Bulletin of Electrical Engineering and Informatics
Vol. 10, No. 3, June 2021, pp. 1611~1620
ISSN: 2302-9285, DOI: 10.11591/eei.v10i3.2996

Solving examination timetabling problem within a hyper-heuristic framework

Shinta Dewi, Raras Tyasnurita, Febriyora Surya Pratiwi
Department of Information Systems, Institut Teknologi Sepuluh Nopember, Indonesia

Article Info

Article history:

Received Jul 25, 2020
Revised Sep 1, 2020
Accepted Apr 7, 2021

Keywords:

Examination timetabling
Hill climbing
Hyper-heuristic
Largest degree
Tabu search

ABSTRACT

Scheduling exams in colleges are a complicated job that is difficult to solve conventionally. Exam timetabling is one of the combinatorial optimization problems where there is no exact algorithm that can answer the problem with the optimum solution and minimum time possible. This study investigated the University of Toronto benchmark dataset, which provides 13 real instances regarding the scheduling of course exams from various institutions. The hard constraints for not violate the number of time slots must be fulfilled while paying attention to fitness and running time. Algorithm of largest degree, hill climbing, and tabu search within a hyper-heuristic framework is investigated with regards to each performance. This study shows that the Tabu search algorithm produces much lower penalty value for all datasets by reducing 18-58% from the initial solution.

Corresponding Author:

Raras Tyasnurita
Department of Information System
Institut Teknologi Sepuluh Nopember
Sukolilo Surabaya, Jawa Timur, 60111, Indonesia
Email: raras@is.its.ac.id

1. INTRODUCTION

In academic fields, such as universities, they have to manage with the examination timetabling. This timetabling is allocating process space and exam time to the college student. Recurring activities at the examination timetabling create an issue for each college student to determine a scheduling time that takes that course. Therefore, timetabling is one of the interesting issues in the field of Combinatorics optimization. In general, the problem of combinatorial optimization is a mathematical study to find an optimal solution to the preparation, sorting, grouping or selection of discrete objects that usually have finite number [1].

The benchmark dataset is from Toronto or commonly referred to as the Toronto data set. Toronto's data sets contain agencies that correspond to the real-world problem variables because they come from different educational institutions and are categorized as stable data sets and are often used in research [2]. This study investigates on solving the problem of Carter data sets (Toronto) using a hyper-heuristic approach where the search is more focused on the heuristic space. The algorithm used within the hyper-heuristic framework is the largest degree, hill-climbing, and tabu search.

This experimental was conducted by presenting several previous studies by the corresponding approach to solve Toronto's data sets problem. These studies including K. Graham and M. Naimah [3] who used tabu search hyper-heuristic, their research shown that their generic method is able to generate good solutions compared to other studies. Di Gaspero and Schaerf [4] using the Tabu search algorithm. Carter *et al.* [19] use constructive heuristics with backtracking. Caramia *et al.* [5] use greedy constructive heuristics, their research shows that combined of the backtracking strategy and saturation degree sorting rule can create shorter results solutions in shorter computing time. Burke and Newall [6] use local search method, their

research prove that simulated annealing and the great deluxe method is able to increase a good-quality solutions. I. Gabriella & P. Etria [7] using the tabu search algorithm can produce decent solutions with 100,000 iterations. The purpose of this experimental results is to show that the tabu search algorithm is able to produce good penalty results even though the value produced is not the best results.

2. RESEARCH METHOD

2.1. Combinatorial optimization

Combinatoric optimization research is carried out by entering each possible value or developing a search algorithm to choose the best value. Combinatorial optimization is used to determine the minimum or maximum value based on the problem being discusses. Algorithms in combinatorial optimization can solve quite complex problems with a broad scope [8].

2.2. Toronto benchmark dataset

The Toronto benchmark dataset dataset consists of 13 real-time exam schedule problems comprised of three high schools in Canada, five universities in Canada, one university in America, one university in the UK, and one university in the Middle East. It is listed in Table 1 that each dataset shows the final score, number of college students, and timeslot that had been executed in the previous research. The dataset has two boundaries: two examinations with the same participants but different schedules [9].

Table 1. Toronto dataset

Instance	Exams I/II/IIc	Students I/II/IIc	Timeslot
CAR 91	682	16925	35
CAR 92	543	18419	32
EAR 83	189	1108	24
HEC 92	80	2823	18
KFU 93	461	5349	20
LSE 91	381	2726	18
PUR 93	3158	30032	42
RYE 92	486	11483	23
STA 83	138	549	13
TRE 92	261	4360	23
UTA 92	638	21330	35
UTE 92	184	2750	10
YOR 83	180	919	21

2.3. Hyper-heuristic

An approach that uses machine learning is called hyper-heuristic. Hyper-heuristic is used to automate the selection process and the combination of existing heuristic components. Heuristic usage results are common frameworks that can be used to troubleshoot cross-domain problems [10]. Hyper-heuristics frameworks have a feature with selection of mechanisms and moving acceptance criteria. The first thing to do is to choose a heuristic to apply to a single candidate solution, then determine whether the result of that solution yields an acceptable soultions or not [11].

