2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)
April 09-11. 2021, ISTTS Surabaya, Indonesia

2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT) | 978-1-6654-0514-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/EIConCIT50028.2021.9431892

Automation and Optimization of Course Timetabling Using the Iterated Local Search Hyper-Heuristic Algorithm with the Problem Domain from the 2019 International Timetabling Competition

1st Umar Rizki Kusumo Widayu
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
umarrizki14@gmail.com

2nd Ahmad Mukhlason
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indoensia
mukhlason@is.its.ac.id

3rd Ika Nurkasanah
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
ika.nurkasanah@its.ac.id

***Abstract*— University Courses timetabling is scientifically known as a nondeterministic polynomial time (NP)-hard problem and is still an exciting topic to study due to the difficulty to find an exact algorithm that can solve the problem in polynomial time. Prior studies in the scientific literature have recognized the importance of automation and optimization of course timetabling problems, especially for the university level that require a fast and effective method to timetable thousands of courses at the beginning of the academic period. The complexity of this problem has attracted the interest of competition, namely the International Timetabling Competition (ITC) 2019, to raise ideas of algorithms to solve the problem. This study investigates the performance of Iterated Local Search-Hill Climbing (ILS-HC) and Iterated Local Search-Simulated Annealing (ILS-SA), Algorithms within hyper-heurism in solving university course timetabling problem of ITC 2019 problem domain and datasets. Tested over tiny and small datasets of ITC 2019 problems, the experimental results show that ILS-SA outperforms ILS-HC for both datasets. Specifically, over tiny dataset, ILS-SA could minimize the objective function to 6 compared to 37 as result of ILS-HC algorithm. While over small datasets ILS-SA outperforms ILS-HC by 776 compared to 1034.**

Keywords*— Course Timetabling Problem, Iterated Local Search, Hyper heuristic, International Timetabling Competition 2019

I. Introduction

Effective timetabling plays a pivotal role to many stakeholders involved in higher educational teaching and learning activities, both lecturers who teach and students who take these courses [1], [2], [3], [4]. Nonetheless, there are many obstacles when formulating the optimal timetabling solution since it is a complicated combinatorial problem. Recent evidence reveals that course timetabling is a type of non-deterministic-polynomial-time-hard (NP-hard) problem, a problem that is difficult to solve for large sizes [5]. Therefore, it has been attracting a lot of interest, as such the committee of International Timetabling Competition (ITC) was held. ITC has been held four times, which are ITC 2002, ITC 2007, ITC 2011, and ITC 2019. The first competition, i.e. ITC2002 focused on a simplified version of timetabling courses problems at universities. Then, ITC 2007 introduced three tracks, namely curriculum-based timetabling, post-enrolment timetabling, and examination timetabling. The ITC 2011 focused on secondary school timetabling problems.

The ITC 2019 adopted in this study focused on complex course timetabling problems. There are various methods in the literature applied to solve university course timetabling problem, including Iterated Local Search Algorithm which has been proven to be effective in finding optimal solution for timetabling, taking account a broader range of candidate solutions through iterative process. For example, in [6], [7] iterated local search was employed to solve university course timetabling problem domain different from ITC 2019 problem domain. Therefore, the performance of iterated local search over ITC 2019 problem domain is interesting to be investigated. The main convolutions of this study are two folds. Practically, we develop an application that could generate course timetable automatically as opposed to generate timetable manually that is tedious work. Scientifically, this is the first study on investigating iterated local search within hyper-heuristic framework to solve university course timetabling problem of ITC 2019 datasets.

II. Literature Review

Timetabling is categorized as a combinatorial problem because it has a wide choice of alternative solutions, and there are many optimal locales. Every timetabling problem has constraints divided into two, hard constraints and soft constraints. The hard one has conditions that must be met and should not be violated. Meanwhile, soft constraint is a limitation that does not have to be met, but if it is completed, it will generate a better solution. However, soft constraints usually have a penalty if they are not met. Hence, a solution to the timetabling problem can be considered to be feasible if it meets the hard constraints and is made to meet the soft constraints [8].

