IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 13, No. 2, June 2024, pp. 2155∼2164
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i2.pp2155-2164

The performance analysis of hyper-heuristics algorithms over examination timetabling problems

Ahmad Muklason, Yusnardo Tendio, Helena Angelita Depari, Muhammad Arif Nuriman, I Gusti Agung Premananda
Department of Information Systems, Faculty of Information and Communication Technology, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia

Article Info

Article history:

Received Mar 30, 2023
Revised Oct 8, 2023
Accepted Oct 22, 2023

Keywords:

Exam timetabling problem
Great deluge
Hyper-heuristics
Simulated annealing
Tabu search

ABSTRACT

In general, uncapacitated exam timetabling is conducted manually, which can be time-consuming. Many studies aim to automate and optimize uncapacitated exam timetabling. However, pinpointing the most efficient algorithm is challenging since most studies assert that their algorithms surpass previous ones. To identify the optimal algorithm, this research evaluates the performance of four algorithms: Hill climbing (HC), simulated annealing (SA), great deluge (GD), and tabu search (TS) in addressing the exam timetabling problem. The Kempe chain operator's influence on optimization solutions is also examined. A simple random method is employed to select the low-level heuristic (LLH). The Carter (Toronto) dataset served as the test material, with each algorithm undergoing 200,000 iterations for comparison. The results indicate that the TS algorithm is superior, providing the best solution in 13 instances. The use of a tabu list enhanced the search process's efficiency by preventing redundant modifications. The Kempe chain LLH exhibited a tendency towards achieving better solutions.

Corresponding Author:

Ahmad Muklason
Department of Information Systems, Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
St. Raya ITS, Kampus ITS Sukolilo Surabaya, 60111, East Java, Indonesia
Email: mukhlason@is.its.ac.id

1. INTRODUCTION

The uncapacitated exam timetabling is a prevalent issue encountered by educational institutions and private or state companies. This problem falls within the realm of optimization, which aims to find solutions that take into account limitations or constraints. Specifically, optimization seeks to identify the optimal solution by minimizing an objective function [1], [2]. In previous studies [3]-[7], a variety of algorithms, especially single-based metaheuristic algorithms, have been used to address uncapacitated exam timetabling problems. Many of these studies claim that the developed algorithms outperform others. However, the challenge lies in the fact that each study utilizes different comparative algorithms, making it challenging to determine which algorithm performs the best. Moreover, there are several factors that can impact the results of these studies, including the utilization and selection methods of low-level heuristics (LLH) or neighborhood operators.

To address this issue, this study employs a variety of algorithms and compares their performance in order to determine the most effective solution of metaheuristic single-based solution in hyper heuristic method to solve exam timetabling. The algorithms that will be compared include hill climbing (HC), simulated annealing (SA), great deluge (GD), and tabu search (TS). The HC algorithm is inspired by the strategies used by mountain climbers in finding the most efficient path to their destination [8]. The SA algorithm is a metaheuristic technique inspired by the annealing process, and modified to find global optimum solutions [9], [10]. The GD algorithm is a local search procedure that has been widely used in research to solve exam timetabling problems at universities [11], [12]. The TS algorithm is another local search metaheuristic that is commonly used to solve combinatorial optimization problems [13], [14]. These algorithms will be evaluated in terms of their cost penalty values, in order to identify the algorithm that yields the most optimal solution for the exam timetabling problem. Additionally, this research also investigates the Kempe chain LLH, which has produced great results in previous studies [15].

The research will be presented in several sections. Section 2 will include a literature review, section 3 will outline the methodology, and section 4 will present the results and discussion. Finally, section 5 will provide a conclusion, allowing readers to gain a comprehensive understanding of the research process and findings.

2. DATASET CARTER

This research utilizes the Carter dataset as the primary source of data. The Carter dataset is well-known for its stability and its variables that are representative of real-world scenarios [16]. The characteristics of the Carter dataset can be seen in Table 1.