Hyper-heuristic (HH) is introduced as a method of general optimization, able to explore the heuristic space rather than directly to the solution space. HH support traditional algorithms of heuristic [12]. Such ideas can be developed from the excess of each algorithm or combine two or more algorithms, known as the metaheuristic hybrid. At metaheuristic standard and hybrid metaheuristic, the process focuses on a search space solution for a problem. The difference lies in the number of heuristic strategies used. At the standard metaheuristic, the approach used only one, but not with a metaheuristic hybrid. On the other hand, hyper-heuristic focuses on the heuristic search space (strategy being general) [1]. The main component in the problem section of the hyper-heuristic domain is the objective function, formation of the solution's initials, and has a low-level heuristic set. Low-level heuristic is used by hyperheuristics to specific optimization problems. In this case, a hyperheuristic is able to choose low-level heuristic to be used at some point of decision until the termination conditions are met [13]. The methodology part has two phases: the selection of the low-level heuristic (heuristic selection) and the move acceptance. The HH framework can be seen in Figure 1.

2.4. Mathematical model

The problem of exam scheduling, especially in the Toronto dataset, aims to minimize the proximity cost P with the following calculation:

$$P = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C_{ij} W_{|ti-tj|} \quad (1)$$

Where,

$$W_{|ti-tj|} = 2^{5-|tj-ti|}$$

With the following restrictions:

$$\forall_{i \neq j}, ti \neq tj \; if \; C_{ij} > 0$$

In which,
N=number of students.
C_{ij}=number of students taking courses i and j together.
$W_{|ti-tj|}$=weight of the penalty when i and j are scheduled at timeslot t_i and t_j.

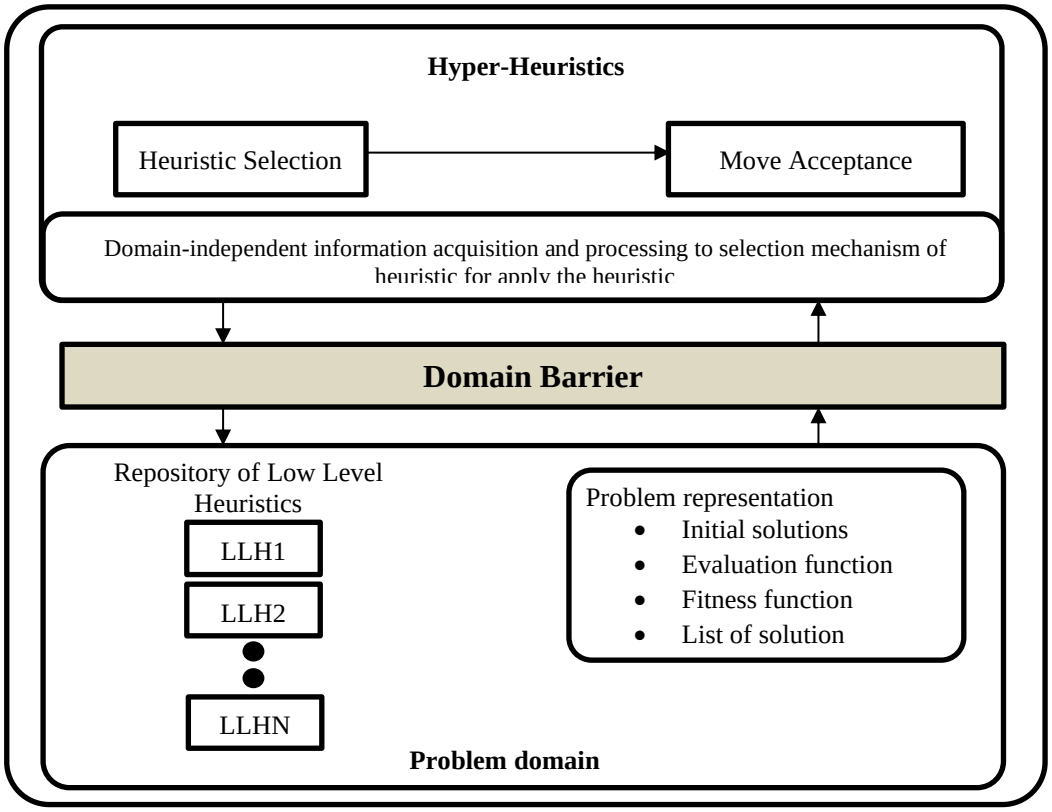


Figure 1. Hyper-heuristic framework

2.5. Algorithm implementation

This process implements an algorithm that will be used at the University of Toronto benchmark datasets. In order to fulfill the objective function, three methods are used, namely the largest degree algorithm, hill climbing, and tabu search. The largest degree algorithm is used to make initial solutions; the results of these algorithms will be evaluated using move acceptance in finding solutions that are better than the initial solution. When the solution received is not feasible, then a penalty will be introduced from the next two algorithms, namely the hill-climbing algorithm and tabu search.

