Automated Examination Timetabling Optimization Using Greedy-Late Acceptance-Hyperheuristic Algorithm

Ahmad Muklason
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
mukhlason@is.its.ac.id

Putri C Bwananesia
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
putribwananesia@gmail.com

Sasmi Hidayatul Y T
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
sasmi16@mhs.is.its.ac.id

Nisa D Angresti
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
nisa.dwi.angresti16@mhs.is.its.ac.id

Vicha Azthanty Supoyo
Department of Information Systems
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
vicha.supoyo16@mhs.is.its.ac.id

Abstract— **Due to its non-deterministic polinomial (NP)-hard nature, exam timetabling problem is one of challenging combinatorial optimisation problems. Therefore, it attracts researchers especially in operation research and artificial intelligence fields for decades. Since the problem is very complex, exam timetable in many universities is developed manually which is very time consuming. This paper presents a new hybrid algorithm, i.e. greedy-late acceptance within hyper-heuristic framework to generate and optimise exam timetable automatically. Greedy algorithm is used to generate initial solution, whereas late acceptance is used as move acceptance strategy. The algorithm is simple but proven powerfull. The algorithm is tested over two datasets from real-world exam timetabling problem from Information Systems Department, Institut Teknologi Sepuluh Nopember (ITS). Over 11 different scenarios, the experimental results show that in addition to its ability to generate feasible solution, the algorithm also could produce more optimal solutions compared to the timetables generated manually.**

Keywords*— *exam timetabling, automated timetabling, greedy algorithm, late acceptance, hyper-heuristic

I. Introduction

The exams are still the main key success factors measuring the students' academic achievement within almost all universities. Therefore, a good exam timetable is believed could help students to succeed in their exams. In the Department of Information Systems, ITS, within a semester, there are at least two required exams that are compulsory for the students, namely the midterm exam and the final exam. Currently, the exam timetabling is still being a problem. This problem can be seen from the two different perspectives, i.e. the department as organisers and students taking the exam. From the department perspective, it is urgently needed a good support system that could generate and optimise exam timetable automatically. From the students' perspective, to achieve maximum results, the students expect a good timetable that give them enough time to prepare the exams, i.e. only an exam per day. Therefore, this work come up to cope with these problems.

Currently, the exam timetable in the Department of Information Systems, ITS is constructed manually. Normally, the department needs 1-2 weeks to prepare the timetable. In addition, exam timetabling is not flexible with changes. Making changes in the manual timetable is very time consuming because there are many hard constraints must be checked manually. The other problem is that within the manually generated timetable, it is found that there are still some students who must sit for two exams in the same timeslot or more than an exam per day. Ideally, the students should have enough time to prepare the exam, by having only an exam per day.

To cope with this problem, this research aims at developing an automated exam timetabling solver to generate an optimal exam timetable automatically. The main contributions of this paper are two folds. Firstly, investigating a new hyper-heuristic algorithm, by hybridising two algorithms, i.e. greedy and late acceptance algorithms, to solve exam time tabling problems which have been proven in [1] as NP-complete problem. Secondly, providing new real-world datasets for exam timetabling problem that could be used other researchers as a new benchmark problem in testing different algorithms in the future works.

The rest of this paper is organised as follow. The literature review in section II presents the state-of-the art of the problem followed by the methodology in section III. Section IV discusses the experiment carried out in this work, whereas Section V elaborates the experimental results. This paper will be concluded in Section VI highlighting the main findings and the future works.

II. Literature Review

This section presents the state-of-the-art of the research in examination timetabling problems as well as the algorithms to solve the problem. Further, in the end of this section a gap analysis of this work compared to the previous works are provided.

2.1 Examination Timetabling

Examination timetabling problem is one of the most challenging combinatorial optimisation problems[2]. Basically, timetabling can be defined as a problem with four parameters, namely **T** (set of times), **R** (the set of resources), **M** (the set of meetings), and **C** (the set of constraints). Timetabling is used to allocate time and resources on a particular meeting by satisfying the maximum constraints[3]. According to Schaerf [4], educatoional timetabling problem

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

can be divided into two folds: courses timetabling and exam timetabling.