The ITC 2019 is a competition on international-based timetabling problems. ITC has been held four times in 2002, 2007, 2011, and 2019 consecutively. Timetabling problems that have been provided by ITC 2019 include some constraints such as rooms, classes, courses, distribution, and students. In this problem, all costs are represented as non-negative integer penalties, where the smaller the penalty, the better the solution [5]. In addition to constraints, the competition defines some decision variables such as course allocation in the classroom and existing timeslots and students' distribution into classroom according to the courses taken. The objective function of the problem is to minimize costs, represented by penalties, to generate a more optimal solution. There are two types of constraints here, hard constraints indicated by *required* and soft constraints represented by penalty. The detail of those constraints is expressed in Table 1.

Within this study hyper-heuristic framework was employed. Hyper-heuristic represents a methodology that automatically selects or generates heuristics to solve difficult computational search problems [7]. It also produces a series of low-level metaheuristics to solve complicated searches and optimization problems [9]. The hyper-heuristic approach aims to: (1) select and combine simpler heuristics (heuristic selection), or (2) produce new heuristics from existing heuristics components (heuristic generation) [10]. The hyper-heuristic strategy does not directly touch the solution space because there is a problem domain barrier and it only intersects with low-level heuristics. In contrast to hyper-heuristic, there is another methodology called metaheuristics that work on search space of solution space.

Iterated Local Search (ILS) refers to a metaheuristic that adapts the heuristic improvement to an iterative process and produces a solution chain [11]. Within this study, ILS was employed within hyper-heuristic framework instead of meta-heuristics. ILS was employed in this study because it has been successful in various domains [9]. ILS attempts to improve stochastic Multi-Restart Search by sampling across a broader range of candidate solutions and using Local Search techniques to refine more optimal solutions. In addition, ILS explores the sequence of solutions created as perturbations of the best current solutions, so the results are refined using embedded heuristics [12]. The following Fig. 1 describes the pseudocode of the ILS algorithm.

The ILS algorithm starts with forming the initial solution and will continue to search for a better solution until it meets the desired criteria [13]. The Local Search and Perturbation contained in the ILS, uses the Hyper heuristic framework. The low-level heuristic used is the move and swap methods. Previous research has established that Perturbation's candidate solutions in ILS-Hyper heuristic often result in better quality and faster time [14], [15], [16], [6], [17]. Another study revealed that ILS-Hyper heuristic generates competitive solution and better performance than other methodologies, such as Genetic Algorithm Hybrid, Iterative Forward Search, Adaptive Tabu Search, SAT, MaxSAT, Lower Bound with ILP, and Iterative Tabu Search with the ILP solver.

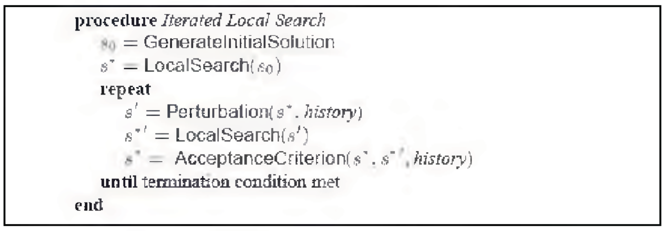
Similar prior works of this study include [18] that employ Self Adaptive and Simulated Annealing Hyper-Heuristics to solve University Course Timetable of ITC 2007 Problem domain. In addition, in [19] the same problem was solved using taboo-variable neighbourhood search based hyper-heuristic algorithm. However, since the course timetabling problem of ITC 2019 is relatively new, there are few prior works solving ITC 2019 datasets reported in the literature. To best of our knowledge, the only work found in the literature is [20] that solve ITC 2019 problem with Max SAT method.

III. Methodology

The dataset from ITC 2019 are divided into five subsets: tiny, small1, small2, medium, and large. However, this research is limited to test data on the small. Each data has several attributes, such as courses, classes, rooms, distribution constraints, and students. See [8] to consult the detailed explanation. The proposed method within this study consists of two phases: initial solution construction and optimization.