Table 1. Dataset Carter

Dataset	Timeslots	Exams	Students	Conflict Density
Car91	35	682	16,925	0.13
Car92	32	543	18,419	0.14
Ear83	24	190	1,125	0.27
Hec92	18	81	2,823	0.42
Kfu93	20	461	5,349	0.06
Pur93	42	2,419	30,029	0.03
Rye92	23	486	11,483	0.07
Sta83	13	139	611	0.47
Tre92	23	261	4,360	0.18
Uta92	35	622	21,266	0.13
Ute92	10	184	2,749	0.08
Yor83	21	181	941	0.29

The examination timetabling problem in the Carter dataset involves allocating a set of exams to specific time slots while adhering to the constraints that are present. These constraints can be divided into two categories: soft constraints and hard constraints. Soft constraints refer to requirements such as ensuring that the allocated exam time is sufficient for students to have adequate time for studying and making improvements. Hard constraints, on the other hand, refer to restrictions that must be strictly followed, such as ensuring that no student is scheduled to take more than one exam during the same time period [17].

Hard constraints are constraints that must be met, while soft constraints are met as much as possible. When the hard constraint has been met, then the objective function is used to reduce the penalty value. The penalty value is obtained from how much the soft constraint is violated. The objective function formula to calculate the penalty is (1):

$$fitness = \frac{(\sum_{s=0}^{4} \omega_s \times N_s)}{S} \tag{1}$$

The distance between the two timeslots is calculated with the weight $\omega_s = 2^s$. While N_s is a representation of the number of students who violate the soft constraint. And S is the total number of students in the schedule. For example, a student has 2 exams a and b, where the 2 exams are separated by 3 timeslots so that the weight of the penalty is 2^1, while N_s represents the number of students who violated the soft constraints on the two exams and S is the total of all students.

3. METHOD

This section outlines the procedures implemented to generate and improve solutions through the use of HC, SA, GD, and TS algorithms. It discusses the methods for forming the initial solution, developing the local landscape heuristic, and the algorithms utilized in the study. Each algorithm employed in this study contributes to the optimization process by providing distinct strategies for improving solutions.

3.1. Formation of initial solution

This study used the saturation degree with backtracking algorithm to produce a feasible initial solution. The saturation degree determines the timeslot in the exam by prioritizing the exam that has the lowest probability of entering the timeslot. If there is no proper timeslot for an exam, a backtracking algorithm will be performed.

3.2. Hyper-heuristic

The hyper-heuristic method is composed of two main components: high-level heuristic (HLH) and LLH [18], [19]. The HLH component comprises two subcomponents: LLH selection and move acceptance. The algorithms compared in this study focus on the move acceptance component. For LLH selection, a random method is employed to select LLH randomly in each iteration. In the LLH component, six distinct LLHs are utilized, which will be discussed in detail in section 3.2.

3.3. Low level heuristic

In this study, a variety of LLH operators are utilized as components that undergo modification in each iteration. These operators are crucial in the optimization process and contribute significantly to the overall performance of the algorithm. The operators employed include:

- Random move: select an exam at random, then move the exam randomly from the original timeslot to another timeslot.
- Random swap two: choose two exams at random, then swap the timeslots of the two exams.
- Random move two: select two exams at random, then move the two exams randomly from the original timeslot to another timeslot.
- Random swap three: randomly pick three exams, and then exchange the timeslots assigned to those exams. For example, exams a, b, c are selected randomly, then exam b timeslot is changed to exam a timeslot, exam c timeslot is changed to exam b timeslot, and exam a timeslot is changed to exam c timeslot.
- Random move three: the same as random move two, but using three random exams.
- Kempe chain: two sets of exams that do not conflict with each other, the timeslots are swapped. In a study conducted by [20], concluded that Kempe Chain is the right choice to optimize the solution of the examination timetabling problem.

3.4. Implementation of hill climbing algorithm

The HC algorithm, a local search technique, follows an iterative approach. It starts by selecting an initial population from the problem's solution space and then iteratively modifies one element at a time to create new solutions. If a new solution is superior to the current one, it is accepted, and this iterative process continues until further improvements are not possible [8]. In this study, the general HC algorithm is employed, commencing with a randomly generated move and utilizing local search heuristics (LSHs) to discover new solutions. The HC algorithm maintains the best solution found throughout its execution, which is set to a total of 200,000 iterations while utilizing the random move operator. The pseudo-code for this study's implementation can be found in Algorithm 1.

