the Simulated Annealing algorithm. Simulated Annealing algorithm will be implemented on local search by using acceptance criteria, if the iteration produces a better solution than the previous solution, the new solution will be accepted as the current solution, so that the initial solution changes with a better solution. If the result of the iteration produces worse solution than the previous solution, the annealing process is calculated. Annealing process is conducted by Boltzmann equation. Figure 1 show detail Simulated Annealing algorithm.

Algorithm 1 Simulated Annealing

current = initial solution
t = intial temperature
l = intial length
i=0
while $i < l$ **do**
  $candidate \in N(current)$
  if $candidate \leq f(current)$ **then**
    current=candidate
  else[$exp(-(f(current) - f(candidate))/t > random[0, 1]$]
    current=candidate
  end if
  update i and t
end while

Figure 1. Simulated Annealing Algorithm

Tabu Search algorithm is implemented when the random value does not pass in the Boltzmann equation. Tabu Search will be implemented to check whether the solution is in the tabu list or not. If the solution is not in the tabu list, the new solution will be accepted as a current solution and entered that solution structure into tabu list. The solution cannot be accepted in next iteration until the solution exit from tabu list. Figure 2 show detail Tabu Search algorithm.

In this research, Simulated Annealing and Tabu Search algorithm are hybridized making the new approach, Tabu-Simulated Annealing Algorithm. The hybridization of algorithms is shown in Figure 3.

Algorithm 2 Tabu Search

Tabu list T
current = initial solution
best=current
while !stop condition **do**
  $current = argmin_{x \in N(current)} f(x), x : non - tabu$
  if $current < best$ **then**
    best=current
  end if
  record the recent move in T
  delete the oldest entry if necessary
end while

Figure 2. Tabu Search Algorithm

Algorithm 3 Tabu-Simulated Annealing

Tabu list T
current = initial solution
t = intial temperature
l = intial length
i=0
while $i < l$ **do**
  $candidate \in N(current)$
  if $candidate \leq f(current)$ **then**
    current=candidate
  end if
  if $exp(-(f(current) - f(candidate))/t > random[0, 1]$ **then**
    current=candidate
  else
    record the recent move in T
    delete the oldest entry if necessary
  end if
  update i and t
end while

Figure 3. Tabu - Simulated Annealing Algorithm

c. Developing of Tabu-Simulated Annealing Algorithm

1) Reheating

Reheating is the process of increasing the temperature of each iteration. The increasing temperature is carried out when the temperature of iteration reached at the determined temperature. If the number of iterations has reached a multiple of reheating iterations, the temperature will be increased by the temperature of the reheating.

2) Tabu Low-Level Heuristics

The concept of tabu low level heuristics is like tabu list of Tabu Search algorithm concept. If low level heuristics does not give the better result than current solution, the low-level heuristics that have been choose will enter the tabu low level heuristics. Low level heuristics that have entered the tabu low level heuristics cannot be used until low level heuristics exit from tabu low level heuristics list.

3) Roulette Wheel

The roulette wheel is performed on low level heuristics. If a low-level heuristic produces a better solution, this low-level heuristic score will be added, for example +10. Otherwise, if a low-level heuristic produces a value that is no better, the score of the low-level heuristics will be reduced, for example -5. The score of all low-level heuristics will be counted in probability. So, the probability of selected low-level heuristics always changes depend on the low-level heuristics’ performance.

d. Experiment of the Parameters

Tabu-Simulated Annealing algorithm has many parameters that influence the algorithm performance and the penalty result. This research use 10 parameters as that is used to experiment to get the optimum solution. The list of parameters is explained on Table 2.

Table 2. List of Parameters

Parameters	Meaning
LLH	The number of low-level heuristics that used
T0	Initial temperature of Simulated Annealing algorithm
T1	Final temperature of Simulated Annealing algorithm
Alpha	Decreasing temperature coefficient of Simulated Annealing
N Alpha	The number of iterations for each decreasing temperature
Beta	Increasing temperature coefficient of reheating process
N Beta	The number of iterations for each increasing temperature
TL	The length of tabu list of Tabu Search algorithm
TLLH	The length of tabu list of Tabu Search algorithm
RW	The method of selecting low level heuristics based on roulette wheel process
Parameters	Meaning
LLH	The number of low-level heuristics that used
T0	Initial temperature of Simulated Annealing algorithm
T1	Final temperature of Simulated Annealing algorithm
Alpha	Decreasing temperature coefficient of Simulated Annealing

4. Results

The optimization results are determined through several experiments by changing the parameter values. Each experiment is conducted to determine a set of parameters that produced the smallest penalty score for optimization, called optimum solution. In this research, researchers found two sets of parameters that produced optimum solution. The set of parameters that produce the optimum solution is explained in Table 3. The comparison of the experiment results is shown by the boxplot diagram in Figure 4, 5, and 6. Based on the Boxplot diagram, the best optimum solution is Experiment-N.