TABLE I. ITC 2019 Constraints

Dataset	Number of Areas	Number of Cities	Data Type
Dataset 1	10	10	Artificial
Dataset 2	10	15	Artificial
Dataset 3	13	38	Artificial
Dataset 4	40	99	Artificial
Dataset 5	46	138	Artificial
Dataset 6	96	192	Artificial
Dataset 7	150	300	Artificial
Dataset 8	200	300	Artificial
Dataset 9	250	250	Artificial
Dataset 10	300	300	Artificial
Dataset 11	150	200	Real
Dataset 12	200	250	Real
Dataset 13	250	275	Real
Dataset 14	300	300	Real



```
procedure Iterated Local Search
    $s_0$ = GenerateInitialSolution
    $s^*$ = LocalSearch($s_0$)
    repeat
        $s'$ = Perturbation($s^*$, history)
        $s^{*'}$ = LocalSearch($s'$)
        $s^*$ = AcceptanceCriterion($s^*$, $s^{*'}$, history)
    until termination condition met
end
```

Fig. 1. Pseudocode of Iterated Local Search

A. *Initial Solution Construction Phase*

This process occurs prior optimization phase. Within this phase, Greedy algorithm was employed to generate initial solution, in which the first order course in a list of courses is placed in the first available slot. When forming this initial solution, hard constraints are verified without considering the values of the soft constraints. If all courses have been scheduled and meet the hard constraints, the initial solution will be considered.

B. *Optimization Phase*

After the initial solution was generated, the solution is optimized in this phase. The iterated local search algorithm within hyper-heuristic framework was employed in this phase. Within hyper-heuristic framework, there are two strategies should be tuned, i.e. low-level heuristics selection and move acceptance.

The low-level heuristic (LLH) used in this research is *move*, where the timeslot of one course or several courses is moved into another selected timeslot randomly. *Move* has several variations such as moveRM to move the room used by the class, moveTS to move the timeslot used by the class, and moveRMTS to move the room and time of the class. During the iteration the LLH will be chosen randomly.

For move acceptance strategy we use iterated local search algorithm. Two local search algorithms i.e. hill climbing, and simulated annealing are evaluated. See Fig. 2 for the local search algorithm. In each iteration, when an LLH was chosen and result in new solution, the hard constraints are checked out and the penalty for the new solution is calculated. If the hard constraint does not met or has a greater penalty then the solution will be returned to the way it was before running LLH and it will be repeated until it finds a better solution or the limit of the loop has been reached.

```
int randLH = rand.nextInt(6);
        switch (randLH){
            case 0:
                clsLH = moveRMTS();
                break;
            case 1:
                clsLH = moveRMTS2();
                break;
            case 2:
                clsLH = move1RM();
                break;
            case 3:
                clsLH = move2RM();
                break;
            case 4:
                clsLH = move1TS();
                break;
            case 5:
                clsLH = move2TS();
                break;
            default:
                break;
        }
```

Fig. 2. Pseudocode for LLH Local Search

The next step within optimization phase is perturbation. In perturbation or what can be called a disturbance, the thing that should be done is almost the same as the Local Search process. A nuisance was administered using the existing LLH at the beginning of each iteration. The solution with the lower penalty value is accepted and replaces the previous ones. On the contrary, if the solution has a greater penalty value or is not feasible, then it will be returned to the last solution that has been accepted. The last step within optimization phase is final penalty calculation. The final penalty is the total of the penalty value for the timeslot used, the space penalty value used, and the penalty for soft constraints. The final penalty value determines the most optimum solution in several experiments carried out.

IV. Results & Discussions

The results of this study could be presented into two parta, i.e. the initial solution construction results and the optimization results as discussed below.

A. *Initial Solution Construction Result*

Some of the indicators used to sort the course indexes are the number of available times, the number of available rooms, the number of hard constraints per class, and the number of sorted class. In addition, there are several scenarios that are applied to produce an initial solution. Firstly, there is no combination of indicators, so that the order of objects is only based on 1 related indicator. Secondly, the best results from the first scenario are used for the second scenario, involving 2 indicators related to rooms and timeslots (obtained from scenario 1). Thirdly, the scenario involves three combinations of indicators, where the Rooms-Times and Times-HC indicators are in the first and second arrangements. The results of the initial solutions are shown in Table II.

B. *Optimization Phase Result*

Iterated local search algorithm with two local search algortihms: hill climbing (ILS-HC) and Simulated Annealing (ILS-SA). The parameters of the algorithms after trial and error are shown in Table III.