Hyper-heuristic usually has one LLH selection method, with no additional policies and combined with a single-step acceptance approach [14]. This study using two types of LLH, namely "swap" and "move." Swap-move is applied swap operator then followed by the operator of the move. Move-swap implements a displacement operator followed by a swap operator [15]. This case, swap randomly select the exam, then the timeslot from the exam is moved randomly to another timeslot. Meanwhile, the move is made by selecting an exam then moving the exam randomly from the previous timeslot to another timeslot.

2.6. Largest degree algorithm

As a general step, initialization refers to the initial step to generate the initial population. A large number of solutions need to be optimised, but the process that produces viable solutions is difficult for

considerable scheduling problems, so some unworthy solutions become viable in some conventional ways such as the largest degree method of finding initial feasible solutions. The largest degree algorithm is able to use for exam with the largest courses of conflicts than other courses are entered first [16]. This method has the concept of sorting courses based on the number of relationships with other subjects. If the number of courses increases, the opportunity is not scheduled at the same time, and the subject with the highest number of degrees will be scheduled in advance. Pre-scheduled courses minimize or avoid the existence of a timeslot that contains several courses taken by the same student [17].

2.7. Hill climbing algorithm

The hill-climbing algorithm is a well-know algorithm for its simplicity. This algorithm relatively produces worse results than using the metaheuristic. It starts the search for the exploration process from the 'room' section to find highest point [18]. For each iteration from the solution is used to find a new candidate solution which is compared to the current solution and will be accepted if the cost function is not worse [19].

The hill-climbing method is a heuristic search that aims to solve the problem of finding the closest distance. This method works by determining the next step (node), which will show up as close as possible to the target [20]. For optimization, hill climbing requires shuffling, shifting and block swapping [21]. This algorithm uses a move acceptance-based approach finding solutions that are better than the initial solution. In hill-climbing, each iteration will be chosen randomly to be placed randomly on the ticket lot. When the solution received is not feasible, a penalty will be imposed [20]. The pseudocode of hill-climbing is shown as Figure 2.

```
Algorithm Hill Climbing
1:  begin
2:  initialTimeslot ← initial solution
3:  hillClimbingTimeslot ← initialTimeslot
4:  penalty ← initial penalty
5:  hillClimbingTemporaryTimeslot ←initialTimeslot
6:  for i = 0 to iteration do
7:     randomnumber ← random LLH
8:     timeslotLLH ← timeslot from chosen LLH
9:  if penalty LLH < penalty hillClimbingTimeslot
10:    hillClimbingTimeslot ← timeslotLLH
11:    penalty ← new penalty hillClimbingTimeslot
12: else
13:    timeslotLLH ← timeslotHillClimbing
14: end
```

Figure 2. Hill-climbing pseudocode

2.8. Tabu search algorithm

Optimization method that explains local search is search tabu [22]. Tabu search is also an adaptive memory programming method to solve problems in the field of optimization [23]. The selection of the best solution is decided by looking for one solution to the next. The latest quality selection has no obligation to be better than the previous quality. if the latest solution has more benefits and uses than the previous solution, then it can be concluded that the solution is the best [24].

Tabu lists have fundamental algorithms that can prevent searches on previously searched solutions. tabu lists are used to record attributes of some previously applied gestures. it is also used to prevent the search process on the same side [25]. The pseudocode of tabu search is shown as Figure 3.

```
Algorithm Tabu Search
1:  begin
2:  Tabu list TL
3:  currentsolution ← Tabu Search initial solution
4:  bestCandidateSolution ← currentsolution
5:  repeat
6:  currentsolution ← arg min_{x ϵ S(currentsolution)} f(x), where x is not tabu
7:  if currentsolution < bestCandidateSolution then
8:  bestCandidateSolution ← currentsolution
9:  end if
10:    update TL
11:    remove the oldest entry from TL
12: until the stopping criterion is met
13: end
```

Figure 3. Tabu search pseudocode

3. RESULTS AND DISCUSSION

3.1. Testing algorithm implementation

This stage is testing each algorithm in each instance. The testing of these three algorithms was carried out using the Java programming language in NetBeans IDE 8.0, Windows 10 operating system, Intel® Core i7 processor, and 16.0 GB RAM.

3.1.1. Implementation using the largest degree

At this stage, the initial penalty value is implemented when using the largest degree algorithm. The results of the penalty value for using the largest degree algorithm can be seen in Table 2. Based on Table 2, each instance has different fitness and times. If observed based on time, the STA 83 dataset fulfills the constraint with the fastest time of 0.0025727 seconds. STA 83 can run quickly because the number of students and examinations is less compared to other datasets, so it does not require a relatively long running time. It can be seen that the CAR 92 instance is an instance with better performance than other instances. Although based on times, this instance is not the fastest, but this instance can meet hard constraints by not exceeding the timeslot.

