Date of publication JUN-30, 2024, date of current version Jan-18, 2024.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.23.2.3549

Adaptive Learning Modified Great Deluge Hyper-Heuristics

RIZAL RISNANDA HUTAMA, AHMAD MUKLASON
Institut Teknologi Sepuluh Nopember, Surabaya, 60111 Indonesia

Corresponding author: Rizal Risnanda Hutama (e-mail: r.risnanda@its.ac.id)

ABSTRACT The International Timetabling Competition (ITC) 2021 focuses on sports scheduling, a domain intricately connected to optimizing combinatorics problems. Within the framework of the ITC 2021 challenge, a crucial task is to precisely allocate matches to their designated time slots. Addressing this challenge involves the utilization of the Adaptive Learning Modified Great Deluge (ALMGD) algorithm, which belongs to the realm of hyper-heuristics. This algorithm represents an evolutionary step from the foundational great deluge algorithm, incorporating an acceptance mechanism intricately woven with self-adaptive learning. To assess its efficacy, the performance of the ALMGD algorithm is scrutinized through a comparative analysis with the hill climbing and great deluge algorithms. As a result, the proposed algorithm can produce a solution that is superior to the comparison algorithm. The modified great deluge algorithm can reduce the penalty by 36%, while the hill climbing algorithm can only reduce the penalty by 29% and the great deluge algorithm reaches 34%.

KEYWORDS Combinatorial Optimization; Sport Scheduling; Adaptive Learning Modified Great Deluge; Hyper-Heuristics.

I. INTRODUCTION

IN recent years, the field of optimization of combinatorial problems has often discussed issues related to sports scheduling [1]. Several factors cause sports scheduling to be an interesting discussion [2]. The ability of an algorithm to solve sports scheduling that has a large enough number of teams is one of these factors. With a fairly large number of teams, it is challenging for researchers to develop an algorithm that can solve them better and in less time.

From a computational point of view, sports scheduling optimization is included in the category of NP-Hard problems [3]. So, it can be interpreted that the optimization of sports scheduling cannot be solved using an exact algorithm with polynomial time. However, it can be solved using heuristics algorithm with relatively fast time and produce a fairly good solution close to optimal [4]. The solution generated using the heuristics algorithm does not guarantee to be an optimal solution. However, the solution can be said to be quite good and close to optimal. The optimal solution is measured based on the objective function. Until now, sports scheduling has quite diverse objective functions [5], for example, minimizing soft constraint violations, minimizing match costs, minimizing match distances and some others.

Research related to sports scheduling in the field of operations research or optimization has been carried out quite a lot. Examples of sports that have been conducted in scheduling research are football [6, 7], table tennis [8], basketball [9–11], and so on. Algorithms that have been used to solve sports scheduling optimization are adaptive large neighborhood search [5], a combination of greedy randomized adaptive search and iterated local search [12], harmony search [13] and so on. Sports scheduling often appears with a round robin model [14].

ITC is a competition related to scheduling optimization. In previous years the ITC held a competition for optimization of university exam scheduling. Meanwhile, ITC 2021 is currently hosting a competition for sports scheduling for the first time [15]. In the case of ITC 2021, there is an objective function to minimize soft constraint violations. So, that it can be interpreted that the best solution is the solution that has the least soft constraint violation. No regulatory algorithm is used to solve ITC 2021 but it must be the same for each instance provided.

In this research, we propose an enhanced algorithm named adaptive learning modified great deluge (ALMGD). The algorithm is a combination of two algorithms, namely self-adaptive learning and modified great deluge. In this research, a self-adaptive learning algorithm is used to determine the low-level heuristic used in each iteration or what is known as LLH selection. The modified great deluge algorithm is a development of the great deluge algorithm which is used to determine the acceptance of the candidate solution in each

iteration or what is called move acceptance. The application of the combination of the two algorithms is carried out based on a hyper-heuristics framework. In the modified great deluge algorithm, modifications will be made to the mechanism for accepting candidate solutions. The ALMGD algorithm is expected to produce the most optimal solution compared to the comparison algorithm.

II. LITERATURE REVIEW

In this section, we will explain some of the theoretical bases used in this research. The theoretical basis that will be discussed in this section is related to the problem and the algorithm used. The theoretical basis is used as a reference from previous research.