The general objective to be achieved in the timetabling problem is to satisfy all the hard constraints and minimize violations of the soft constraints, which can vary depending on the policies of each agency. These constrains have been examined by Burke [5]. A solution that can satisfy all hard constraints is referred to as a feasible solution. If the solution can also satisfy all the soft constraints, then that solution is referred to as the perfect solution. However, to produce an optimal solution, the existence of a perfect solution becomes not so important. For example in terms of exam timetabling, may occur a violation of one of the soft constraints i.e. the distance between the two exams is at least 5 timeslots.

The exam timetabling problem studied in this paper is formulated with desicion variables, hard constrainsts and soft constraints that in turn become the objective function defined in (1), (2), (3), (4), (5) as follow:

$$X_{it} \begin{cases} 1, \text{ if } i^{th} \text{ course is scheduled at } t^{th} \text{ timeslot} \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

X_{it} is the decision variable, i = {1,2,3,...,n} is the course exams sequence and t = {1,2,3,...,k} is timeslot sequence of exam. The hard constraints are:

1) Ensure that all exams are scheduled.

$$\sum_{i=1}^{n}\sum_{t=1}^{k} x_{it} = 1 \quad (2)$$

2) Ensure that there is no clashed exams.

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij} v_{ij} = 0, \; v_{ij} \begin{cases} 1, \text{ if } t_i = t_j \\ 0, \text{ otherwise} \end{cases} \quad (3)$$

i,j = {1,2,3,...,n} is course exams sequence, c_{ij} and v_{ij} are the number of students taking the ith and jth exams, t_i is the timeslot where exam i is scheduled.

3) Ensure that in the same timeslot, total number of students is not exceed the maximum capacity of available exam room. s_i is number of students taking the ith exam.

$$\sum_{i=1}^{n} s_i x_{it} \leq \text{Maximum room capacity} \quad (4)$$

The objective function is to minimize the soft constraint violation that is formulated as proximity cost value (P):

$$P = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij} w_{|t_j - t_i|}}{M}, \; w_{|t_j - t_i|} \begin{cases} 2^{5-|t_j - t_i|}, \text{ if } 1 \leq |t_j - t_i| \leq 5 \\ 0, \text{ if } |t_j - t_i| > 5 \end{cases} \quad (5)$$

$w_{|t_i - t_j|}$ is weight defined as the different of timeslots allocated to ith and jth exams respectively. M is total number of students.

Though it has been studied for decades, the exam timetabling problem is still intensively studied by the researchers especially in the area of operation research and artificial intelligence. For instance, the current study in [12] investigates fairness in exam timetabling problem.

2.2 The Problem Dataset

The problem dataset used in this study is from real-world examination problem during the midterm and final exams in even semester academic year 2016–2017 at the Department of information systems, Sepuluh Nopember Institute of Technology Surabaya. The data consists the list of exam, the list of students who take the exams, and the list of the room, and manual exam timetabling (day and exam sessions). TABLE I shows the summary of the dataset.

2.3 Greedy Algorithm

Greedy algorithm is the most popular algorithms used to solve optimization problems. Optimization aims to find the optimum solution for minimization or maximization problem. It has been done in some researches [5]–[7].

TABLE I. ITS DATASET CHARACTERISTIC

Exams	Number of exams	Number of students	Number of Time slots
Midterm exam	32	567	15
Final exam	32	519	15

In the greedy algorithm, optimization problems were solved gradually. The principle used in the greedy algorithm is to take advantage as much as possible at each decision point, without regard to the consequences that will come up next [8]. Greedy algorithm also has shortcomings, i.e. the optimum solution produced is not necessarily being the global optimum solution, but sub-optimum or pseudo-optimum. It is caused by the case that the greedy algorithm is not extensively search over all alternative solutions. To get more optimal solution, in this study, the greedy algorithm is coupled with other algorithms as carried out in [7].

2.4 Late Acceptance Algorithm

Late Acceptance is a new metaheuristic approach designed for general purpose optimisation over cross problem domains. This algorithm was proposed by Burke and Bykov [8]. Although it is still in the the early stage investigations, the prior studies shown that it is a promising method.