Table 2. Largest degree algorithm implementation results

Instance	Timeslot	Fitness	Times (s)
CAR 91	32	10.615	0.0222718
CAR 92	34	11.495	0.0297755
EAR 83	26	72.063	0.0050797
HEC 92	20	32.726	0.005263
KFU 93	20	46.516	0.04371
LSE 91	19	27.048	0.0080651
PUR 93	38	16.701	0.0688106
RYE 92	25	34.183	0.0262566
STA 83	13	194.3	0.0025727
TRE 92	23	15.89	0.0093977
UTA 92	36	7.376	0.0329999
UTE 92	11	54.32	0.0058148
YOR 83	23	64.68	0.0035062

3.1.2. Implementation using the hill climbing

Iterations in 13 instances. From the Table 3, it can be noted that the delta value which states the change in penalty increases from the initial solution. The largest increase in value in the HEC 92 dataset was 66,567%, while the smallest value in the STA 83 dataset was 16,915%. Unfortunately, although the HEC 92 instances has the greatest value, the timeslot of this dataset does not meet the hard constraint. The largest delta value that meets the hard constraint is the KFU 93. If observed based on time, the YOR 83 instance has the fastest running time than other instances with a time of 143,208 seconds.

Table 3. Hill climbing algorithm implementation results

Instance	Timeslot	Fitness	d	Times (s)
CAR 91	32	6.312	40.536%	217.682
CAR 92	34	7.730	32.750%	324.382
EAR 83	26	48.090	33.265%	191.165
HEC 92	20	10.941	66.567%	167.796
KFU 93	20	19.850	57.326%	217.723
LSE 91	19	14.615	45.965%	189.619
PUR 93	38	7.830	53.115%	1983.488
RYE 92	25	11.993	64.913%	236.840
STA 83	13	161.512	16.915%	151.820
TRE 92	23	12.593	20.762%	174.393
UTA 92	36	4.765	35.393%	316.679
UTE 92	11	31.147	42.659%	192.574
YOR 83	23	41.499	35.847%	143.208

From the results of experiments using hill-climbing, states that this algorithm produces a penalty value that is more optimal than the previous penalty value using the largest degree. The hill-climbing algorithm has the concept of accepting a better solution and rejecting a worse solution. If the result of hill-climbing shows the best solution than the initial solution, then the initial solution is replaced by the best solution from hill-climbing.

3.1.3. Implementation using the tabu search

Based on Table 4, the dataset obtained different fitness and time with 10,000 iterations. Each determines the level of goodness of optimization. Based on fitness values, the HEC 92 dataset has the best fitness increase of 58,284%. A dataset with better performance that can be used is HEC 92; this is due to the rise in the value of the best fitness with fast computing time.

Table 4. Tabu search algorithm implementation research

Instance	Timeslot	Fitness Tabu	d	Times
CAR 91	32	7.723	27.248%	19.195
CAR 92	34	8.782	23.597%	25.881
EAR 83	26	49.204	31.7215%	21.168
HEC 92	20	13.652	58.284%	1.889
KFU 93	20	23.029	50.491%	7.078
LSE 91	19	17.555	35.097%	5.926
PUR 93	38	10.875	34.879%	121.284
RYE 92	25	18.276	46.535%	11.365
STA 83	13	158.525	18.452%	1.896
TRE 92	23	12.919	18.709%	6.469
UTA 92	36	5.431	26.364%	56.598
UTE 92	11	31.037	42.861%	2.411
YOR 83	23	46.114	28.712%	4.634

The experiment was carried out using 10,000 iterations on 13 instances using the hill-climbing algorithm and the tabu search algorithm. From Table 5, it can be noted that the tabu search fitness value is better than the hill-climbing fitness value. The delta value (the percentage increase in penalty values from Hill Climbing to Tabu Search) shows that all datasets have a gain or no decrease. The PUR 93 obtains the highest increase in penalty values with a total of 23,939%, while the low rise is in the UTE 92 with 3,127%.

Table 5. Comparison of final results pf both algorithm

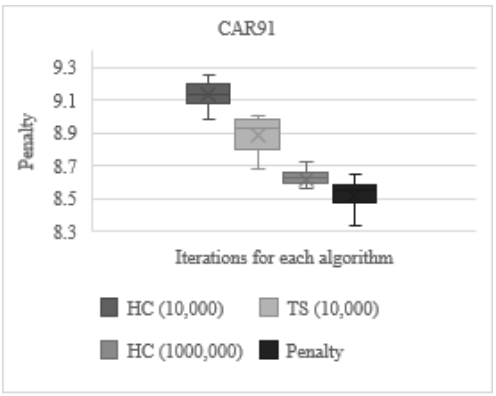
Instance	Timeslot	Fitness HC	Fitness Tabu	d (tabu search-hill climbing)	Times
CAR 91	32	8.711	7.723	9.313%	19.195
CAR 92	34	10.209	8.782	12.416%	25.881
EAR 83	26	56.237	49.204	9.761%	21.168
HEC 92	20	17.658	13.652	12.241%	1.889
KFU 93	20	28.292	23.029	11.314%	7.078
LSE 91	19	20.698	17.555	11.621%	5.926
PUR 93	38	14.873	10.875	23.939%	121.284
RYE 92	25	21.399	18.276	9.138%	11.365
STA 83	13	167.623	158.525	4.680%	1.896
TRE 92	23	14.020	12.919	6.923%	6.469
UTA 92	36	6.382	5.431	12.891%	56.598
UTE 92	11	32.736	31.037	3.127%	2.411
YOR 83	23	52.986	46.114	10.622%	4.634

