Self Adaptive and Simulated Annealing Hyper-Heuristics Approach for Post-Enrollment Course Timetabling

H M Kartika¹, M Ahmad¹

¹ Information Systems Department, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia

Abstract. One of timetabling problem in education field is Post-Enrollment Course Timetabling (PE-CTT). The challenges faced in the PE-CTT are differences in type of problems, a number of limitations, and requirements that differ from one university to another. It is difficult to find common and effective solutions. State-of-art method that can develop more general systems by using cheaper methods and still being able to solve problems is Hyper-Heuristic approach.

The Self-Adaptive Strategy is used as a strategy for selecting Low-Level-Heuristics (LLH) and Simulated Annealing as a Move Acceptance (MA) strategy to solve the course timetabling problems. Self-adaptive can increase the level of convergence to the optimal value in the optimization process. Simulated Annealing can accept solutions that are no better so that the final solution is not trapped in the local optima solution.

The contributions of this paper are state-of-art hybridization of Self-Adaptive and Simulated Annealing Hyper-Heuristics approach to solve post-enrollment course timetabling problem. The Self Adaptive and Simulated Annealing Hyper-Heuristics then compared with Simple Random and Simulated Annealing Hyper-Heuristics. Tests will be carried out on Socha dataset. The hybridization of Self Adaptive and Simulated Annealing shows competitive results.

1. Introduction

Course timetabling in university has variation of complexity. The effect of not scheduling properly for students with different semester taking the same courses can cause schedules to clash. Availability of equipment in the room, room features, and time slots must also be considered.

Timetabling is allocating resources being used in place and time within constraints to satisfy a set of desirable objectives [1]. Appropriate solutions for scheduling problems are usually solutions that meet all hard constraints and calculate soft constraints [2]. Violations of hard constraints can render a solution unfeasible, while violations of soft constraints can incur a penalty. Timetabling can be applied to education, sports, transportation, work by machines or employees, etc. Timetabling in the field of education is very important in university's operational. Timetabling in the field of education can be divided into Curriculum-Based Course Timetabling (CB-CTT), Post-Enrollment Course Timetabling (PE-CTT), and Examination Timetabling. In examination timetabling, there have been many researchs and one of them the complexity also add a fairness perspective from students [3].

Course timetabling is hard to solve with exact algorithm because it is NP hard. Along with the development of technology, timetabling can now be automated. Various kinds of timetabling problems have been discussed in the literature. The course timetabling problem also has different problems, a number of restrictions, and different requirements in one university to another so that it is difficult to
find a general and effective solution. One of the state-of-the-art method for solving course timetabling problem is Hyper-heuristics approach. Hyper-Heuristics proved to be effective for providing generalized solutions by exploring heuristic space instead of a solution space [4]. Post-Enrollment Course Timetabling has solved by different solutions approach. The solutions approach mainly categorized by traditional methods and heuristics. Traditional methods such as exact algorithm took an expensive cost while solving the NP-Hard problem. Heuristics is applied where traditional methods fail to give solutions, heuristic does not promises globally best solution. The recent trend for solving university timetabling is hybridization concept [5] and Hyperheuristics. An iterated local search Hyper-heuristics which combines a set of move operators has been proposed [6]. The algorithm produced competitive results. A hybridization of Tabu Search and Variable Neighborhood Search based on Hyper-Heuristic Algorithm has been conducted to solve course timetabling [7]. The hybridization algorithm tested on two real word problems and successfully automated the process of course timetabling. In examination timetabling, Greedy and Late Acceptance based on Hyperheuristic has been proposed [8]. The results are better than manual timetables. A Simulated Annealing approach has been used to the PE-CTT problem [9][10]. The results proves that the algorithm outperform most of the previous approaches to the problem. Self-adaptive strategy has been proposed to solve Lot-streaming flow shop timetabling problem [11]. Self-adaptive could produce better solutions with the knowledge of previously successful values [12]. Self-adaptive can increase the level of convergence to the optimal value in the whole optimization process [13]. The hybridization of Simulated Annealing and Self-Adaptive is expected to produce an optimal solution.

2. Problem definition

2.1. PE-CTT
The timetabling problem has been applied to the fields of education, sports, transportation, and employees. Types of timetabling in the education field are examination timetabling and course timetabling. The differences between examination timetabling and course timetabling are the space and time period [14]. In examination timetabling, multiple exams can be scheduled in the same room while course timetabling can't have courses scheduled in the same room. On course timetabling, students can take two or more courses a day, while on the examination timetabling it is minimized as much as possible.