A. INTERNATIONAL TIMETABLING COMPETITION 2021

International Timetabling Competition (ITC) 2021 is a sports scheduling competition with a double round robin system [15]. Double round robin is a sport system in which each team competes 2 times with the same opposing team. So, it can be said that there are 2 stages of the match. The objective function of ITC 2021 is to minimize soft constraint violations. The mathematical model of the 2021 ITC problem was listed in RobinX's research [16]. However, the mathematical model has adjustments. ITC 2021 has 3 types of data sizes based on the number of slots and teams in an instance. The size of the data can be seen in Table 1. In addition, ITC 2021 is a time-constrained problem. So, all teams must compete in each slot. Illustration of the ITC 2021 match schedule can be seen in Fig. 1.

Table 1. Symbols

Data Size	Number of Teams	Number of Slots
Small	16	30
Medium	18	34
Big	20	38

Slot1	Slot2	Slot3	Slot4	Slot5	Slot6	Slot7	Slot8	Slot9	Slot10
(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)
(3,5)	(2,4)	(2,5)	(2,6)	(2,3)	(5,3)	(2,4)	(5,2)	(6,2)	(3,2)
(4,6)	(5,6)	(3,6)	(3,4)	(4,5)	(6,4)	(6,5)	(6,3)	(4,3)	(5,4)

Figure 1. ITC 2021 Schedule Illustration

ITC 2021 is the newest ITC. In previous years, ITC has also been held. However, in previous years, the ITC held a scheduling competition that focused on scheduling exams and courses. An example of previous research that has been carried out is at ITC 2002 using a simulated annealing algorithm to complete course scheduling [17]. Furthermore, at ITC 2007, still in the same field, a combination of hill climbing algorithm with great deluge and simulated annealing was used [18]. At ITC 2011 a combination of iterated local search and simulated annealing algorithms was used to solve the problem of scheduling high school course [19].

B. GREAT DELUGE ALGORITHM

The great deluge algorithm is an algorithm inspired by flooding. By analogy, a higher point will be searched for to avoid flooding. According to this analogy, the mechanism for accepting candidate solution must be equal to or higher than the water level for the maximization problem [20]. And vice versa, if the problem is minimization, the acceptance of the candidate solution must be lower or equal to the water level.

Several previous studies also modified the great deluge algorithm. The first example is modification of the acceptance of the candidate solution with a flexible coefficient to increase flexibility [21]. In addition, modifications to the great deluge algorithm were also carried out on the low level heuristic (LLH) selection mechanism and water level reduction [22]. In this study, the great deluge algorithm is modified in the mechanism for accepting candidate solutions. The modification is done by adding a late acceptance strategy [23] to get better results.

C. SELF ADAPTIVE LEARNING STRATEGY

With the number of LLH more than 1, a mechanism is needed to select LLH in each optimization iteration. The self-adaptive learning algorithm is chosen as the mechanism. Self-adaptive learning has 2 components, namely the neighborhood list (NL) and the winning neighborhood list (WNL). NL is useful for storing LLH which is used in each iteration while WNL is used to store LLH which has good performance. NL filling is obtained from the contents of WNL. So, it can be said that if LLH has good performance, in the next iteration it will have a greater chance of being re-selected. Several previous studies implemented this algorithm and showed great results. An example is combining it with particle swarm optimization algorithm [24] and combining it with artificial bee colony [25].

D. HYPER-HEURISTICS

Hyper-heuristics is a framework or approach in optimization to solve problems that cannot be solved manually by selecting and running a simpler heuristic (heuristic selection) or building a new heuristic (heuristic generation) [26]. Hyper-heuristics do not deal directly with the problem but through lower heuristics [27] or low level heuristics (LLH) [28]. So, in this study a hyper-heuristics approach was used to select LLH.

III. METHODOLOGY

In this section, the flow of the research carried out is explained. The flow of the research carried out is based on Fig. 2. Due to this figure, process search begins with constructing the initial solution, then tuning parameters, and finally carrying out the optimization process.