3.2. Comparing hill climbing algorithm and tabu search algorithm using box and whisker plot

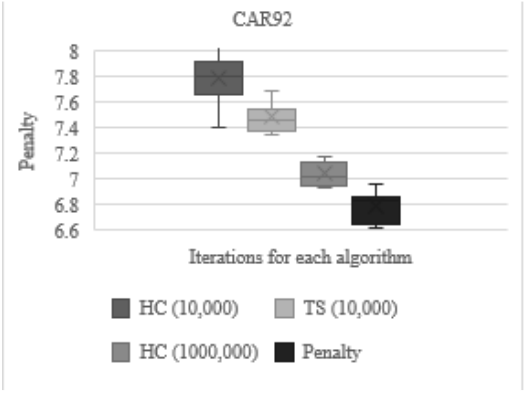
Comparison of the tabu search and hill-climbing algorithms is displayed using a box and whisker plot. Each box has lines at one quartile in the bottom line of the box (Q1), the middle line as lower limit (median-Q1), and top-most line as the upper limit (Q3-median). There are also top whiskers (max-Q3) and bottom whiskers (median-Q1). The experiment runs 11 times in two iteration variations in each algorithm (10,000 iterations and 1 million iterations). The whisker is the above line (referred to as the "top whisker") and below (referred to as the "bottom whisker"). From Figure 5, it can be seen from the five data sets that both algorithms tend to have the best penalty value in the iteration 1 million compared to 10,000. It is also shown that the tabu search (TS) algorithm always has the best penalty value compared to hill-climbing (HC).

3.3 Perfomance comparison on hill climbing and tabu search

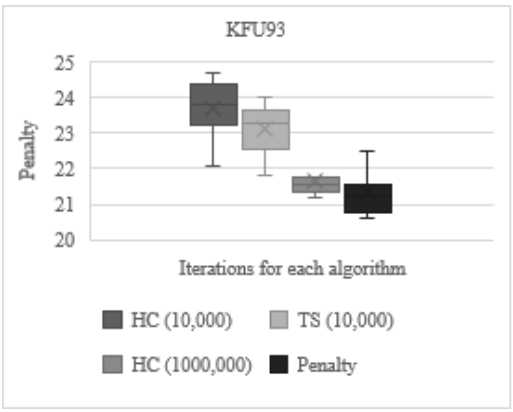
In Figure 6, hill-climbing and tabu search algorithms' performance is run from the iteration steps 10,000 to 1,000,000. From the given line chart, it shows that the 10,000 iterations of the tabu search algorithm are far superior and have lower penalty values than hill-climbing.



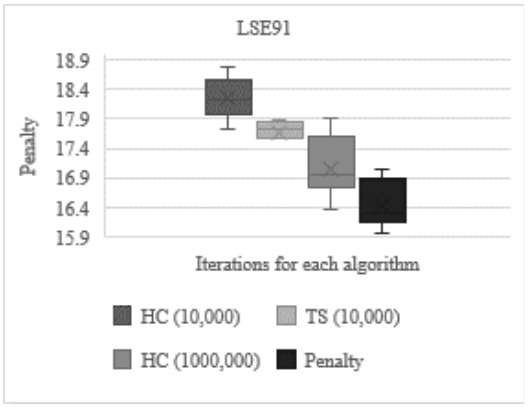
(a)



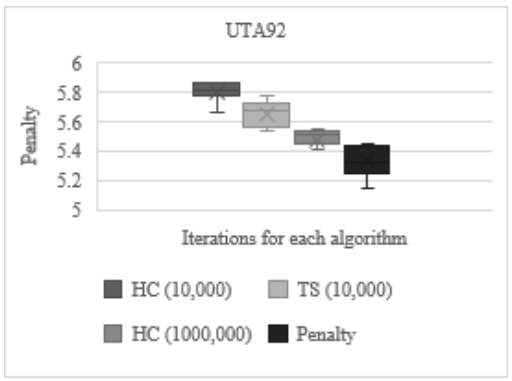
(b)



(c)



(d)



(e)

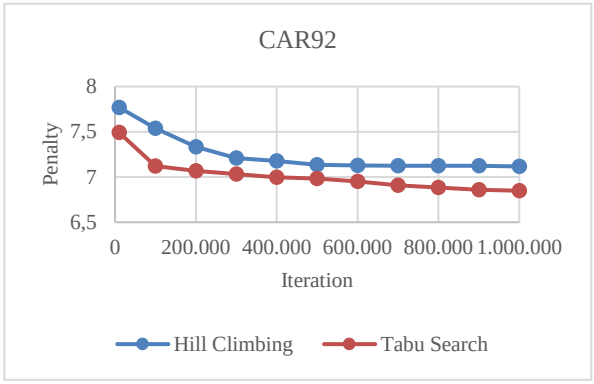
Figure 5. Box and whisker plots of hill climbing and tabu search algorithms on carter's data set (Toronto), (a) Car91, (b) Car92, (c) Kfu93, (d) Lse91, (e) Uta92