2.2. Socha dataset
Socha is a dataset developed by Ben Paechter. Socha has 11 instances, consisting of 5 small-sized instances, 5 medium-sized instances, and 1 large-sized instances. The timetabling will arrange 45 timeslot with 9 timeslot every day in 5 working days from Socha dataset. In Socha dataset, there are three hard constraints:
- At the same timeslot, students can't be scheduled more than one course
- The room has features needed by the course and can accept student capacity who enrolled to the course
- The room used for a timeslot only for a course
As for the soft constraints, a penalty is given for violating:
- A student has a class in the last timeslot on that day
- Students who have two courses in a row
- Students who have only one course in a day

3. Methodology
The methodology used for the research can be seen in Figure 1.

3.1. Literature review
In the literature review step, a literature review conducted concerning the course timetabling and hyperheuristics. Collection of references such as reference lists, previous research, and documents that support the research are studied in this step. Literature study explained in the introduction.

### 3.2. Data selection
In this stage, dataset Socha was selected. Socha dataset is a popular dataset in university course timetabling. Socha dataset will be tested for the algorithm performa.

### 3.3. Modelling course timetabling
In this step, mathematical model of course timetabling problem which exist in Socha dataset was formulated. The model should ensure that limitations are considered and objective function is satisfied. The mathematical model then processed by algorithm implementation. The goal of course timetabling is ensured to satisfy hard constraints and minimized the violations of soft constraints. Notations table can be seen in Table 1.

#### Table 1. Notations Mathematical Model

| Notations | Explanations |
|-----------|--------------|
| E         | Event \( (e_1, \ldots, e_n) \) |
| S         | Student \( (s_1, \ldots, s_m) \) |
| T         | Timeslot \( (t_1, \ldots, t_p) \) |
| R         | Room \( (r_1, \ldots, r_k) \) |
| F         | Features \( (f_1, \ldots, f_l) \) |

Objective functions of this problem is minimizing the soft constraints violations as seen in (1).

\[
z = \min \sum_{i=1}^{n} SC_i
\]  \hspace{1cm} (1)

A feasible timetable is one that meet all the hard constraints. Hard Constraint 1, no students scheduled to attend more than one class at a time as seen in (2). \( m_{se} \) valued 1, if student \( s \) attend course \( e \). \( c_{etr} \) valued 1, if event \( e \) scheduled to hold in room \( r \) and timeslot \( t \).

\[
HC_1 = \sum_{e \in E, r \in R} c_{etr} \cdot m_{se} \leq 1 \hspace{1cm} e \in E, s \in S, r \in R, t \in T
\]  \hspace{1cm} (2)
Hard Constraint 2, the room has features needed by the course and can accommodate the capacity number of students who take the course as seen in (3). \( k_{er} \) valued 1, if event size e ≤ room r capacity. \( b_{er} \) valued 1, if event e need features f, and room r has features f.

\[
HC_2 = k_{er} \cdot b_{er} \cdot x_{etr} = x_{etr}, \quad e \in E, r \in R, t \in T
\]  
(3)

Hard Constraint 3, room used in each timeslot by only one subject.

\[
HC_3 = \sum_{e \in E} c_{etr} \leq 1, \quad r \in R, t \in T
\]  
(4)

3.4. Algorithm implementation

The algorithm implemented are Self Adaptive and Simulated Annealing in a Hyperheuristics framework. Java programming language is used for implementation. Input of course timetabling solver is the courses that students enrolled, total rooms available, room size, room features, and the features that needed for a course. The output is an optimum course timetabling. The state-of-art of this work is hybridization of Self-Adaptive and Simulated Annealing Hyper-Heuristic approach.

In this step, Self-Adaptive and Simulated Annealing was designed and implemented. Self-Adaptive strategy in Hyper-Heuristics LLH selection choose the LLH that successfully produce a better solution from the previous solution. The initial chosen LLH called Neighboring List (NL) and the LLH which successfully produce a better solution called Winning Neighboring List (WNL). According to [15], 75% elements of NL filled with WNL and 25% elements of NL filled with random LLH.

Simulated Annealing Algorithm is a local search algorithm that was first conceived by Kirkpatrick in 1983. The initial idea of Simulated Annealing was to avoid being trapped in local optima, by accepting a solution that was no better if it didn't exceed a certain number of iterations / "temperatures". The naming of Simulated Annealing is taken from the physics theory from strengthening steel process. Strengthening of the steel is done by heating the steel to its boiling point, the atoms in the steel will move freely. Then the steel is cooled gradually until it reaches a certain point with the aim of its energy decreasing slowly. Hybridization of Self-Adaptive and Simulated Annealing Hyper-Heuristic flowchart could be seen in Figure 2.