TABLE II. Result of Initial Solution

Instance	Avg Initial Solution
Test-Tiny	48
Test-Small 2	1790

TABLE III. Parameters of Optimization Scenarios

Parameters	Description
LLH	*Low level heuristics* used
Max Iterasi LS	Maximum iteration of the Local Search Algorithm
Iterasi ILS	Number of iterations used in the ILS
To (khusus ILS SA)	Initial temperature of the Simulated Annealing algorithm
Alpha (khusus ILS SA)	Temperature drop

1) ILS-HC: The first scenario experiments ILS Iteration parameters with a range of 10 to 10000, and there is no value change for the Max ILS and LLH iterations. The output of 1st scenario shows the impact of the variables from the ILS iteration, and it was chosen from ILS Iteration number 1500. Since it still had not reached the optimal solution or had not been exposed to local optima, the ILS iteration variants' impact can be checked.

Subsequently, the 2nd scenario aims to know the impact of the ILS iteration variable and the Max Iteration of the LS with a range of 2 to 100. The ILS and LLH iterations do not change in value and the result shows that iteration 7 produces the smallest penalty value for both the tiny and small dataset. Therefore, this iteration is used for the next scenario. The 3rd scenario is then observing LLH parameters using five out of six existing LLH. For ILS and Max Iteration parameters, the values do not change. As a result, from ten runs experiement, the results is shown by Table IV.
The fourth scenario investigates LLH parameters by using a combination of 4 out of 6 existing LLH. The ILS and LS Iteration parameters do not change in value and the ILS iteration used is 7th. One result from ten runs experiments of the combination of four LLH can be seen in Table V. The fifth scenario experiments on LLH parameters using a combination of 3 out of 6 existing LLH. Hence, there are 15 combinations where The ILS and LS Iteration parameters do not change in value. The ILS iteration used is the 7^{th,} and one of the combined results of ten runs experiment can be seen in Table VI.

TABLE IV. An Example of five LLH Combinations

ILS Iteration	1500	
Max. Iter. LS	7	
LLH	acdef	
Experiment	TINY	SMALL
Average	39.4	1087
Worst	42	1212
Best	37	972

TABLE V. An Example of Four LLH Combinations

ILS Iteration	1500	
Max. Iter. Ls	7	
Llh	Bcdf	
Experiment	Tiny	Small
Average	24.5	1069.8
Worst	42	1180
Best	13	954

TABLE VI. AN EXAMPLE OF THREE LLH COMBINATIONS

Iteration ILS		**1500**
Max. Iter. LS		*7*
LLH		***abc***
Trials	***TINY***	***SMALL***
Average	20.9	1182
Worst	23	1242
Best	18	1086

TABLE VII. AN EXAMPLE OF TWO LLH COMBINATIONS

Iteration ILS		**1500**
Max. Iter. LS		7
LLH		**ab**
Trials	***TINY***	***SMALL***
Average	28	1199.6
Worst	28	1254
Best	28	1140

TABLE VIII. AN EXAMPLE OF TWO LLH COMBINATIONS

Model	**TINY**	**SMALL**
Combination of 5	abcef*	acdef
	19.3^	1087
Combination of 4	bdef	bcdf
	19.1	1069.8
Combination of 3	bde	abf
	18.3	1076.6
Combination of 2	bc	bf
	19.5	1105

TABLE IX. RESULT OF ILS-SA

Cool Rate	**0.7**	
Temp	**100**	
Iteration ILS	**2000**	**5000**
Data	***SMALL***	***TINY***
Average	778	8.6
Worst	934	10
Best	620	6

TABLE X. COMPARISON ILS-SA WITH BENCHMARK

Dataset	**Small**	**Tiny**
ILS-SA	6	776
Marlúcio Alves Pires and D. Holm & R. Mikkelsen	4	200

In the sixth scenario, there is an experiment on LLH parameters by using a combination of 2 out of 6 existing LLH. The ILS and LS Iteration parameters do not change in value and the ILS iteration used was 7 with examples as in Table VII. Table VIII shows the best combination scenario for each dataset, both tiny and small. The tiny dataset will produce the smallest penalty if a combination of LLH move2TS, move2RM, and move RMTS are used. On the other hand, the small dataset can produce the smallest penalty if the combination of LLH move2TS, move1RM, move2RM, and moveRMTS2 are used.

2) ILS-SA: The first scenario of ILS-SA is to changes the value of cooling rate (alpha) to determine the effect of the cooling rate variable on the algorithm performance. The cooling rates used are 0.1, 0.5, 0.7, and 0.9. In 2nd scenario, the initial temperature is changed with the ILS iteration value and constant cooling rate (Table IX). This scenario aims to determine the effect of the initial temperature. In this scenario, the modified parameter is the ILS iteration with a constant cooling rate and initial temperature. This scenario aims to determine the effect of iteration variables on algorithm performance.