Table 3. List of Parameters

Parameters	Experiment-K	Experiment-N
LLH	2	2
T0	95	95
T1	0	0
Alpha	0,999	0,999
N Alpha	50	50
Beta	0,5	0,5
N Beta	25000	25000
TL	3	3
Parameters	Experiment-K	Experiment-N
TLLH	0	0
RW	-	Random Probability
Parameters	Experiment-K	Experiment-N

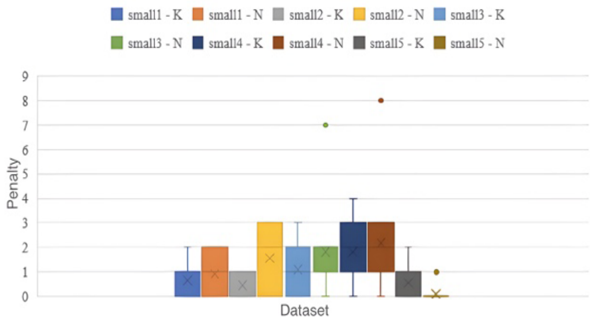

Figure 4. Boxplot Diagram for Small Instance

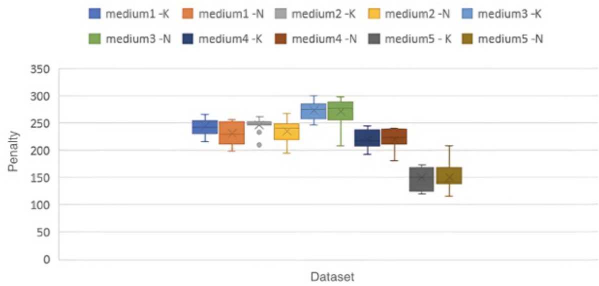

Figure 5. Boxplot Diagram for Medium Instance

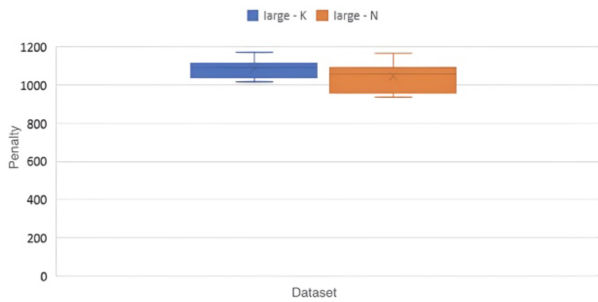

Figure 6. Boxplot Diagram for Large Instance

The automated optimization program using Experiment-N parameters runs 11 times for each both experiment and instance. For each run, the experiment uses the time limit according to the rules of Socha dataset. The time limit for each small instance is 90 seconds, 900 seconds for each medium instance, and 9000 seconds for the large instance. shows the penalty score results in optimum parameter. Table 4 describes the performance of Tabu-Simulated Annealing Hyper-Heuristics algorithm.

Table 4. The Performance of Tabu-Simulated Annealing Hyper-Heuristics

Instance	Average Initial	Best	Worst
small1	315,3	0	2
small2	327,7	0	3
small3	297,6	0	7
small4	164,5	0	8
small5	454,6	0	1
medium1	1028,6	198	256
medium2	1045,8	195	268
medium3	1071,2	208	299
medium4	1169,6	181	242
medium5	1169,5	116	209
Large	1960	936	1169

5. Discussion

The results of this research were compared with previous studies, as demonstrated in Table 5.

Table 5. List of Benchmark Solution

Code	Algorithm
MBO	Migrating Bird Optimization [23]
FMH	Fuzzy Multiple Heuristics [24]
MA	Memetic Algorithm [25]
RII	Randomised Iterative Improvement [26]
TVNS	Tabu - Variable Neighbourhood Search [27]
GC	Graph Coloring [28]
TS	Tabu Search [29]
MSLS	MultiSwap Algorithm with Local Search [30]
MMAS	Max-Min Ant Systems [19]

The results of this research were compared to previous studies and are presented in Table 6. The Tabu-Simulated Annealing Hyper-Heuristics algorithm performed best for the small instance, with a penalty score of 0. For medium1, the algorithm was ranked 4th out of 10 compared algorithms. In the case of medium2, the algorithm was ranked 6th, while for medium3, it was ranked 2nd, only behind the MultiSwap Algorithm with Local Search. The algorithm was ranked 5th for medium4 and produced the best solution for medium5 compared to other benchmark solutions. The produced the best value of 936 and the algorithm was ranked 5th out of 10 compared algorithms. Overall, the developed algorithm was ranked 2nd among the 10 algorithms compared.