Late acceptance algorithm is categorised within iterative search techniques but it uses more sophisticated move acceptance mechanism. It is not like the mechanism commonly used in metaheuristic search methods (e.g. on a Hill-Climbing), in which in at each iteration the candidate of the new generated solution will be compared to the current solution. In late acceptance, the new candidate solution is compared to some solutions from the previous iterations. In the final stage of late acceptance algorithm, the cost function of a candidate solution is compared to the competitor's "previous" solution and only candidate solution with a better or same cost function value are accepted [2].

2.5 Hyperheuristic Algorithm

The basic idea of hyperheuristic algorithm is to develop a more general algorithm contrast to the standard approach of metaheuristic that is specifically developed for particular problem domain. Generally, metaheuristic requires intensive parameter tuning and spesific problem domain knowledge. So, the different problem instances requires different parameter tuning. This problem can be tackled by a new approach namely Hyperhuristic [8].

Hyperheuristic is defined as a collection of approaches to automate the process of parameter tuning. This automation could be carried out by employing machine learning methods. This method selects and combines the simple heuristic or generate new heuristic from the existing heuristic, thus it solved a very difficult manually computational search [9].

Fig. 1 shows the framework of hyperheuristic [10]. It contains problem domain barrier which separates hyperheuristic strategy from problem domain. Hyperheuristic does not depend on a spesific problem domain, so it does not require parameter tuning for each problem domain. This is because parameter tuning is automatically done by hyperheurisitic [8].

Hyperheuristic process is started from generating initial solution, then it is optimized. Stages to find the solution is carried out iteratively until the stopping condition is met. In this study, the initial solution is constructed using greedy algorithm. Further, a simple random selection over the low-level heuristic (LLH) is applied to perturbate the previous solution and generate a new solutio. Finally, in this study the late acceptance algorithm strategy us used as move acceptance method.

Although hyper-heuristic method is relatively new, previous studies have shown that this method is effective to solve many problem domains, such as in [6] that is used to solve orienteering problems. The current survey over hyper-heuristics can be found in [13]. The state-of-art methods to solve exam timetabling problem is meta-heuristics (see [12]), therefore in this study hyper-heuristic approach is used instead.

3 METHODOLOGY

Generally, the methodology of the research is shown by in Fig 2. The detail is explained as follow:

3.1 Problem Identification

Problem identification stage is the stage to understand the exam timetabling problem in Information Systems Department of Sepuluh Nopember Institute of Technology, to set reseach goal, and to specify the limitations of the research. At this stage previous research literatures as well as study which solve exam timetabling problem are reviewed. The main references used in this research [11].

3.2 Preprocessing Data

In this stage, the data used in the study is collected. The data is taken from real-world problem form undergraduate midterm and final exam timetbaling in the even semester academic year 2016/2017 at Information Systems Department of Sepuluh Nopember Institute of Technology. The data is pre-processed to the format that can be read by the developed system.

3.3 Modelling Exam Timetabling

The modeling stage is the stage of making a mathematical model of exam timetabling problem which exist in Information Systems Department of Sepuluh Nopember Institute of Technology. The model should ensure that there are no conflicting timetabling. In addition, the model is expected to optimize time slot distance between exams.

3.4 Algorithm Implementation

Implementation of algorithm stage is performed by applying a hyperheuristic algorithm for greedy and late acceptance to generate exam timetabling solver. The exam timetabling solver is implemented using the Java programming language that serves to schedule midterm and the final exam in the Information Systems Department of Sepuluh Nopember Institute of Technology. Input of exam timetabling solver is the data of students and their taken exams. The output of this stage is optimum midterm and final exam timetabling.