3.5. Implementation of the simulated annealing algorithm

SA is a meta-heuristic search technique inspired by the process of heating a solid to its melting point and then cooling it [21]. The purpose of this method is to efficiently find the optimum solution without spending excessive time, and its ability to avoid local optimality is considered an advantage [20], [22]. In this study, SA is compared with different combinations of LSHs. The first combination, referred to as SA1, utilizes all LLHs except for the Kempe Chain operator. In contrast, the second combination, referred to as SA2, uses only three LLHs: random move, random swap two, and Kempe Chain operators. This variation is implemented to investigate the impact of the Kempe Chain operator, which is known to be time-consuming. Consequently, the second combination relies on just two simpler operators. The initial temperature is set to 10, and the cooling rate is set to 0.0000115. The iteration continues until the temperature drops below 1. These parameter values have been determined as optimal to ensure approximately 200,000 iterations, facilitating a fair comparison against other algorithms. The pseudo-code for implementing this algorithm can be found in Algorithm 2.

Algorithm 1: Hill Climbing

```
bestScoreHc=intialScore;
hcSolution=intialSolution;
counter=0;
maxCounter=200000;
while $counter < maxcounter$ do
    generate new solution with LLH Random Move;
    newScore=evaluate(newSolution);
    if $newScore < bestScore$ then
        hcSolution=newSolution;
        bestScoreHc=newScore;
    end
    counter++;
end
```

Algorithm 2: Simulated Annealing

```
currentSolution=initialSolution;
bestSolution=intialSolution;
while $temp > 1$ do
    newSolution=generateNewSolution(LLH);
    d=evaluate(newSolution) - evaluate(currentSolution);
    if $d > 0$ then
        currentSolution=newSolution;
    else if $random[0, 1 < exp(-d/T_0)$ then
        currentSolution=newSolution;
    end
    if $evaluate(currentSolution) < evaluate(bestSolution)$ then
        bestSolution=currentSolution;
    end
    updateTemp();
end
```

3.6. Great deluge algorithm implementation

The GD algorithm, introduced by Dueck in 1993, is a real-world applicable local search technique that allows slight degradation of solution quality to reach new maxima. It leverages the concept of descending a hill temporarily to avoid rain and thereby reach greater heights efficiently [23]. In this study, the GD algorithm from [24] is modified to include the Kempe Chain operator in every iteration. Additionally, a count variable triggers a reheating procedure if there's no improvement in the solution for 1,000 iterations, incrementing until reaching 50 iterations, at which point the reheating procedure is activated, involving the random move and random swap operators applied to the current best solution, as detailed in Algorithm 3.

3.7. Implementation of the tabu search algorithm

The TS algorithm, a heuristic method, employs short-term memory to prevent getting trapped in local optima. It utilizes a tabu list to store recently evaluated solutions, and during each optimization iteration, the solution is compared to the tabu list. If it's already in the list, it's not reevaluated in the next iteration. When there are no new solutions outside the tabu list, the best obtained value becomes the actual solution, enabling the algorithm to escape local optima and enhance the chances of finding the global optimum [25]. In this study, we use the TS algorithm, which is based on the method described in [24]. The algorithm employs the use of a tabu list to store recently evaluated solutions. At each iteration, the solution to be evaluated is first matched with the contents of the tabu list. If the solution is already on the tabu list, it is not considered in the next iteration. If all remaining solutions are members of the taboo list, the best solution found so far is considered as the actual solution. The pseudocode for this algorithm can be found in Algorithm 4. In this study, the maximum taboo list size is set to 10 solutions and all LLH operators are used, with a total of 200,000 iterations.

Algorithm 3: Great Deluge Pseudo-code

```
bestSolution = initialSolution;
level = initialScore;
counter = 0;
maxCounter = 200000;
notImprovingCounter = 0;
rehatingCounter = 0;
while $counter < maxCounter$ do
    newSolution = KempeChain(generateNewSolution(Random(LLH1, LLH2)));
    if $evaluate(newSolution) < evaluate(bestSolution)$ then
        bestSolution = currentSolution = newSolution;
        notImprovingCounter = 0;
    end
    else
        notImprovingCounter++;
    end
    if $notImprovingCounter \geq 1000$ then
        notImprovingCounter = 0;
        rehatingCounter++;
    end
    if $rehatingCounter \geq 50$ then
        bestSolution = RandomSwapTwo(RandomMove(bestSolution));
        currentSolution = bestSolution;
    end
    updateLevel();
    counter++;
end
```