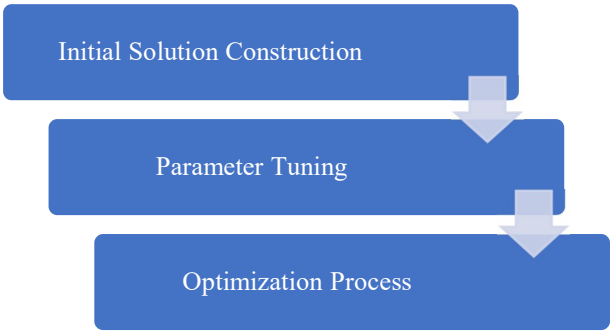

Figure 2. Research Methodology

A. INITIAL SOLUTION CONSTRUCTION

Making the initial solution is a stage to produce a feasible solution. A solution can be said to be feasible if the solution does not violate the hard constraints. If a solution is feasible, then the solution can be continued at the optimization stage. Vice versa, if it is not feasible then it cannot be continued at the optimization stage. Constructing a feasible initial solution is

done by using the tabu late acceptance hill climbing algorithm. The algorithm was chosen because it is easy to implement. This means that it only has 1 parameter, which is only the length of the fitness array. In addition, the algorithm does not have a time limit to run like what happened to the great deluge algorithm. So, this algorithm will run for a maximum of 6 hours for each instance. If up to 6 hours an instance still has a hard constraint violation, it is said that the instance is not feasible.

B. PARAMETER TUNING

The proposed modified great deluge algorithm has 2 parameters that must be filled in and tuned. These parameters are the length of the fitness array and the desired value. Parameter tuning aims to determine the appropriate parameter values to obtain the best solution. If the tuning parameters are not carried out, the results obtained can be even worse than the hill climbing algorithm. Not all instances have parameter tuning done. Parameter tuning is only performed on example instances. The example instance used for the tuning parameter is an instance that represents the size of the data. The selection of instances used for tuning parameters is based on the fastest optimization duration for each data size with the same number of iterations. The best tuning parameter values obtained will be applied to all another instances.

C. OPTIMIZATION PROCESS

The optimization results obtained will be analyzed. This analysis aims to determine the performance of the ALMGD algorithm when compared to other algorithms. The first analysis compares the average optimization value of all algorithms in 10 repetitions. The analysis aims to determine the comparison of the average optimization results in general. Furthermore, an analysis related to the comparison of the percentage reduction in penalties and running duration is carried out. With this comparison, it will be known whether the algorithm modification for improving the penalty reduction is commensurate with the running duration. Then an analysis of the consistency of the optimization results is carried out. Consistency analysis is carried out using a boxplot graph. The more consistent they are, the better the optimization results. The last part is to analyze the penalty reduction in each iteration. The analysis was carried out using a trajectory graph. With the trajectory graph, the pattern and speed of penalty reduction that causes differences in the final optimization result are revealed. With the overall analysis that has been done, it is clear which algorithm has the best performance.

IV. RESULTS AND DISCUSSION

In this section, we explain the results of the experiments that are carried out. The experiment is carried out on devices with Intel Core i5-6200u specifications and 8 GB RAM capacity. Each experiment is carried out with 100,000 iterations and 10 repetitions.

A. INITIAL SOLUTION CONSTRUCTION

The initial solution is constructed by optimizing the hard constraint violation. At first, the resulting solution is a solution that is still free from the base constraint violation of the ITC 2021 problem. However, the solution still violates the hard constraint. A solution can be said to be feasible if there is no violation of the hard constraint. To eliminate the hard constraint violation, optimization is carried out using the tabu late acceptance hill climbing algorithm based on previous research that was done in [29]. The results of the construction of the initial solution are shown in Table 2.

Due to the table, 24 out of 54 instances (44.44%) are feasible. The most feasible data package is the Late data package with a total of 8 data. The Late data package will then be used as input for the parameter tuning process.

Table 2. Result of Initial Feasible Solution

Instance	Result	Instance	Result
Test Ins. Demo	Feasible	Middle4	Not feasible
Test1	Feasible	Middle5	Feasible
Test2	Feasible	Middle6	Not feasible
Test3	Feasible	Middle7	Not feasible
Test4	Not feasible	Middle8	Feasible
Test5	Feasible	Middle9	Feasible
Test6	Not feasible	Middle10	Not feasible
Test7	Not feasible	Middle11	Not feasible
Test8	Not feasible	Middle12	Feasible
Early1	Feasible	Middle13	Not feasible
Early2	Not feasible	Middle14	Not feasible
Early3	Feasible	Middle15	Feasible
Early4	Not feasible	Late1	Not feasible
Early5	Not feasible	Late2	Not feasible
Early6	Not feasible	Late3	Feasible
Early7	Not feasible	Late4	Feasible
Early8	Feasible	Late5	Not feasible
Early9	Feasible	Late6	Not feasible
Early10	Not feasible	Late7	Feasible
Early11	Not feasible	Late8	Feasible
Early12	Not feasible	Late9	Feasible
Early13	Feasible	Late10	Not feasible
Early14	Feasible	Late11	Feasible
Early15	Not feasible	Late12	Not feasible
Middle1	Not feasible	Late13	Not feasible
Middle2	Not feasible	Late14	Feasible
Middle3	Not feasible	Late15	Feasible