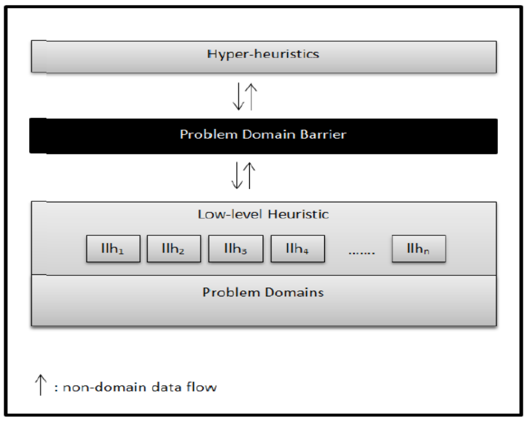

Fig. 1. Framework Hyperheuristic

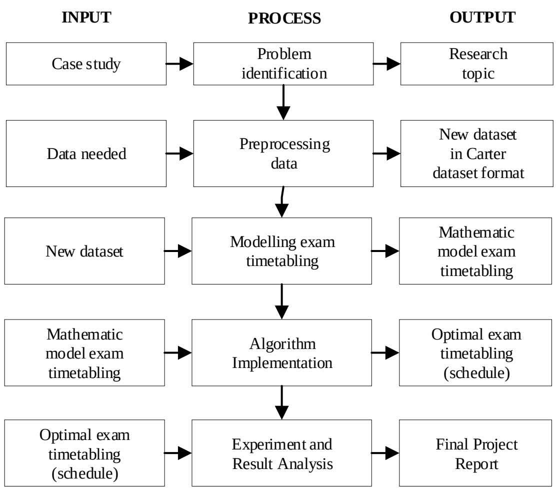
INPUT
PROCESS
OUTPUT
Case study
Problem identification
Research topic
Data needed
Preprocessing data
New dataset in Carter dataset format
New dataset
Modelling exam timetabling
Mathematic model exam timetabling
Mathematic model exam timetabling
Algorithm Implementation
Optimal exam timetabling (schedule)
Optimal exam timetabling (schedule)
Experiment and Result Analysis
Final Project Report

Fig. 2. Research Methodology

3.5 Experiment and Result Analysis

This stage conducts evaluation of the results of the modelling produced by exam timetabling solver. If the result meets all hard constraints and can minimize the number of violations of the soft constraints, then it can move on to the next process. If the result is not optimal, it can be done a review against the model obtained.

The algorithm performance analysis was conducted to evaluate the exam timetabling results obtained. The goal is to achieve optimal exam timetabling by providing sufficient breaks for students, that means a student can obtain a test schedule that dispersed/distributed properly. Here the penalty value used to measure the quality of the generated schedules. The lower the value of penalty the better performance of the algorithm. For calculating the value of penalty, this study uses an executable program that has been provided free of charge through the website of the School of Computer Science, The University of Nottingham [11].

At this analysis stage, the exam timetabling have been produced by the system are compared with the result of exam schedule manually. It also conducted a comparison test with the test schedule schedule produced by the system but with different parameter values to find the results of the schedule the best. Performance of the algorithm is tested by doing a comparison between the exam timetabling are made manually with automated test fixtures produced by solver scheduling exams. The analysis performed includes whether the resulting auto exam timetabling solver already meet all the hard and soft constraints constraints that exist and whether the automatic exam timetabling is the timetabling more optimal compared to manual exam timetabling.

4 IMPLEMENTATION

In general, the algorithms proposed in this study consists of two phases. First phase aims to produce an initial feasible solution by satisfying all the hard constraints. While second phase aims to optimize the initial solution by minimising the total penalties due the violations of the soft constraints.

4.1 Implementation of Greedy Algorithm

In first pahse used greedy algorithm. Greedy algorithm applied to the construction schedule of the initials. Greedy algorithms produce an initial exam timetabling. The initial solution must be clash-free and within a timeslot the maximum room capacity constraint must not be violated.

To make sure the timetable is clash-free, first and second exams will be compared. If both of these courses are conflicting, i.e. share the same students, then the second exam will be scheduled on the different time session from the first exam.

To make sure that the room capacity is not violated, the number of students who take some exams compared to the total capacity of available test room. If the total capacity of the test room is still available (there is still time), then the course will be scheduled on that session and the total available capacity of room will be reduced as much as the number of students who take the test. However if it exceeds the total capacity of the test room, then the subjects that will be scheduled in the next exam session/slot.

If a subject has met both of method above, these courses can be scheduled on a certain slot by using the explore method. However, if the subjects do not meet one or both clash-free and room capacity constraints, it will be transferred to the subsequent slots. The result of the greedy algorithm is a feasible initial exam timetabling. It already meets the limits set, i.e. There should be no conflicting timetabling and should not exceed the total capacity of the exam room.