C. Algorithm Comparison

To know the relative performance of ILS-SA algorithm, the performance ILS-SA and ILS-HC are compared with hill climbing algorithm (HC). The trials used the Tiny dataset and also the Small dataset with the result as in Fig. 3 and Fig. 4. The result with a tiny dataset shows that ILS algorithm has a better performance by generating a penalty of 37 for ILS-HC and a penalty of 6 for ILS-SA after 2000 iterations. In contrast, Hill Climbing tends to produce a higher penalty of 42. Similarly, the small dataset also shows that the ILS performance is better than HC by resulting penalty of 1034 using ILS-HC and 776 using ILS-SA, lower than HC that lead a total penalty of 1116. In summary, ILS-SA outperforms ILS-HC and HC algorithms. Whereas ILS-HC outperforms HC algorithms. This finding shows that iterated local search algorithms outperform local search algorithm. The comparison of ILS-SA with the reported benchmark is shown in Table X. As shown by Table X, the proposed algorithm ILS-SA is competitive with the benchmark algorithm.

CONCLUSION

The study has identified some of the following conclusions. The application of the Iterated Local Search algorithm with hyper heuristic framework for course schedule automation and optimization is proven to be able to produce better final objective function values. However, if there are too many LS iterations, the resulting solution can be trapped in the local optima. The choice of the number of LS Max Iterations also affects the final objective function's value. The more the number of Max Iterations on the LS is performed, the better final objective function will be obtained.

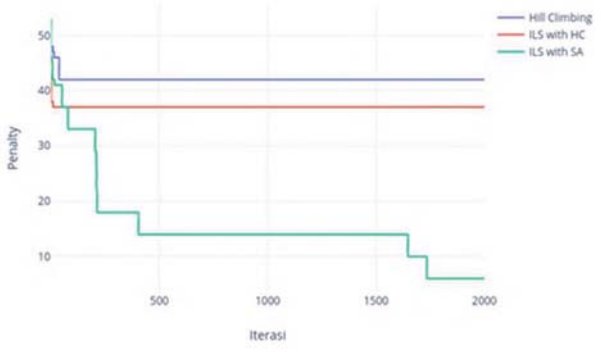


Fig. 3. Comparison between ILS and HC with Small Data

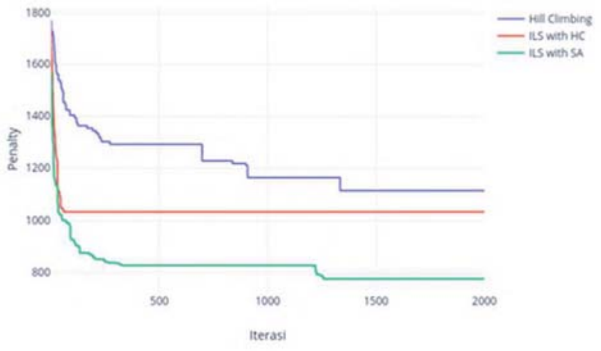


Fig. 4. Comparison between ILS and HC with Small Data

The choice of low-level heuristics influences the objective function where even though the method used is only *move*, the application of the move method with a combination of transferred attributes results in a better value for the final destination function. The choice of cooling rate influences the final objective function in which the more the cooling rate is close to 1, the more likely it will be to accept the worse solution. However, it does not rule out the possibility of getting a better solution. The selection of the initial temperature affects the final objective function in which a greater the initial temperature leads to a possibility of receiving a worse solution so that the value obtained will have a fluctuating value and is more likely to reach a better solution. The Iterated Local Search algorithm, ie. ILS-SA and ILS-HC outperforms HC algorithm, whereas ILS-SA outperforms ILS-HC. Finally, compared to benchmark algorithm, ILS-SA is very competitive.

FURTHER RESEARCH

Further studies regarding the efficiency of TSP to solve similar problem would be worthwhile if more dataset and more complex scenarios is performed. Another improvement area is to fully accomplish all parameter combinations that can help find the most influential parameters in producing an optimal solution. Furthermore, low-level heuristics used in this study are limited to moves and swaps, so adding other ones will generate a more optimum solution.