B. INITIAL SOLUTION CONSTRUCTION

The use of the ALMGD algorithm has parameters that must be tuned. Parameters that are tuned in this study are the values for the length of the fitness array and the desired value. The length of the fitness array needs to be tuned because it is adjusted to the number of iterations carried out. If the length of the fitness array does not match the number of iterations used, the optimization results will be of lower value while the desired value must be tuned because it is related to the decay rate at the water level. If the decay rate is higher, the acceptance limit for worse solutions will decrease faster. This resulted in the trapping of the local optima.

The values of fitness array to be experimented with are 5, 10, and 15. This value is obtained from the trial process starting from 500 and decreasing to get a better solution until it gets a value of 10. From a value of 10, a tolerance value of 5 is given so that there are values of 5 and 15. Meanwhile, the desired value to be experimented with is 0 and corresponds to the best-known solution (symbolized by d). The value 0 is the most optimal optimization value. Meanwhile, the best known-solution value is the best optimization result that has been obtained from another research. The instances used for the parameter tuning experiment are Late3, Late8 and Late11. The three instances represent their respective data sizes with the fastest optimization duration.

The experimental results of the tuning parameters are shown in Table 3. In the table the best parameter values and compared with hill climbing and great deluges based on average result, are obtained using the fitness array value of 10 and the desired value according to the known best solution. So, the optimization process that is carried out on all instances will use this value to produce a good solution.

Table 3. Results of Parameter Tuning

		HC	GD	ALMGD					
				5, 0	10, 0	15, 0	5, d	10, d	15, d
Late3	Max	14699	14559	14449	14644	14319	14394	14579	14504
	Min	14069	12610	12759	12519	11959	12789	12389	12085
	Avg.	14396.5	13952.2	13920.9	13855.6	13731	13926	**13602**	13659.6
Late8	Max	4409	4244	3883	4287	3735	4244	3768	3989
	Min	3608	3170	3215	3302	3132	3301	3217	2381
	Avg.	4016.3	3623.1	3486.9	3588.5	3418.3	3658	**3391.2**	3397.6
Late11	Max	1001	1066	1026	936	1096	945	941	1036
	Min	856	781	796	761	716	731	745	761
	Avg.	928.5	927.9	910	875.4	882.5	875.5	**837.9**	865

C. OPTIMIZATION PROCESS

The proposed algorithm for optimization is adaptive learning modified great deluge (ALMGD). Adaptive learning is a strategy used to choose LLH which is applied in each iteration. There are 4 types of LLH used in this study. 3 LLH is from previous research [30], and 1 LLH is its improvement while the modified great deluge is a great deluge algorithm that is modified as a move acceptance to get better results. The modification made is by adding a move acceptance strategy. If at the time of optimization, the water level drops lower than the penalty, then the mechanism for accepting candidate solutions is only hill climbing or can only accept better solutions. It is prone to being trapped in the local optima. So, we need a mechanism for accepting candidates for water level replacement. Then a modification is made by adding a late acceptance strategy as a substitute for water level. The complete pseudocode of the proposed algorithm is shown in Fig. 3.

```
1: procedure Adaptive Learning Modified Great Deluge
2:     current ← initial solution
3:     best ← initial solution
4:     i ← iteration length
5:     w ← initial waterlevel or initial solution
6:     dv ← desired value
7:     d ← decayrate or ((w-dv)/i)
8:     f ← fitness length
9:     for all f ← initial solution
10:    repeat
11:    for i = 1 to l do
12:            generateCandidate using SelfAdaptiveLearning
               mechanism
13:            if p(candidate) <= p(current) or
14:              p(candidate) <= w or
15:              p(candidate) <= p(f % i) then
16:                    accept the candidate
17:                    if current < best
18:                            best = current
19:            else
20:                    reject the candidate
21:            update w ← (w-d)
22:    end for
23:    save best sol
24: end procedure
```

Figure 3. Pseudocode of Adaptive Learning Modified Great Deluge

The optimization results using ALMGD are compared with the hill climbing and great deluge algorithms. Comparison with hill climbing is done because hill climbing is the basic algorithm of optimization. While the comparison of great deluge is done because great deluge is the original algorithm before modifications are made. The results of the comparison of the ALMGD algorithm with the two algorithms are shown in Table 4. Due to the table it is seen that the optimization results when using the ALMGD algorithm are better than the two comparison algorithms on all optimized instances. This is caused by the trapping of the hill climbing algorithm on the local optima and the decrease in the water level which is lower than the penalty on the great deluge algorithm.