4.2 Implementation of Late Acceptance Algorithm

In second phase used late acceptance algorithm. This method aims to optimize initial timetabling that was made using the greedy algorithm. So the value of a penalty due to a violation of soft constraints being minimum. The penalty value used namely proximity cost. Late acceptance algorithms is expected to generate optimum exam timetabling with minimal proximity cost value.

Steps performed on late acceptance method is as follows. First step is defined an initial solution which is obtained from the greedy algorithm results. Then, calculate proximity cost value of initial exam timetabling from greedy algorithm. After that, created a queue with the specified amount. At each iteration, new candidate solution will be put in the queue first. Then, proximity cost value of each candidate solution is calculated. Its value is compared to some previous solution proximity cost values. If the new solution proximity cost value is smaller than previous solutions, new solution will replace previous solutions.

5 RESULT AND DISCUSSION

5.1 Objective Function of Manual Timetabling

Before creating automated timetabling systems, proximity cost value of manual exam timetabling must be calculate first. The formula of proximity cost value is defined in (5). Based on calculation result, proximity cost value of manual midterm exam timetabling is 39.570 and 70.265 for manual final exam timetabling. But this result is not feasible because it contains some conflicting exam session. Therefore, exam timetabling made the manual is considered failed to meet one of the limits specified, i.e. it should be no conflicting exam. So the value of proximity cost obtained should get a penalty for violating the hard constraints. After getting penalty, the new proximity cost values of manual timetabling are 208,712 for midterm exam and 19,521,693 for the final exam. These results will be compred to the result using automated timetabling system.

5.2 Result of Greedy Algorithm

Greedy algorithm is used to generate a feasible initial solution. The initial solutions are solutions that have met the limit where there is no conflicting exam schedule and do not exceed the total capacity of room.

Timetabling generated by the greedy algorithm has a value of 60.885 for cost proximity midterm exam and 75.079 for the final exam. This indicates that proximity cost value of exam timetabling result of greedy algorithm is still high enough but it is already feasible. It implies that the result of greedy algorithm is better than manual timetabling. Furthermore, the test timetabling produced by the greedy algorithm is still necessary for optimized by using algorithms late acceptance so that the next value of proximity cost obtained will be better.

5.3 Result of Late Acceptance

This step implement hyperheuristic algorithm. It contains of 2 phases. First phase is selecting low level heuristic (move and swap) randomly, second phase is move acceptance using late acceptance algorithm. It is also used to optimize the exam timetabling result of the greedy algorithm. This study conduct 11 kinds of experiment scenarios with different parameter to get the most optimum exam timetabling. The paremater consist of number of queues and number of iteration. Some scenarios use different heuristic methods to generate new solution, i.e both of swap and move, swap only or move only. TABLE II shows the test scenario used in this research experiment.

The parameter which produce best result are 30 for queue number and 500000 for iteration number. Both of datasets run 11 times and obtained the value of proximity cost as shown in the TABLE III. It shows that midterm exam dataset has better proximity cost value than final exam dataset because midterm exam dataset produce less of minimum, maximum and average proximity cost value than final exam dataset.

TABLE II. TESTING SCENARIOS

Experiment	Number of queues	Number of iterations	Method
1	5000	1000000	Simple Random (Move and Swap)
2	5000	1000000	Move
3	5000	1000000	Swap
4	100	1000000	Simple Random (Move and Swap)
5	100	500000	Simple Random (Move and Swap)
6	50	1000000	Simple Random (Move and Swap)
7	50	500000	Simple Random (Move and Swap)
8	30	1000000	Simple Random (Move and Swap)
9	30	500000	Simple Random (Move and Swap)
10	10	1000000	Simple Random (Move and Swap)
11	10	500000	Simple Random (Move and Swap)

Fig. 3 below shows the scatter plot comparison of experiment results using midterm (UTS) and final exam (UAS) dataset that run 11 times. The computational results showed that final exam dataset (UAS) produce worse proximity cost value but relatively more stable than midterm dataset. It can be infered that midterm exam timetabling is more optimum than final exam timetabling.

5.4 *Comparison of Experiment Result*

This study conduct 11 kinds of experiment scenarios given in TABLE III. Each scenario has different result, so it is important to compare each of them. The value of minimum, maximum and average proximity cost in each experiment is shown by TABLE IV. Some comparations are made for deeper analysis. There are 7 comparison of experiment result as follows.