3.5. Experiment and result analysis

In this stage, results analysis conducted after the experiment by course timetabling solver. If the result is not optimal, review against the model obtained will be done. If the results meets all hard constraints and penalty value of soft constraints is minimized then move on to the next process. Performa analysis of Self Adaptive – Simulated Annealing will be compared with Simple Random – Simulated Annealing.

![Figure 2. Self-Adaptive and Simulated Annealing Design and Implementation](image)

4. Result and analysis
Results of Self-Adaptive and Simulated Annealing for Post-Enrollment Course Timetabling is in Table 2. Table 2 shows the average initial solution, the best & worse solution, the average final solution, and the standard deviation of each instances.

### Table 2. Self-Adaptive and Simulated Annealing Results

| Instance | Avg Initial Sol | Best | Worst | Avg | Std. Dev |
|----------|-----------------|------|-------|-----|----------|
| SMALL1   | 312.9           | 0    | 2     | 0.6 | 0.67     |
| SMALL2   | 322.7           | 0    | 4     | 1.2 | 1.48     |
| SMALL3   | 300.7           | 0    | 5     | 1.6 | 1.59     |
| SMALL4   | 167.9           | 0    | 5     | 2.3 | 1.66     |
| SMALL5   | 456.4           | 0    | 2     | 0.4 | 0.88     |
| MEDIUM1  | 1034.7          | 229  | 309   | 259.2 | 24.65 |
| MEDIUM2  | 1060.2          | 211  | 277   | 247.3 | 21.20 |
| MEDIUM3  | 1050.8          | 257  | 295   | 281.6 | 10.96 |
| MEDIUM4  | 1082.9          | 204  | 259   | 225.1 | 16.69 |
| MEDIUM5  | 1163.8          | 143  | 229   | 173.8 | 27.04 |
| LARGE    | 1928.3          | 956  | 1181  | 1046.8 | 59.62 |

Hybridization of Simple Random and Simulated Annealing is selected for performance comparison. The parameter of Simulated Annealing for both algorithm tuned in the same value. Comparison of Self-Adaptive and Simulated Annealing with Simple Random and Simulated Annealing is in Table 3.

### Table 3. Comparison of Self-Adaptive and Simulated Annealing with Simple Random and Simulated Annealing

| Instance | Self-Adaptive Simulated Annealing | Simple Random Simulated Annealing |
|----------|-----------------------------------|-----------------------------------|
|          | Best | Average | Best | Average |
| SMALL1   | 0    | 0.6     | 0    | 1.1     |
| SMALL2   | 0    | 1.2     | 0    | 1.2     |
| SMALL3   | 0    | 1.6     | 0    | 2.2     |
| SMALL4   | 0    | 2.3     | 0    | 1.7     |
| SMALL5   | 0    | 0.4     | 0    | 0.4     |
| MEDIUM1  | 229  | 259.2   | 230  | 255.2   |
| MEDIUM2  | 211  | 247.3   | 202  | 235.4   |
| MEDIUM3  | 257  | 281.6   | 234  | 270.1   |
| MEDIUM4  | 204  | 225.1   | 219  | 234.8   |
| MEDIUM5  | 143  | 173.8   | 154  | 171.3   |
| LARGE    | 956  | 1046.8  | 1002 | 1085    |

As seen in table 3, Self-Adaptive Simulated Annealing produce a competitive results compared to simple random simulated annealing. In small instances, Self-Adaptive produce more optimal results with smaller average final solution. In large instances, Self-Adaptive produces more optimal results with smaller average final solution.

Best results of Self Adaptive – Simulated Annealing will be compared to list benchmark in table 5. All the three algorithms on the list has been applied to socha dataset.

### Table 4. List Benchmark

| ID   | Algorithm                        | Researcher                  |
|------|----------------------------------|-----------------------------|
| RII  | Randomised Iterative Improvement [16] | Salwani Abdullah, dkk.     |
| TS   | Tabu Search [17]                 | Edmund K. Burke, dkk.       |
| MMAS | Max-Min Ant Systems [18]         | Krzysztof Socha, dkk.       |
### Table 5. Benchmark Result Comparison

| Instances | SASA |    | RII |    | TS |    | MMAS |    |
|-----------|------|----|-----|----|----|----|------|----|
|           | Best | Avg | Best |    | Best |    | Best |    |
| Small 1   | 0    | 0,6 | 0    | 1  | 1  |    |      |    |
| Small 2   | 0    | 1,2 | 0    | 2  | 3  |    |      |    |
| Small 3   | 0    | 1,6 | 0    | 0  | 1  |    |      |    |
| Small 4   | 0    | 2,3 | 0    | 1  | 1  |    |      |    |
| Small 5   | 0    | 0,4 | 0    | 0  | 0  |    |      |    |
| Medium 1  | 229  | 259,2 | 242 | 146 | 195 |    |      |    |
| Medium 2  | 211  | 247,3 | 161 | 173 | 184 |    |      |    |
| Medium 3  | 257  | 281,6 | 265 | 267 | 248 |    |      |    |
| Medium 4  | 204  | 225,1 | 181 | 169 | 164,5 |    |      |    |
| Medium 5  | 143  | 173,8 | 151 | 303 | 219,5 |    |      |    |
| Large     | 956  | 1046,8 | 757 | 1166 | 851,5 |    |      |    |