3.4 Comparison of results with other published results

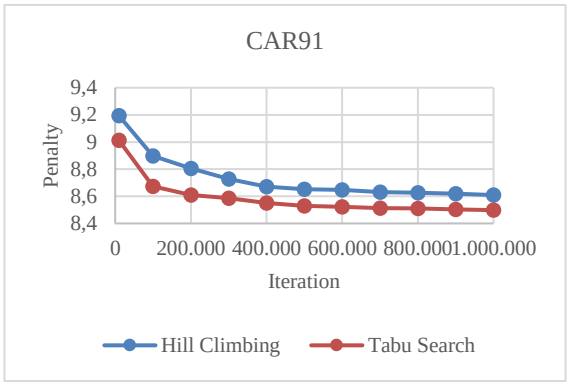
The performance of the tabu search algorithm was carried out ten times, running at 1,000,000 iterations. The results of the experiment are the average of each instance. In Table 6, the best value of the trial results is displayed and compared with some previous studies on the problems of the Toronto dataset.

The results of this trial were compared with several previous studies, namely, K. Graham and M. Naimah [3] who used tabu search hyper-heuristic. Gaspero and Schaerf [4] using the tabu search algorithm, Carter *et al.* [26] use constructive heuristics with backtracking, Caramia *et al.* [5] use greedy constructive heuristics, Burke and Newall [6] use a local search method, I. Gabriella & P. Etria use tabu search algorithm with 100,000 iterations [7].

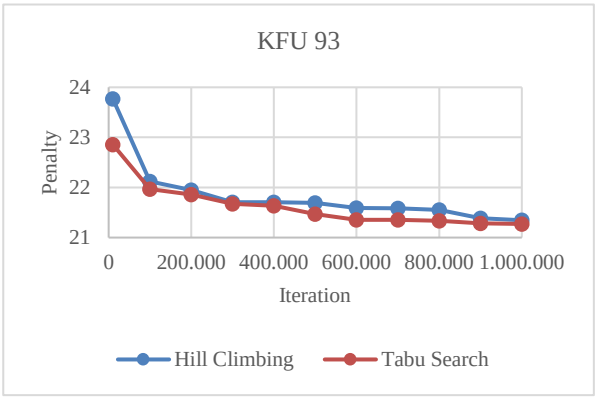
The purpose of this experimental results is to show that the tabu search algorithm is able to produce good penalty results even though the value produced is not the smallest. The results show that there is one instance that provides the best value, STA 83, compare to previous studies.



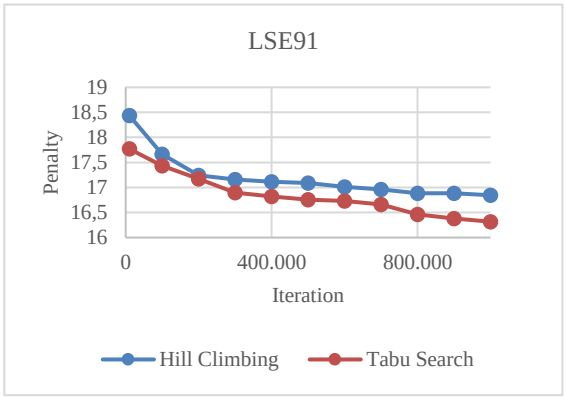
(a)



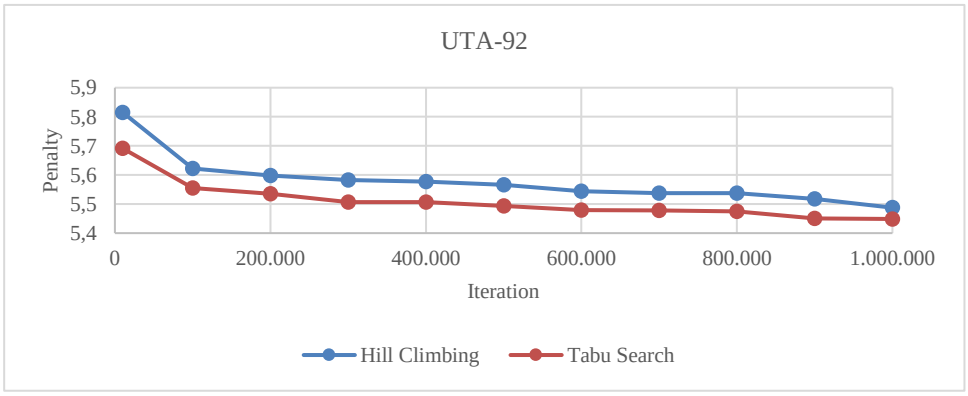
(b)



(c)



(d)



(e)

Figure 6. Performance comparison of both algorithms, (a) Car92, (b) Car91, (c) Kfu93, (d) Lse91, (e) Uta92