Table 4. Comparison of Optimization Result

Instance	Average Penalty		
	HC	GD	ALMGD
TestInstanceDemo	**0**	**0**	**0**
Test1	**1120**	**1120**	**1120**
Test2	189.7	177.9	**177.2**
Test3	**1253**	**1253**	**1253**
Test5	241.2	181.1	**178.6**
Early1	2756.7	2681.9	**2614.4**
Early3	2909.9	2840.1	**2745.4**
Early8	3143.3	2971.5	**2953.8**
Early9	2216.9	1648.5	**1479.6**
Early13	1644.8	1496.6	**1474.5**
Early14	1922.9	873.3	**855.3**
Middle5	2331.6	2001	**1863.9**
Middle8	1368.7	1231.5	**1228.3**
Middle9	2216.9	2168	**2144**
Middle12	5109.7	4994.5	**4856.6**
Middle15	3360.9	2645.5	**2523.3**
Late3	14396.5	13952.2	**13602**
Late4	625.4	473.6	**427.5**
Late7	16932.4	16932.7	**16711.7**
Late8	4016.3	3623.1	**3391.2**
Late9	2308.3	2086.1	**2012.4**
Late11	928.5	927.9	**837.9**
Late14	3375.6	3298.4	**3282.2**
Late15	1314	820	**801**

To make it easier to measure the performance of the ALMGD algorithm, a comparison is made against the percentage of penalties reduction. The percentage value of penalty reduction is obtained from the comparison of the optimization value with the penalty value of the initial solution. Comparison is also equipped with duration to determine the effect of the running duration. The comparison is shown in Table 5.

Table 5 shows that the best penalty reduction is obtained by the ALMGD algorithm by 36%. The penalty reduction using ALMGD has a significant difference when compared to hill climbing. Meanwhile, when compared to the great deluge it only has a difference of 2%. That number may seem small, but if the penalty is worth thousands, it will be significant. Especially with the comparison of the duration of time required between the great deluge algorithm and the adaptive learning modified great deluge algorithm, there is a time difference of 11.5 seconds. The difference in duration when converted to a percentage is less than 1% when compared to the duration of the great deluge duration. So, the 2% penalty difference is better because it takes the same time.

Table 5. Percentage of Penalties Reduction and Running Duration Comparison

Instance	HC	Duration (s)	GD	Duration (s)	ALMGD	Duration (s)
TestInstanceDemo	0%	0	0%	0	0%	0
Test1	2%	10	2%	12	2%	12
Test2	27%	4	31%	5	32%	5
Test3	0%	0	0%	0	0%	0
Test5	22%	20	41%	175	49%	178
Early1	2%	161	5%	2690	11%	2698
Early3	45%	308	47%	2766	52%	2770
Early8	41%	588	44%	2972	47%	2994
Early9	63%	423	72%	1501	77%	1540
Early13	9%	1056	18%	1499	27%	1520
Early14	63%	708	83%	870	87%	876
Middle5	40%	168	49%	1840	58%	1897
Middle8	10%	1379	19%	1237	23%	1241
Middle9	27%	1397	28%	2140	34%	2167
Middle12	12%	1009	14%	4933	26%	4940
Middle15	71%	241	77%	2410	83%	2415
Late3	6%	677	9%	659	19%	671
Late4	69%	37	76%	75	91%	78
Late7	0%	753	0%	805	4%	810
Late8	21%	150	28%	170	37%	177
Late9	28%	1629	35%	1933	42%	1940
Late11	44%	213	44%	279	55%	284
Late14	16%	1290	18%	2974	21%	2997
Late15	80%	789	87%	1224	89%	1235
Average	**29%**	**542.1**	**34%**	**1382**	**36%**	**1393.5**

Then an analysis is carried out related to the consistency of the optimization results for each algorithm. The more consistent the optimization value of the algorithm, the better the optimization results. Measurement of the consistency of results is carried out using box plots for all algorithms. The box plot results are shown in Fig. 4.