Table 6. The Comparison of the Penalty Score Result

Instance	TSA		MBO	FMH	MA	RII	TVNS	GC	TS	MSLS	MMAS
	Best	Average	Best	Best	Best	Best	Best	Best	Best	Average	
small1	0	0.9	25	10	0	0	0	6	1	2	1
small2	0	1.5	22	9	0	0	0	7	2	4	3
small3	0	1.8	19	7	0	0	0	3	0	2	1
small4	0	2.2	14	17	0	0	0	3	1	2	1
small5	0	0.1	17	7	0	0	0	4	0	0	0
medium1	198	230.9	394	243	221	242	317	372	146	174	195
medium2	195	235.5	378	325	147	161	313	419	173	184	184
medium3	208	271.3	305	249	246	265	357	359	267	188	248
medium4	181	219.7	282	285	165	181	247	348	169	180	164,5
medium5	116	151.2	276	132	130	151	292	171	303	132	219,5
large	936	1048	1015	1138	529	757	932	1068	1166	994	851,5

6. Conclusion

The research aimed at developing a hybrid algorithm to tackle the course timetabling problem. The algorithm was created by combining the strengths of both simulated annealing and tabu search algorithms. The results showed that the developed hybrid algorithm had a promising performance. It ranked second among the 10 algorithms developed in previous studies and produced the best results for 6 out of the 11 datasets tested. The study had limitations, particularly in the utilization of low-level heuristics, therefore future research could focus on enhancing the exploration of low-level heuristics.

References

[1] J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar, 'A survey of the state-of-the-art of

optimisation methodologies in school timetabling problems', Expert System with Application, vol. 165, p. 113943, Mar. 2021, doi: 10.1016/j.eswa.2020.113943.

[2] R. A. Oude Vrielink, E. A. Jansen, E. W. Hans, and J. van Hillegersberg, 'Practices in timetabling in higher education institutions: a systematic review', Annals of Operation Research, vol. 275, no. 1, pp. 145–160, Apr. 2019, doi: 10.1007/s10479-017-2688-8.

[3] M. M. Tavakoli, H. Shirouyehzad, F. H. Lotfi, and S. E. Najafi, 'Proposing a novel heuristic algorithm for university course timetabling problem with the quality of courses rendered approach; a case study', Alexandria Engineering Journal, vol. 59, no. 5, pp. 3355–3367, Oct. 2020, doi: 10.1016/j.aej.2020.05.004.

[4] T. Thepphakorn and P. Pongcharoen, 'Performance improvement strategies on Cuckoo Search algorithms for solving the university course timetabling problem', Expert System with Application, vol. 161, p. 113732, Dec. 2020, doi: 10.1016/j.eswa.2020.113732.

[5] I. G. A. Premananda, A. Tjahyanto, and A. Muklason, 'Hybrid Whale Optimization Algorithm for Solving Timetabling Problems of ITC 2019', in 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), IEEE, Jun. 2022, pp. 317–322. doi: 10.1109/CyberneticsCom55287.2022.9865647.

[6] M. H. Cruz-Rosales et al., 'Metaheuristic with Cooperative Processes for the University Course Timetabling Problem', Applied Sciences (Switzerland), vol. 12, no. 2, 2022, doi: 10.3390/app12020542.

[7] N. Pillay and E. Özcan, 'Automated generation of constructive ordering heuristics for educational timetabling', Annals of Operation Research, vol. 275, no. 1, pp. 181–208, Apr. 2019, doi: 10.1007/s10479-017-2625-x.

[8] X. Pan, L. Xue, Y. Lu, and N. Sun, 'Hybrid particle swarm optimization with simulated annealing', Multimedia Tools and Applications, vol. 78, no. 21, 2019, doi: 10.1007/s11042-018-6602-4.

[9] M. Abdel-Basset, W. Ding, and D. El-Shahat, 'A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection', Artificial Intelligence Review, vol. 54, no. 1, 2021, doi: 10.1007/s10462-020-09860-3.

[10] H. Lv, X. Chen, and X. Zeng, 'Optimization of micromixer with Cantor fractal baffle based on simulated annealing algorithm', Chaos, Solitons and Fractals, vol. 148, 2021, doi: 10.1016/j.chaos.2021.111048.

[11] X. Hao, J. Liu, Y. Zhang, and G. Sanga, 'Mathematical model and simulated annealing algorithm for Chinese high school timetabling problems under the new curriculum innovation', Frontiers of Computer Science, vol. 15, no. 1, p. 151309, Feb. 2021, doi: 10.1007/s11704-020-9102-4.

[12] Y. Alotaibi, 'A New Meta-Heuristics Data Clustering Algorithm Based on Tabu Search and Adaptive Search Memory', Symmetry (Basel), vol. 14, no. 3, 2022, doi: 10.3390/sym14030623.