Experiment 1, 2 and 3 have same parameter but different low level heuristic methods. It has been shown in TABLE III. From TABLE IV, it can be infered that experiment 3 is outperforms experiment 1 and 2 because it has the least minimum, maximum and average proximity cost value. More optimum an exam timetabling, less the value of its proximity cost. The box plot result of experiment 1, 2, and 3 is shown in Fig. 4.

TABLE III. COMPARISON OF EXPERIMENT RESULT

Comparison	Midterm exam	Final exam
Minimum Proximity Cost	36.083	40.896
Maximum Proximity Cost	44.473	44.468
Average Proximity Cost	41.062	42.618

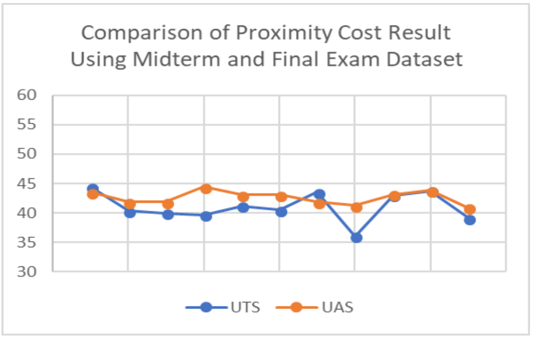

Fig. 3. Experiment result between midterm and final exam dataset

Fig. 4 shows range value of experiment 2 is the highest, it implies that experiment 2 have varied values. Experiment 1 has the most narrow range of values, but experiment 1 does not yield the lowest proximity cost. The lowest proximity cost is indicated by experiment 3 results. However, the difference in proximity cost between experiment 1 and 3 is relatively small. So it can be concluded that experiment 1 and 3 can produce almost the same proximity cost.

The next comparison is between some experiment which has the same queue number and different iteration number. The result of each comparison are shown as follows :

- Comparison between result of experiment 4 and 5, experiment 4 is better than experiment 5. It has lower proximity cost value.
- Comparison between result of experiment 6 and 7, experiment 7 has higher proximity cost value. It implies that experiment 6 is better than experiment 7.
- Comparison between result of experiment 8 and 9, experiment 9 is better than experiment 8. It has lower proximity cost value.
- Comparison between result of experiment 10 and 11, experiment 11 is better than experiment 10.

TABLE IV. EXPERIMENT RESULTS

Experiment	Comparison		
	Minimum Proximity cost	***Maximum Proximity cost***	***Average Proximity Cost***
1	41.252	46.806	52.457
2	50.792	56.157	59.111
3	40.801	46.004	49.910
4	38.882	44.654	47.660
5	39.843	46.466	51.810
6	37.935	41.277	44.635
7	38.644	42.470	45.160

Experiment	Comparison		
	Minimum Proximity cost	***Maximum Proximity cost***	***Average Proximity Cost***
8	37.795	40.083	47.028
9	36.083	41.062	44.473
10	37.795	40.824	44.735
11	36.503	41.686	45.824

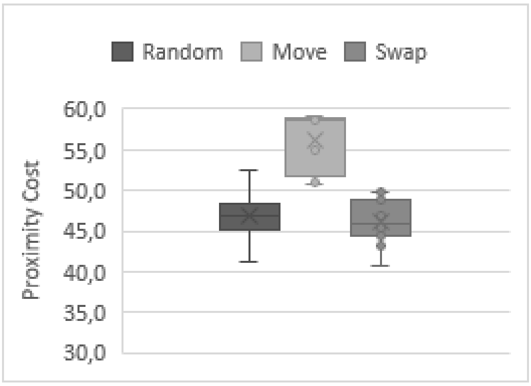

Fig. 4. Box plot of Experiment 1,2 and 3.

These comparison results imply that number of iterations have impact on system performance. The higher number of iteration sometimes produce timetabling which has less proximity cost value, but it takes more time.