Algorithm 4: Tabu Search Pseudo-code

```
sBest=initialSolution;
bestCandidate=intialSolution;
tabuList={};
counter=0;
maxCounter=200000;
while $counter < maxcounter$ do
    sNeighbhor=getNeighborByRandom(bestCandidate);
    for $sCandidate\ in\ sNeighbor$ do
        if $not\ tabuList.contains(sCandidate)\ andfitness(sCandidate)fitness > (bestCandidate$ then
            bestCandidate=sCandidate;
        end
    end
    if $fitness(bestCandidate > fitness(sBest$ then
        bestCandidate=sCandidate;
    end
    tabuList.append(bestCandidate);
    if $tabuList.size > maxTabuSize$ then
        tabuList.pop[0];
    end
    counter++;
end
```

4. EXPERIMENT RESULT

The testing of these algorithms was conducted using the Python programming language on a Jupyter Notebook environment. The experiments were performed on a Windows 10 Pro operating system, utilizing specific hardware specifications for the testing setup. The hardware configuration employed during the experiments consisted of an Intel (R) Core(TM) i7-4510U CPU running at a base frequency of 2.00 GHz, which can boost up to 2.60 GHz, and the system was equipped with 8 GB of RAM.

4.1. Test results

The experimental results from the steps that have been carried out can be seen in Table 2. The TS algorithm demonstrated superior performance over other algorithms by producing the best average and optimal solutions for all 13 instances. Both the GD and SA2 algorithms yielded relatively comparable outcomes.

However, for some datasets, SA2 provided superior results, while in others, GD emerged as the better option. SA1 underperformed compared to SA2, an outcome attributable to the effects of the employed Kempe chain, which will be elaborated upon in the subsequent subsection. Of the four algorithms assessed, the HC algorithm consistently recorded the least favorable outcomes across all datasets.

In this study, the HC algorithm was designed as a direct approach that only acknowledges improved solutions. This method, while straightforward, bears the inherent risk of becoming ensnared in local optima, thus hampering its potential to identify the global optimum. Additionally, the exclusive reliance on a singular heuristic, specifically the random move within the LLH implementation, limits the exploration of alternative solutions, often leading to less-than-optimal results. These observations underscore the importance of integrating advanced algorithms and a diverse heuristic portfolio to mitigate these limitations and enhance the optimization process's search efficacy.

Table 2. Optimization result

Dataset	HC		SA1		SA2		GD		TS	
	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg
Car91	9.03	10.3	8.6	8.7	6.7	6.7	6.4	6.5	6.0	6.1
Car92	8.0	9.5	7.3	7.4	5.3	5.4	5.1	5.2	4.8	4.9
Ear83	70.3	79.8	52.0	53.4	40.5	41.9	40.7	41.6	39.1	40.5
Hec92	16.1	18	15.9	16.5	11.2	11.5	11.3	11.5	11.1	11.5
Kfu93	22.2	23.9	21.6	22.3	16.9	17.3	16.5	16.8	15.1	15.5
Lse92	18.5	19.2	17.7	18.4	13.0	13.5	12.7	13.4	12.1	12.3
Pur93	9.5	9.7	9.3	9.3	8.1	8.2	7.7	7.8	7.1	7.2
Rye92	15.6	17.1	14.6	15.2	11.2	11.6	10.9	11.0	9.8	10.1
Sta8	172.5	174.1	172.4	173.4	152.2	154.2	157.3	157.4	157.1	157.4
Tre92	12.9	13.1	12.7	12.8	9.6	9.7	9.4	9.7	9.5	9.4
Uta92	6.3	6.5	6.1	6.2	4.3	4.4	4.2	4.2	4.0	4.0
Ute92	37.1	38.3	36.9	38.0	26.7	27.4	26.1	27.1	25.6	26.6
Yor83	54.1	55.2	53.7	54.0	40.3	40.9	40.5	41.3	39.6	40.6

4.2. Kempe chain effect

From these results, it can be seen that the Kempe Chain LLH plays an important role in producing good optimization. The HC and SA1 algorithms, which do not use the Kempe Chain, produced worse results in all datasets compared to the other three algorithms that use the Kempe Chain. Figure 1 illustrates the difference in cost reduction between the SA1 and SA2 algorithms on the Ear83 dataset. In SA1, which does not use the Kempe Chain, the cost reduction occurs gradually. Meanwhile, in SA2, from the beginning of the iteration, the cost reduction occurs drastically. At iteration 100,000, a drastic increase in cost is seen, followed by a drastic decrease in cost. This is caused by the Kempe Chain, which can change many exams directly without damaging the feasibility of a solution. The application of the Kempe Chain and the SA algorithm that allows accepting worse solutions leads to drastic cost jumps like in Figure 1 . However, this is a beneficial thing, as it shows massive solution exploration, and in the end, the results prove that the produced solution is better.