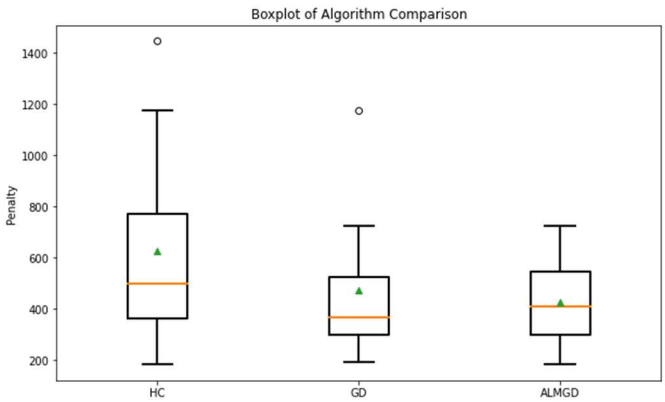

Figure 4. Boxplot of Algorithm Comparison

According to the results displayed on the boxplot, the penalty value of 10 repetitions using the ALMGD algorithm has more consistent results because there are no outliers like in hill climbing and great deluge. Outliers in Fig. 4 are shown by circle marks. In addition, it is also known that the average result of the ALMGD algorithm is better than the two comparison algorithms which are displayed by a triangle sign.

Next is the analysis of the penalty reduction that occurs in the three algorithms. The analysis is carried out using 100 data for every 1000 iterations. The analysis is presented in a trajectory graph. An example of a trajectory graph that occurs in the Middle5 instance is shown in Fig. 5.

In Fig. 5 the movement of the penalty reduction that occurs when using the hill climbing algorithm is very quickly reduced at the beginning of the iteration. This is because the hill climbing algorithm only accepts better solutions. The only acceptance of hill climbing resulted in encountering a local optimum when the initial iteration was less than 10.000. Which means the next iteration does not find a better solution. Meanwhile, the great deluge and ALMGD algorithms tend to

experience a slow decline, even up and down penalties. However, in the great deluge algorithm iterations approaching 60,000 also experience local optima. This may be caused by the penalty value that has not decreased while the water level value is already below the penalty value. So, the next iteration only accepts a better solution and causes it to be stuck in the local optima. The ALMGD algorithm has the best result from the comparison algorithm because it has an additional candidate acceptance mechanism to avoid local optima.

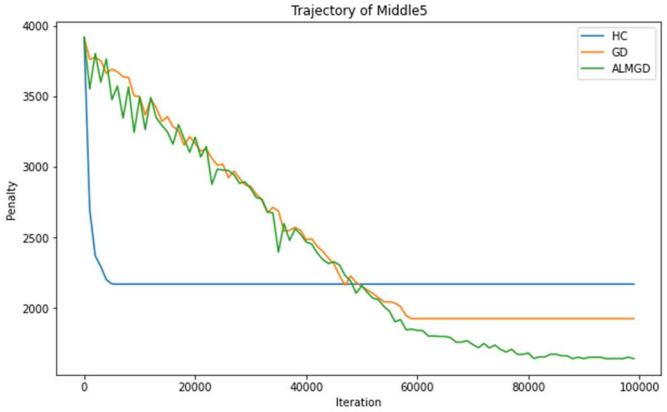

Figure 5. Trajectory Graph for Algorithm Comparison

V. CONCLUSION

This research outlines the development or modification of the great deluge algorithm into ALMGD in the hope of getting a better solution. In addition, this research also proposes a method to produce a feasible solution for the 2021 ITC problem. There are 24 out of 54 total instances or 44.44% of instances that can be found feasible by using hard constraint optimization with tabu late acceptance hill climbing strategy. The best parameter tested is a value of 10 for the length of the fitness array in the late acceptance strategy and the value of the best-known solution for the desired value. The proposed ALMGD algorithm can produce a better solution than the comparison algorithm, namely hill climbing and great deluges. A better solution is generated on all tested feasible instances. The average value of optimization using ALMGD algorithm can reduce the penalty by 36%. Meanwhile, hill climbing is only 29% and using great deluge is 34%.

References

[1] L. Zeng and S. Mizuno, "On the generalized mirrored scheme for double round robin tournaments in sports scheduling," *Asia-Pacific J. Oper. Res.*, vol. 30, no. 3, pp. 1–16, 2013, https://doi.org/10.1142/S0217595913400083.