REFERENCES

[1] W. A. Puspaningrum, A. Djunaidy, and R. A. Vinarti, "Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika di Jurusan Sistem Informasi ITS," *J. Tek. ITS*, vol. 2, no. 1, pp. A127–A131, 2013.

[2] D. A. R. Wati and Y. A. Rochman, "Model Penjadwalan Matakuliah Secara Otomatis Berbasis Algoritma Particle Swarm Optimization (PSO)," *J. Rekayasa Sist. Ind.*, vol. 2, no. 1, pp. 22–31, 2013.

[3] G. H. G. da Fonseca, H. G. Santos, T. Â. M. Toffolo, S. S. Brito, and M. J. F. Souza, "GOAL solver: a hybrid local search based solver for high school timetabling," *Ann. Oper. Res.*, vol. 239, no. 1, pp. 77–97, Apr. 2016, doi: 10.1007/s10479-014-1685-4.

[4] K. Alaykiran and M. Hacibeyoglu, "Using Iterated Local Search to Solve the Course Timetabling Problem at Engineering Faculty of Necmettin Erbakan University," *Int. J. Eng. Appl. Sci.*, vol. 3, no. 12, p. 257538, 2017.

[5] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A Classification of Hyper-heuristic Approaches," Springer, Boston, MA, 2010, pp. 449–468.

[6] T. Song, S. Liu, X. Tang, X. Peng, and M. Chen, "An iterated local search algorithm for the University Course Timetabling Problem," *Appl. Soft Comput. J.*, vol. 68, pp. 597–608, Jul. 2018, doi: 10.1016/j.asoc.2018.04.034.

[7] J. A. Soria-Alcaraz, E. Özcan, J. Swan, G. Kendall, and M. Carpio, "Iterated local search using an add and delete hyper-heuristic for university course timetabling," *Appl. Soft Comput. J.*, vol. 40, pp. 581–593, Mar. 2016, doi: 10.1016/j.asoc.2015.11.043.

[8] T. Müller, H. Rudová, Z. Müllerová, T. Müller, H. Rudová, and Z. Müllerová, "University course timetabling and International Timetabling Competition 2019."

[9] A. Muklason, "Solver penjadwal ujian otomatis dengan algoritma maximal clique dan hyper-heuristics," in *Seminar Nasional Teknologi Informasi, Komunikasi, dan Industri*, 2017, pp. 94–101.

[10] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated Local Search*, In Handboo. Boston, MA: Springer, 2003.

[11] F. Glover, *Clever Algorithms*. Australia: Lulu, 2011.

[12] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated Local Search," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, 2006, pp. 320–353.

[13] Institut de Recherches Interdisciplinaires et de D´eveloppements en Intelligence Artificielle (IRIDIA), "Technical Report Series," Bruxelles, 2018.

[14] [N. Pillay, "Hyper-Heuristics for Educational Timetabling."

[15] N. Pillay, "A review of hyper-heuristics for educational timetabling," *Ann. Oper. Res.*, vol. 239, no. 1, pp. 3–38, Apr. 2016, doi: 10.1007/s10479-014-1688-1.

[16] S. L. Goh, G. Kendall, and N. R. Sabar, "Improved local search approaches to solve the post enrolment course timetabling problem," *Eur. J. Oper. Res.*, vol. 261, no. 1, pp. 17–29, Aug. 2017, doi: 10.1016/j.ejor.2017.01.040.

[17] A. Abuhamdah *et al.*, "Population based Local Search for university course timetabling problems," *Appl Intell*, vol. 40, pp. 44–53, 2014, doi: 10.1007/s10489-013-0444-6.

[18] H. M. Kartika and M. Ahmad, "Self Adaptive and Simulated Annealing Hyper-Heuristics Approach for Post-Enrollment Course Timetabling," *J. Phys. Conf. Ser.*, vol. 1577, no. 1, p. 012033, 2020.

[19] A. Muklason, R. G. Irianti, and A. Marom, "Automated course timetabling optimization using tabu-variable neighborhood search based hyper-heuristic algorithm," in *Procedia Computer Science*, 2019, vol. 161, pp. 656-664.

[20] A. Lemos, P. T. Monteiro, and I. Lynce., "ITC 2019: University Course Timetabling with MaxSAT," 2021.