In addition, the comparison of feasibility between Kempe Chain LLH and other LLH shows a significant difference. Table 3 shows the level of feasibility produced in its application using the SA algorithm. LLHs other than Kempe Chain show a very high probability of producing new solutions that are not feasible. The random move LLH recorded a level of producing feasible solutions at 15.8%. While other LLHs produced percentages below 6%. Despite in the random swap three and random move three LLHs, the percentage of producing feasible solutions was below 1%. This shows that the use of LLHs other than Kempe Chain is dominated by producing infeasible solutions. So in the iteration that produces infeasible solutions, this iteration becomes wasted. This is what causes the significant difference between algorithms that use Kempe Chain and those that do not.

4.3. Algorithm ranking

To figure out the best algorithm, we ranked them based on test scores. For each dataset, we gave each algorithm a rank, then averaged those ranks for all datasets. Table 4 gives the average ranks for the five algorithms. From this, the TS algorithm did the best on the carter dataset with an average rank of 1. SA2 came in second, with GD in third.

Figure 2 shows how the five algorithms differ when working on the Ear83 dataset. The TS algorithm we used is a lot like the HC algorithm, but with a twist. It only takes on better solutions, and it uses a 'tabu list' to avoid doing the same work over and over. This makes it get to better solutions quickly without getting stuck.