Table 6. Comparison of final results of both algorithm

Instance	Tabu Search (1000,000 iteration)	K. Graham and M. Naimah [3]	Di Gaspero and Schaerf [4]	Caramia et al [5]	Burke & Newall [6]	I. Gabriella & P. Etria [7]	Carter *et al*. [26]
CAR 91	8.52	5.37	6.2	6.6	4.6	8.7	7.1–7.9
CAR 92	6.79	4.67	5.2	6.0	4.1	7.1	6.2–7.6
EAR 83	44.48	40.18	45.7	29.3	37.05	44.9	36.4-46.5
HEC92	12.5	11.86	12.4	9.2	11.54	12.5	10.8-15.9
KFU 93	21.33	15.84	18.0	13.8	13.9	21.8	14.0-22.1
LSE91	16.45	13.67	15.5	10.5	10.82	17.4	10.5-13.1
PUR93	10.5	6.06	-	-	-	-	3.9-5.0
RYE92	15.51	-	-	-	-	-	7.3-10.0
STA 83	152.33	157.38	160.8	158.2	168.73	152.5	161.5-165.7
TRE 92	11.26	8.39	10.0	9.4	8.35	11.8	9.6-11.0
UTA92	5.33	3.92	4.2	3.5	3.2	5.5	3.5-5.3
UTE92	30.18	27.60	29.0	24.4	25.83	33.7	25.8-38.3
YOR83	42.01	39.42	41.0	36.2	37.28	41.9	41.7-49.9

4. CONCLUSION

Based on the results of trials and analysis of the results that have been carried out, it can be concluded that the proposed method produces a fitness value that is much smaller than that of the constructive heuristic. The hill-climbing algorithm can reduce the number of timeslots in several datasets that were not initially feasible as a feasible solution. Furthermore, optimization using the tabu search algorithm results in better performance by giving a smaller penalty or fitness value to Toronto datasets than the hill-climbing algorithm. The difference is 18-58%. Compare to the previous studies; this study result is not the worst, which shows the potential of the proposed method if further investigated.

REFERENCES

[1] L. Eugene L, "Combinatorial Optimization: Networks and Matroids," *Holt Rinehart and Winston*, 1976.

[2] V. Supoyo and A. Muklason, " Hyper Heuristic Approach with Algorithm Combination in Examination Timetabling Problem (in Indonesia: Pendekatan Hyper Heuristic dengan Kombinasi Algoritma pada Examination Timetabling)," *ILKOM Jurnal Ilmiah*, vol. 11, no. 1, pp. 34-44, 2019.

[3] A. Muklason, " Solver for Automatic Exam Scheduler with Maximal Clique Algorithm and Hyper-heuristics (in Indonesia Solver Penjadwal Ujian Otomatis Dengan Algoritma Maximal Clique dan Hyper-heuristics)," *in Seminar Nasional Teknologi Informasi, Komunikasi and Industri*, vol. 9, pp. 94-101, 2017.

[4] N. M. Hussin, "Tabu Search Based Hyper-Heuristic Approaches to Examination Timetabling," thesis, University of Nottingham, pp. 1-230, 2005.

[5] L. Di Gaspero and A. Schaerf, "A Tabu Search Techniques for Examination Timetabling," *The Practice and Theory of Automated Timetabling III (PATAT 2000), Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, vol. 2079, pp. 104-117, 2001.

[6] M. Caramia, P. Dell'Olmo and G. Italiano, "New Algorithms for Examination Timetabling," *Algorithm Engineering 4th International Workshop*, Springer, Berlin, Heidelberg, pp. 230-241, 2000.

[7] E. Burke and J. Newall, "Enhancing Timetable Solutions with Local Search Methods," *Practice and Theory of Automated Timetabling IV*, Springer, Berlin, Heidelberg, pp. 195-206, 2003.

[8] G. Icasia, R. Tyasnurita, and E. S. Purba, "Application of Heuristic Combination in Hyper-Heuristic Framework for Exam Scheduling Problems," vol. 4, no. 4, pp. 8, 2020.

[9] H. Lanny. (2017). *Bab 2 Optimisasi Kombinatorial [Chapter 2 Combinatorial Optimization]* [Online]. Available: Https://Docplayer.Info/38509228Bab-2-Optimisasi-Kombinatorial.Html. [Accessed: 08-May-2020].

[10] "Benchmark Exam Timetabling Datasets", Asap.cs.nott.ac.uk, 2020. [Online]. Available: http://www.asap.cs.nott.ac.uk/external/resources/. [Accessed: 22- May- 2020].

[11] D. Kusumawardani, A. Muklason and V. A. Supoyo, "Examination Timetabling Automation and Optimization using Greedy-Simulated Annealing Hyper-heuristics Algorithm," *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, Surabaya, Indonesia, 2019, pp. 1-6.

[12] D. Peter, B. Bilgin, P.D Causmaecker and G.V Berghe, "A Hyperheuristic Approach to Examination Timetabling Problems: Benchmarks and a New Problem from Practice," *Article In Journal Of Scheduling*, 2012.

[13] E. Özcan, M. Misir, G. Ochoa and E. Burke, "A Reinforcement Learning - Great-Deluge Hyper-Heuristic for Examination Timetabling," *International Journal of Applied Metaheuristic Computing*, vol. 1, no. 1, pp. 39-59, 2010.