### 5. Conclusions

This paper aimed at minimizing total penalties for post-enrollment course timetabling. The algorithm used in this work is Self-Adaptive and Simulated annealing that produced a competitive results. Comparison of Self-Adaptive Simulated Annealing with other benchmarks results shows that Self-Adaptive could produce an optimal value in small instances. The hybridization of Self Adaptive and Simulated Annealing successfully delivered a competitive results. However it still can be improved in the search / LLH selection or move acceptance.

### 6. References

[1] A. Wren, “Scheduling, timetabling and rostering - A special relationship?,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1153, pp. 46–75, 1996.

[2] E. K. Burke and G. Kendall, *Search Methodologies*. 2014.

[3] A. Muklason, A. J. Parkes, E. Özcan, B. McCollum, and P. McMullan, “Fairness in examination timetabling: Student preferences and extended formulations,” *Appl. Soft Comput. J.*, vol. 55, pp. 302–318, 2017.

[4] N. Pillay, “A review of hyper-heuristics for educational timetabling,” *Ann. Oper. Res.*, vol. 239, no. 1, pp. 3–38, 2016.

[5] R. Ilyas and Z. Iqbal, “Study of Hybrid Approaches used for University Course Timetable Problem (UCTP),” in *The 10th IEEE Conference on Industrial Electronics and Applications (ICIEA 2015)*, 2015, pp. 696–701.

[6] J. A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga, and E. K. Burke, “Effective learning hyper-heuristics for the course timetabling problem,” *Eur. J. Oper. Res.*, vol. 238, no. 1, pp. 77–86, 2014.

[7] A. Muklason, R. Galih, and A. Marom, “ScienceDirect Automated Course Timetabling Optimization Using Tabu-Variable Neighborhood Search Based Hyper-Heuristic Algorithm,” vol. 00, 2019.

[8] A. Muklason, P. C. Bwananesia, Y. T. Sasmi Hidayatul, N. D. Angresti, and V. A. Supoyo, “Automated Examination Timetabling Optimization Using Greedy-Late Acceptance-Hyperheuristic Algorithm,” *Proc. 2018 Int. Conf. Electr. Eng. Comput. Sci. ICECOS 2018*, vol. 17, pp. 201–206, 2019.

[9] S. Leng, 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, 2017.

[10] S. Ceschia, L. Di Gaspero, and A. Schaerf, “Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem,” *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1615–1624, 2012.
[11] Q. K. Pan, M. Fatih Tasgetiren, P. N. Suganthan, and T. J. Chua, “A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem,” Inf. Sci. (Ny.), vol. 181, no. 12, pp. 2455–2468, 2011.

[12] X. Li and M. Yin, “Self-adaptive constrained artificial bee colony for constrained numerical optimization,” Neural Comput. Appl., vol. 24, no. 3–4, pp. 723–734, 2014.

[13] W. Liu, T. Zhang, Y. Liu, N. Zhang, H. Tao, and G. Fu, “Improved artificial bee colony algorithm based on self-adaptive random optimization strategy,” Cluster Comput., vol. 4, pp. 1–10, 2018.

[14] E. K. Burke and S. Petrovic, “Recent research directions in automated timetabling,” Eur. J. Oper. Res., vol. 140, no. 2, pp. 266–280, 2002.

[15] A. Muklason, “Hyper-heuristics and fairness in examination timetabling problems,” 2017.

[16] S. Abdullah, E. K. Burke, and B. McCollum, “Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem,” Oper. Res. Comput. Sci. Interfaces Ser., vol. 39, no. January, pp. 153–169, 2007.

[17] E. K. Burke, G. Kendall, and E. Soubeiga, “A Tabu-Search Hyperheuristic for Timetabling,” J. Heuristics, vol. 2003, no. 9, pp. 451–470, 2004.

[18] K. Socha, J. Knowles, and M. Sampels, “A MAX-MIN ant system for the university course timetabling problem,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 2463, pp. 1–13, 2002.