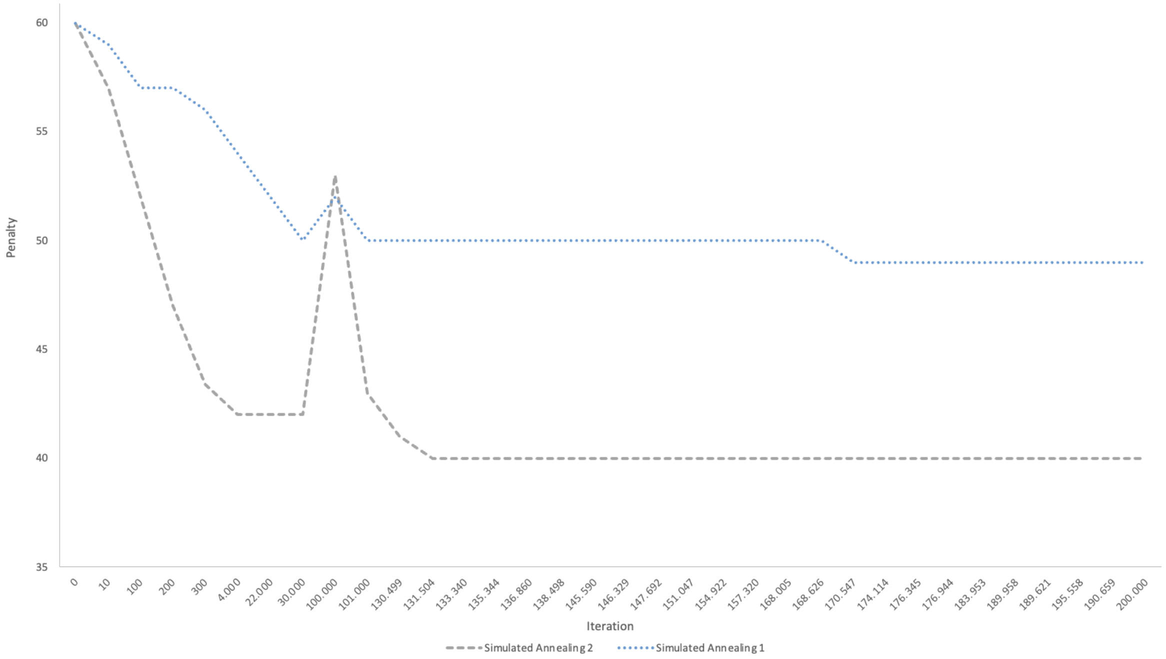

Figure 1. Comparison of SA 1 with SA 2 on Ear83 Dataset

Table 3. LLH comparison

Neighbor Search	Feasible Amount	Number of Solutions Not Feasible
Random Move	372 (15,8 %)	1971 (84,2 %)
Random Swap Two	143 (5,7 %)	2345 (94,3 %)
Random Move Two	60 (2,5 %)	2340 (97,5 %)
Random Swap Three	13 (0,5 %)	2421 (99,5 %)
Random Move Three	4 (0,2 %)	2331 (99,8 %)
Kempe Chain	4824 (100 %)	0

Table 4. Algorithm rangking

Algorithm	Rangking
TS	1
SA2	3
GD	3.7
SA1	4
HC	5

The SA and GD algorithms are different from TS. They sometimes take worse solutions to explore more options, which helps them avoid getting stuck. The GD algorithm has a special 'reheating' step. This helps it do really well on some datasets but not as well on others. You can see this reheating in Figure 2 where costs sometimes go up, especially at the end. The SA algorithm is less likely to accept worse solutions as it nears the end of its work. But in this case, that didn't always lead to the best results.

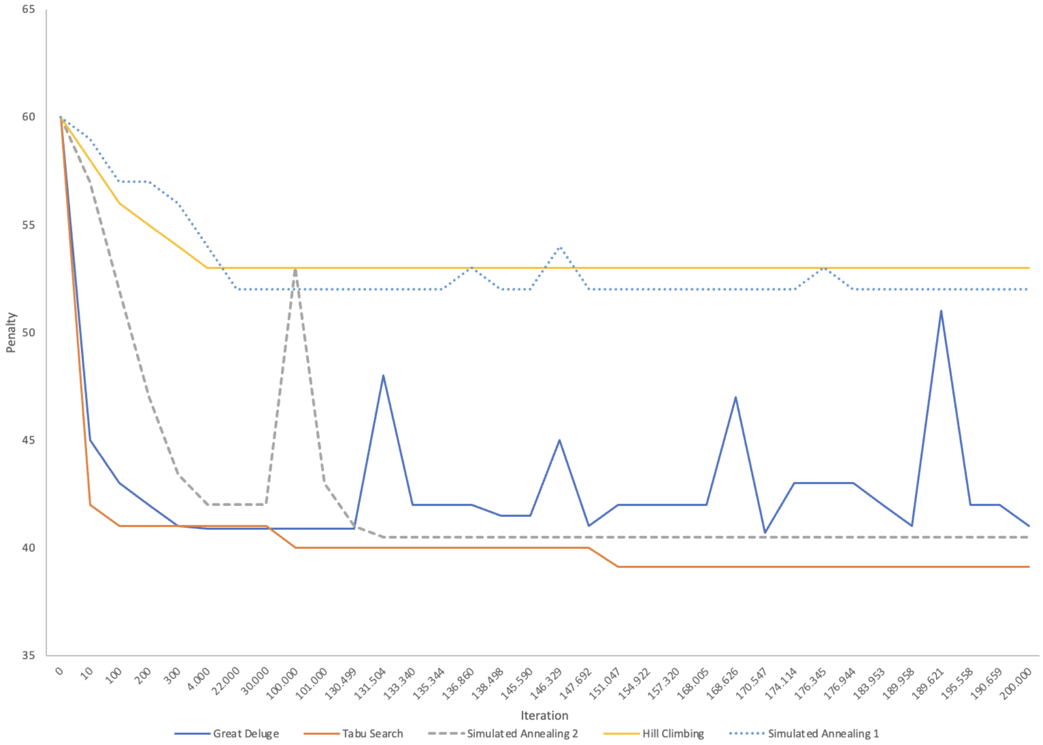

Figure 2. Illustration of the solution search process from the five algorithms on Ear83