Comparisons between some experiment which have the same number of iterations and different number of queues are performed on the following comparison. Comparison between result of experiment 4,6,8, and 10. Experiment 10 has the lowest proximity cost value, so exam timetabling of experiment 10 is the most optimum and the best timetabling. The second comparison is between result of experiment 5, 7, 9, and 11. The best timetabling is generated by experiment 9 because it has the lowest proximity value. From both of these comparisons can be infered that the number of queues have impact on the value of proximity cost. The more number of queue will produce timetabling which has less proximity cost value, so it is more optimum timetabling.

Information Systems Department Sepuluh Nopember Institute of Technology has been arranged exam timetabling manually. It results not feasible exam timetabling and high value of proximity cost. The result of exam timetabling generated by greedy algorithm has lower proximity cost value and no conflict exist. It means that the result of greedy algorithm is better than manual result. The best result are generated by late acceptance algorithm because it has lowest proximity cost. The best exam timetabling optimize by late acceptance from initial solution by greedy algorithm. The three strategy result comparison are shown in Fig. 5.

6 CONCLUSION

Based on the results of this study, it can conclude that Greedy – late acceptance – hyperheuristic algorithm is effective to solve the exam timetabling problem. The proposed algorithm could produce feasible solution for all datatset. In addition, the timetable produced by the algorithm is better than the timetbale generated manually. It is shown by better proximity cost. In term of running time, the algorithm could produce better timetable within only 5 minuted running time. It is really time saving compared to develeop the timetable manually that usually requires 1-2 weeks. For future works, it is worthy to investigate more sophisticated algorithm to be tested over the benchmark datasets investigated in this study.

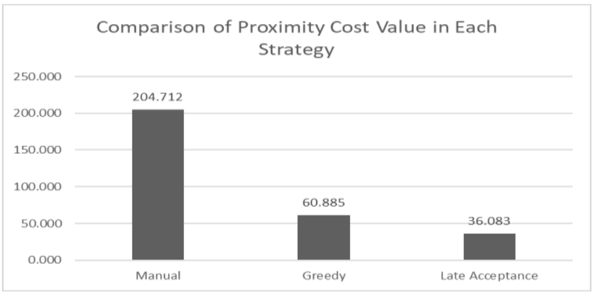

Fig. 5. Proximity cost value of exam timetabling result of manual, greedy and late acceptance algorithm.

REFERENCES

[1] T. B. Cooper and J. H. Kingston, “The complexity of timetable construction problems,” in International Conference on the Practice and Theory of Automated Timetabling, 1995, pp. 281–295.

[2] E. L. Lawler, Combinatorial optimization: networks and matroids. Courier Corporation, 1976.

[3] E. and K. Burke and D. Jeffrey and De Werra, “5.6: Applications to Timetabling,” in Handbook of graph theory, vol. 445, 2004.

[4] A. Schaerf, “A survey of automated timetabling,” Artif. Intell. Rev., vol. 13, no. 2, pp. 87–127, 1999.

[5] E. K. Burke, A. J. Eckersley, B. McCollum, S. Petrovic, and R. Qu, “Hybrid variable neighbourhood approaches to university exam timetabling,” Eur. J. Oper. Res., vol. 206, no. 1, pp. 46–53, Oct. 2010.

[6] J. S. Wijaya, W. Anggraeni, A. Muklason, F. Mahananto, E. Riksakomara, and A. Djunaidy, “Advanced Traveler Information System: Itinerary Optimization as an Orienteering Problem Using Iterative Local Search-Hill Climbing Algorithm,” Procedia Comput. Sci., vol. 124, pp. 173–180, 2017.

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

[8] 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.

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

[10] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, “Hyper-heuristics: An emerging direction in modern search technology,” in Handbook of metaheuristics, Springer, 2003, pp. 457–474.

[11] M. W. Carter, G. Laporte, and S. Y. Lee, “Examination Timetabling: Algorithmic Strategies and Applications,” J. Oper. Res. Soc., vol. 47, no. 3, pp. 373--383, 1996.

[12] Muklason, Ahmad, et al. "Fairness in examination timetabling: Student preferences and extended formulations." Applied Soft Computing 55 (2017): 302-318.

[13] Burke, Edmund K., et al. "Hyper-heuristics: A survey of the state of the art." Journal of the Operational Research Society 64.12 (2013): 1695-1724.