[2] M. A. Trick, "Sports scheduling," *Springer Optim. Its Appl.*, vol. 45, pp. 489–508, 2011, https://doi.org/10.1007/978-1-4419-1644-0_15.

[3] T. Januario, S. Urrutia, C. C. Ribeiro, and D. De Werra, "Edge coloring: A natural model for sports scheduling," *Eur. J. Oper. Res.*, vol. 254, no. 1, pp. 1–8, 2016, https://doi.org/10.1016/j.ejor.2016.03.038.

[4] A. Daliri, M. Alimoradi, M. Zabihimayvan, and R. Sadeghi, "World Hyper-Heuristic: A novel reinforcement learning approach for dynamic exploration and exploitation," *Expert Syst. Appl.*, vol. 244, p. 122931, 2024, https://doi.org/10.1016/j.eswa.2023.122931.

[5] D. Van Bulck and D. Goossens, "Handling fairness issues in time-relaxed tournaments with availability constraints," *Comput. Oper. Res.*, vol. 115, p. 104856, 2020, https://doi.org/10.1016/j.cor.2019.104856.

[6] X. Yi, D. Goossens, and F. T. Nobibon, "Proactive and reactive strategies for football league timetabling," *Eur. J. Oper. Res.*, vol. 282, no. 2, pp. 772–785, 2020, https://doi.org/10.1016/j.ejor.2019.09.038.

[7] D. Van Bulck and D. Goossens, "A traditional Benders' approach to sports timetabling," *Eur. J. Oper. Res.*, vol. 307, no. 2, pp. 813–826, 2023, https://doi.org/10.1016/j.ejor.2022.10.044.

[8] S. Knust, "Scheduling non-professional table-tennis leagues," *Eur. J. Oper. Res.*, vol. 200, no. 2, pp. 358–367, 2010, https://doi.org/10.1016/j.ejor.2009.01.015.

[9] G. Durán, S. Durán, J. Marenco, F. Mascialino, and P. A. Rey, "Scheduling Argentina's professional basketball leagues: A variation on the Travelling Tournament Problem," *Eur. J. Oper. Res.*, vol. 275, no. 3, pp. 1126–1138, 2019, https://doi.org/10.1016/j.ejor.2018.12.018.

[10] M. B. Wright, "Scheduling fixtures for Basketball New Zealand," *Comput. Oper. Res.*, vol. 33, no. 7, pp. 1875–1893, 2006, https://doi.org/10.1016/j.cor.2004.09.024.

[11] F.-C. Yang, "NBA Sports Game Scheduling Problem and GA-Based Solver," *Proceedings of the 2017 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)*, 2017, pp. 1–5. https://doi.org/10.1109/ICIMSA.2017.7985598.

[12] C. C. Ribeiro and S. Urrutia, "Heuristics for the mirrored traveling tournament problem," *Eur. J. Oper. Res.*, vol. 179, no. 3, pp. 775–787, 2007, https://doi.org/10.1016/j.ejor.2005.03.061.

[13] M. Khelifa, D. Boughaci, and E. Aïmeur, "Evolutionary Harmony Search Algorithm for Sport Scheduling Problem," In: Thanh Nguyen, N., Kowalczyk, R. (eds) Transactions on Computational Collective Intelligence XXX. Lecture Notes in Computer Science, 2018, vol 11120, pp. 93–117. Springer, Cham. https://doi.org/10.1007/978-3-319-99810-7_5.

[14] R. Lewis and J. Thompson, "On the application of graph colouring techniques in round-robin sports scheduling," *Comput. Oper. Res.*, vol. 38, no. 1, pp. 190–204, 2011, https://doi.org/10.1016/j.cor.2010.04.012.

[15] D. Van Bulck and D. Goossens, "The international timetabling competition on sports timetabling (ITC2021)," *Eur. J. Oper. Res.*, vol. 308, no. 3, pp. 1249–1267, 2023, https://doi.org/10.1016/j.ejor.2022.11.046.

[16] D. Van Bulck, D. Goossens, J. Schönberger, and M. Guajardo, "RobinX: A three-field classification and unified data format for round-robin sports timetabling," *Eur. J. Oper. Res.*, vol. 280, no. 2, pp. 568–580, 2020, https://doi.org/10.1016/j.ejor.2019.07.023.