[13] M. Chen, X. Tang, T. Song, C. Wu, S. Liu, and X. Peng, 'A Tabu search algorithm with controlled randomization for constructing feasible university course timetables', Computers & Operations Research, vol. 123, p. 105007, Nov. 2020, doi: 10.1016/j.cor.2020.105007.

[14] C. W. Fong, H. Asmuni, and B. McCollum, 'A Hybrid Swarm-Based Approach to University Timetabling', IEEE Transactions on Evolutionary Computation, vol. 19, no. 6, pp. 870–884, Dec. 2015, doi: 10.1109/TEVC.2015.2411741.

[15] R. A. Aziz, M. Ayob, Z. Othman, Z. Ahmad, and N. R. Sabar, 'An adaptive guided variable neighborhood search based on honey-bee mating optimization algorithm for the course timetabling problem', Soft Computing, vol. 21, no. 22, pp. 6755–6765, Nov. 2017, doi: 10.1007/s00500-016-2225-8.

[16] S. L. Goh, G. Kendall, N. R. Sabar, and S. Abdullah, 'An effective hybrid local search approach for the post enrolment course timetabling problem', American Journal of Open Research: OPSearch, vol. 57, no. 4, pp. 1131–1163, Dec. 2020, doi: 10.1007/s12597-020-00444-x.

[17] T. Song, M. Chen, Y. Xu, D. Wang, X. Song, and X. Tang, 'Competition-guided multi-neighborhood local search algorithm for the university course timetabling problem', Applied Soft Computing, vol. 110, p. 107624, Oct. 2021, doi: 10.1016/j.asoc.2021.107624.

[18] R. Esmaeilbeigi, V. Mak-Hau, J. Yearwood, and V. Nguyen, 'The multiphase course timetabling problem', European Journal of Operational Research, vol. 300, no. 3, 2022, doi: 10.1016/j.ejor.2021.10.014.

[19] K. Socha, J. Knowles, and M. Sampels, 'A MAX-MIN ant system for the university course timetabling problem', in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2002. doi: 10.1007/3-540-45724-0_1.

[20] S. L. Goh, G. Kendall, and N. R. Sabar, 'Simulated annealing with improved reheating and learning

for the post enrolment course timetabling problem', Journal of the Operational Research Society, vol. 70, no. 6, pp. 873–888, Jun. 2019, doi: 10.1080/01605682.2018.1468862.

[21] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, 'A classification of hyper-heuristic approaches: Revisited', in International Series in Operations Research and Management Science, 2019. doi: 10.1007/978-3-319-91086-4_14.

[22] S. S. Choong, L. P. Wong, and C. P. Lim, 'Automatic design of hyper-heuristic based on reinforcement learning', Information Science (N Y), vol. 436–437, 2018, doi: 10.1016/j.ins.2018.01.005.

[23] L. W. Shen, H. Asmuni, and F. C. Weng, 'A modified migrating bird optimization for university course timetabling problem', Jurnal Teknologi (Sciences and Engineering), vol. 72, no. 1, 2015, doi: 10.11113/jt.v72.2949.

[24] H. Asmuni, E. K. Burke, and J. M. Garibaldi, 'Fuzzy multiple heuristic ordering for course timetabling', in Proceedings of the 2005 UK Workshop on Computational Intelligence, UKCI 2005, 2005.

[25] S. Abdullah, 'Heuristic approaches for university timetabling problem', University of Nottingham, 2006.

[26] S. Abdullah, E. Burke, and Barry McCollum, 'A randomised iterative improvement algorithm with composite neighbourhood structures for university course timetabling', in The 6th Metaheuristic International Conference, 2005.

[27] S. Abdullah, E. K. Burke, and B. Mccollum, 'An investigation of variable neighbourhood search for university course timetabling', in Proceedings of the 2nd Multi-disciplinary International Conference on Scheduling: Theory and Applications (MISTA), 2005.

[28] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, 'A graph-based hyper-heuristic for educational timetabling problems', Eur J Oper Res, vol. 176, no. 1, 2007, doi: 10.1016/j.ejor.2005.08.012.

[29] E. K. Burke, G. Kendall, and E. Soubeiga, 'A Tabu-Search Hyperheuristic for Timetabling and Rostering', Journal of Heuristics, vol. 9, no. 6, 2003, doi: 10.1023/B:HEUR.0000012446.94732.b6.

[30] M. A. Al-Betar, A. T. Khader, and O. Muslih, 'A multiswap algorithm for the university course timetabling problem', in 2012 International Conference on Computer and Information Science, ICCIS 2012 - A Conference of World Engineering, Science and Technology Congress, ESTCON 2012 - Conference Proceedings, 2012. doi: 10.1109/ICCISci.2012.6297258.