5. CONCLUSION

This research aimed to evaluate and compare the performance of four optimization algorithms: SA, GD, TS, and HC, in addressing the exam timetabling problem. The study also investigated the influence of the Kempe Chain operator on the derived solutions. The algorithms were tested using the Carter dataset, and their outcomes were systematically analyzed. Results indicated that the TS algorithm outperformed its counterparts, securing an average rank of 1. SA2 followed in second place, with GD ranking third. This performance disparity can be attributed to the intrinsic characteristics of each algorithm. While the TS algorithm, mirroring HC, predominantly accepted superior solutions, its incorporation of a tabu list augmented search efficiency. In contrast, both SA and GD were more exploratory, occasionally accepting suboptimal solutions, facilitating easier evasion of local optima. The Kempe Chain operator markedly enhanced solution quality. Algorithms utilizing this operator demonstrated superior performance; a comparison between Kempe Chain and other neighborhood operators revealed a stark difference. Specifically, Kempe Chain yielded over 94% feasible solutions, whereas other operators hovered below 6%. Future studies might consider hybridizing optimization algorithms, particularly integrating TS with either SA or GD, given their notable efficacy demonstrated herein. Furthermore, delving into alternative neighborhood operators, especially those evolving from the LLH Kempe Chain, might provide insights tailored to the unique challenges of exam timetabling, potentially leading to even higher solution quality.

REFERENCES

[1] B. A. Aldeeb *et al.*, “A comprehensive review of uncapacitated university examination timetabling problem,” *International Journal of Applied Engineering Research*, vol. 14, no. 24, pp. 4524-4547, 2019.

[2] F. Peres and M. Castelli, “Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development,” *Applied Sciences*, vol. 11, no. 14, pp. 1-39, 2021, doi: 10.3390/app11146449.

[3] N. Leite, C. M. Fernandes, F. Melício, and A. C. Rosa, “A cellular memetic algorithm for the examination timetabling problem,” *Computers & Operations Research*, vol. 94, pp. 118–138, 2018, doi: 10.1016/j.cor.2018.02.009.

[4] N. Pillay and E. Özcan, “Automated generation of constructive ordering heuristics for educational timetabling,” *Annals of Operations Research*, vol. 275, no. 1, pp. 181–208, 2019, doi: 10.1007/s10479-017-2625-x.

[5] B. A. Aldeeb *et al.*, “Hybrid intelligent water Drops algorithm for examination timetabling problem,” *Journal of King Saud Univer-*

sity - Computer and Information Sciences, vol. 34, no. 8, pp. 4847-4859, 2021, doi: 10.1016/j.jksuci.2021.06.016.

[6] M. A. Al-Betar, “A -hill climbing optimizer for examination timetabling problem,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 1, pp. 653–666, 2021, doi: 10.1007/s12652-020-02047-2.

[7] R. Bellio, S. Ceschia, L. Di Gaspero, and A. Schaerf, “Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling,” *Computers & Operations Research*, vol. 132, 2021, doi: 10.1016/j.cor.2021.105300.

[8] A. Rjoub, “Courses timetabling based on hill climbing algorithm,” *International Journal of Electrical and Computer Engineering*, vol. 10, no. 6, pp. 6558–6573, 2020, doi: 10.11591/IJECE.V10I6.PP6558-6573.

[9] S. L. Goh, G. Kendall, and N. R. Sabar, “Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem,” *Journal of the Operational Research Society*, vol. 70, no. 6, pp. 873–888, 2019, doi: 10.1080/01605682.2018.1468862.

[10] N. Leite, F. Melício, and A. C. Rosa, “A fast simulated annealing algorithm for the examination timetabling problem,” *Expert Systems with Applications*, vol. 122, pp. 137–151, 2019, doi: 10.1016/j.eswa.2018.12.048.

[11] I. G. A. Premananda and A. Muklason, “Complex University Timetabling Using Iterative Forward Search Algorithm and Great Deluge Algorithm,” *Khazanah Informatika: Jurnal Ilmu Komputer dan Informatika*, vol. 7, no. 2, pp. 39–46, 2021.

[12] M. N. M. Kahar and G. Kendall, “A great deluge algorithm for a real-world examination timetabling problem,” *Journal of the Operational Research Society*, vol. 66, no. 1, pp. 116–133, 2015, doi: 10.1057/jors.2012.169.

[13] M. Chen, X. Tang, T. Song, C. Wu, S. Liu, and X. Peng, “A Tabu search algorithm with controlled randomization for constructing feasible university course timetables,” *Computers & Operations Research*, vol. 123, 2020, doi: 10.1016/j.cor.2020.105007.