[17] P. Kostuch, "The university course timetabling problem with a three-phase approach," *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling*, 2004, pp. 109–125. https://doi.org/10.1007/11593577_7.

[18] T. Müller, "ITC2007 solver description: A hybrid approach," *Proceedings of the 7th Int. Conf. Pract. Theory Autom. Timetabling, PATAT 2008*, pp. 1–15, 2008. https://doi.org/10.1007/s10479-009-0644-y.

[19] G. H. G. Fonseca, H. G. Santos, T. A. M. Toffolo, S. S. Brito, and M. J. F. Souza, "A SA-ILS approach for the High School Timetabling Problem," *Proceedings of the PATAT 2012 9th Int. Conf. Pract. Theory Autom. Timetabling*, August 2012, pp. 493–496.

[20] G. Dueck, "New optimization heuristics the great deluge algorithm and the record-to-record travel," *J. Comput. Phys.*, vol. 104, pp. 86–92, 1993. https://doi.org/10.1006/jcph.1993.1010.

[21] E. K. Burke and Y. Bykov, "An adaptive flex-deluge approach to university exam timetabling," *INFORMS J. Comput.*, vol. 28, no. 4, pp. 781–794, 2016, https://doi.org/10.1287/ijoc.2015.0680.

[22] V. Ravi, "Optimization of complex system reliability by a modified great deluge algorithm," *Asia-Pacific J. Oper. Res.*, vol. 21, no. 4, pp. 487–497, 2004, https://doi.org/10.1142/S0217595904000357.

[23] E. K. Burke and Y. Bykov, "The late acceptance Hill-Climbing heuristic," *Eur. J. Oper. Res.*, vol. 258, no. 1, pp. 70–78, 2017, https://doi.org/10.1016/j.ejor.2016.07.012.

[24] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Inf. Sci. (Ny).*, vol. 181, no. 20, pp. 4515–4538, 2011, https://doi.org/10.1016/j.ins.2010.07.013.

[25] 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, https://doi.org/10.1016/j.ins.2009.12.025.

[26] A. Muklason, "Automatic Exam Scheduler Solver With Maximal Clique Algorithm and Hyper-heuristics," *Semin. Nas. Technol. Information, Commun. and Ind. 9*, pp. 18–19, 2017. (in Indonesian).

[27] T. Dokeroglu, T. Kucukyilmaz, and E.-G. Talbi, "Hyper-heuristics: A survey and taxonomy," *Comput. Ind. Eng.*, vol. 187, p. 109815, 2024, https://doi.org/10.1016/j.cie.2023.109815.

[28] A. Muklason, "Hyper-heuristics and fairness in examination timetabling

problems," no. December, pp. 1–12, 2017, [Online]. Available: http://eprints.nottingham.ac.uk/42912/.

[29] R. R. Hutama and A. Muklason, "Formation of Initial Solutions for the 2021 International Timetabling Competition," *J. Tech. Inform. and Sys. Inf.*, vol. 8, no. 4, pp. 1939–1944, 2021, https://doi.org/10.35957/jatisi.v8i4.1155. (in Indonesian).

[30] L. Di Gaspero and A. Schaerf, "A composite-neighborhood tabu search approach to the traveling tournament problem," *J. Heuristics*, vol. 13, no. 2, pp. 189–207, 2007, https://doi.org/10.1007/s10732-006-9007-x.

RIZAL RISNANDA HUTAMA *received the S.Kom. and M.Kom. degrees in information systems from Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia, in 2020 and 2022, respectively. He has been a lecturer of information systems department in Institut Teknologi Sepuluh Nopember, since 2022. He is affiliated with the Laboratory of Information System Management. His research interests include combinatorial scheduling and timetabling optimization, heuristics, meta-heuristics, hyper-heuristics algorithm. He can be contacted at email: r.risnanda@its.ac.id.*

AHMAD MUKLASON *Assistant Professor at Data Engineering and Business Intelligence Lab., Department of Information Systems, Faculty of Intelligent Electrical, Electronic Engineering and Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Received his doctoral degree from School of Computer Sciences, The University of Nottingham, UK, in 2017; Master of Science degree of Computer and Information Sciences Department, Universiti Teknologi Petronas, Malaysia, in 2009; and Bachelor of Computer Science from Institut Teknologi Sepuluh Nopember, Surabaya in 2006. He can be contacted at email: mukhlason@is.its.ac.id*