[14] E. Burke, G. Kendall and E. Soubeiga, "A Tabu-Search Hyperheuristic for Timetabling and Rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451-470, 2003.

[15] V. A. de Santiago Júnior, E. Özcan, and V. R. de Carvalho, "Hyper-Heuristics based on Reinforcement Learning, Balanced Heuristic Selection and Group Decision Acceptance," *Applied Soft Computing*, vol. 97, part A., Dec. 2020, doi: 10.1016/j.asoc.2020.106760.

[16] A. Turky, N. R. Sabar, S. Dunstall, and A. Song, "Hyper-heuristic local search for combinatorial optimisation problems," *Knowledge-Based Systems*, vol. 205, Oct. 2020, doi: 10.1016/j.knosys.2020.106264.
[17] Y. Lei and J. Shi, "A NNIA Scheme for Timetabling Problems", *Journal of Optimization*, vol. 2017, pp. 1-11, 2017.
[18] D. H. Al-Omari and K. E. Sabri, "New Graph Coloring Algorithms," *American Journal of Mathematics and Statistics*, vol. 2, no. 4, pp. 439–441, 2006.
[19] M. R. Gholami Dehbalaeea, G. H. Shaeisia, and M. Valizadeh, "A Proposed Improved Hybrid Hill Climbing Algorithm with the Capability of Local Search for Solving the Nonlinear Economic Load Dispatch Problem," *International Journal of Engineering*, vol. 33, pp. 575-585, 2020.
[20] E. K. Burke and Y. Bykov, "A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems," *Proceedings of the Conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*, pp. 1-7, 2008.
[21] R. Siti Khatijah, B. Andrzej, Q. Rong, "Hill Climbing versus Genetic Algorithm Optimization in Solving the Examination Timetabling Problem," *Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems*, pp. 43-52, 2013.
[22] E. Dangkua, V. Gunawan and K. Adi, " Application of the Hill Climbing Method in Geographic Information Systems to Find the Shortest Path (in Indonesia Penerapan Metode Hill Climbing Pada Sistem Informasi Geografis Untuk Mencari Lintasan Terpendek)," *Jurnal Sistem Informasi Bisnis*, vol. 5, no. 1, 2015.
[23] A. K. Mandal and M. N. M. Kahar, "Peformance Analysis of Graph Heuristics and Selected Trajectory Metaheuristics," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 8, no. 1, pp. 163-177, March 2020.
[24] Fred Glover, Manuel Laguna, "Tabu search," Handbook of Combinatorial Optimization, Boston: Kluwer Academic Publishers, pp. 2093-2229, 1998.
[25] S. Leo Candra, "Application of the Tabu Search Algorithm for Scheduling Subjects at Pelita-2 Private Vocational Schools Aekkanopan (in Indonesia Penerapan Algoritma Tabu Search untuk Penjadwalan Mata Pelajaran di SMK Swasta Pelita-2 Aekkanopan)," *Jurnal Riset Komputer (JURIKOM)*, vol. 3 no. 6, pp. 74-79, 2016.
[26] N. Graham Kendall, "An Investigation of a Tabu Search Based Hyper-heuristic for Examination Timetabling," *Citeseerx.ist.psu.edu*, 2020. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.2082. [Accessed: 29- Jun- 2020].
[27] M. Carter, G. Laporte and S. Lee, "Examination Timetabling: Algorithmic Strategies and Applications," *The Journal of the Operational Research Society*, vol. 47, no. 3, pp. 373- 383, 1996.

BIOGRAPHIES OF AUTHORS

Shinta Dewi currently pursuing her bachelor's degree at Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. She is actively participated in organizational activities and held the position of secretary in the Information Media Department at the Information Systems Student Association (HMSI) for period 2019-2020. She is currently working on her thesis on the topic of optimization with the problem of optimizing the route using a hyperheuristic algorithm.

Raras Tyasnurita received her bachelor's degree in computing from Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia, and the MBA degree from National Taiwan University of Science and Technology, Taipei, Taiwan, in 2008 and 2012, respectively. She joined Information Technology Application and Integration Laboratory in her Master study. She is a PhD candidate in Information Technology, University of Nottingham., United Kingdom. She is currently a lecturer within the Data Engineering and Business Intelligence research group in Information Systems Department, Institut Teknologi Sepuluh Nopember. She was an Editor in Chief of SISFO Journal (2009-2010), SESINDO 2013 proceeding as a national conference in Information Systems, and International Seminar on Science and Technology (ISST) 2019. Her main research interests are Artificial Intelligence, Combinatorial Optimisation, Machine Learning, and Forecasting.

Febriyora Surya Pratiwi currently in hers 2017 degree in Information Systems at Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Outside of college he found the position of editor-in-chief of the Information Systems Student Association (HMSI) organization of the Information Media (IM) department for the period 2019-2020. She is currently compiling a thesis on the topic of digital transformation research in the field of pandemic education by looking at the sustainability of the adoption of technology in an institution or organization.