[14] F. H. Awad, A. Al-Kubaisi, and M. Mahmood, “Large-scale timetabling problems with adaptive tabu search,” *Journal of Intelligent Systems*, vol. 31, no. 1, pp. 168–176, 2022, doi: 10.1515/jisys-2022-0003.

[15] S. L. Goh, G. Kendall, N. R. Sabar, and S. Abdullah, “An effective hybrid local search approach for the post enrolment course timetabling problem,” *OPSEARCH*, vol. 57, no. 4, pp. 1131–1163, 2020, doi: 10.1007/s12597-020-00444-x.

[16] A. K. Mandal, M. N. M. Kahar, and G. Kendall, “Addressing Examination Timetabling Problem Using a Partial Exams Approach in Constructive and Improvement,” *Computation*, vol. 8, no. 2, pp. 1-28, 2020, doi: 10.3390/computation8020046.

[17] X. Hao, R. Qu, and J. Liu, “A Unified Framework of Graph-Based Evolutionary Multitasking Hyper-Heuristic,” *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 35–47, 2021, doi: 10.1109/TEVC.2020.2991717.

[18] E. K. Burke et al., “Hyper-heuristics: A survey of the state of the art,” *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013, doi: 10.1057/jors.2013.71.

[19] S. S. Choong, L. P. Wong, and C. P. Lim, “Automatic design of hyper-heuristic based on reinforcement learning,” *Information Sciences*, vol. 436–437, pp. 89-107, 2018, doi: 10.1016/j.ins.2018.01.005.

[20] M. Cheraitia and S. Haddadi, “Simulated annealing for the uncapacitated exam scheduling problem,” *International Journal of Metaheuristics*, vol. 5, no. 2, pp. 156-170, 2016, doi: 10.1504/ijmheur.2016.10001113.

[21] K. Amine, “Multiobjective Simulated Annealing: Principles and Algorithm Variants,” *Advances in Operations Research*, vol. 2019, pp. 1-14, 2019, doi: 10.1155/2019/8134674.

[22] X. Pan, L. Xue, Y. Lu, and N. Sun, “Hybrid particle swarm optimization with simulated annealing,” *Multimedia Tools and Applications*, vol. 78, no. 21, pp. 29921–29936, 2019, doi: 10.1007/s11042-018-6602-4.

[23] R. Guha *et al.*, “Deluge based genetic algorithm for feature selection,” *Evolutionary Intelligence*, vol. 14, no. 2, pp. 357–367, 2021, doi: 10.1007/s12065-019-00218-5.

[24] H. Turabieh and S. Abdullah, “An integrated hybrid approach to the examination timetabling problem,” *Omega*, vol. 39, no. 6, pp. 598-607, 2011, doi: 10.1016/j.omega.2010.12.005.

[25] P. McMullan, “An extended implementation of the great deluge algorithm for course timetabling,” in *Computational Science – ICCS 2007*, Berlin, Heidelberg: Springer, 2007, doi: 10.1007/978-3-540-72584-8_71.

BIOGRAPHIES OF AUTHORS

Ahmad Muklason is assistant professor at Data Engineering and Business Intelligence Laboratory, 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 at Department of Computer and Information Sciences, 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.

Yusnardo Tendio is big data consultant in PT. Metrodata Electronics Tbk. He received Bachelor of Computer Sciences (B.Comp.Sc.) degree holder from the Department of Information Systems, Faculty of Intelligent Electrical, Electronics Engineering and Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2020. He can be contacted at email: ytendio@gmail.com.

Helena Angelita Depari received Bachelor of Computer Sciences (B.Comp.Sc.) degree holder from the Department of Information Systems, Faculty of Intelligent Electrical, Electronics Engineering and Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2020. She can be contacted at email: helenaangelitadepari@gmail.com.

Muhammad Arif Nuriman is undergraduate student at the Department of Information Systems, Faculty of Intelligent Electrical, Electronics Engineering and Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2020. He can be contacted at email: nurimanmuhammadarif@gmail.com.

I Gusti Agung Premananda received his master degree from Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2021 in information systems; and Bachelor of Computer Science from Institut Teknologi Sepuluh Nopember, Surabaya in 2019. Presently he is taking the doctoral program in Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia in 2021 in information systems. He can be contacted at email: igustiagungpremananda@gmail.com.