A Great Deluge Algorithm for University Examination Timetabling Problem

Lnasya Syafitrie, Komarudin
Department of Industrial Engineering, Faculty of Engineering
Universitas Indonesia
Depok, Indonesia
lnasya.syafitrie01@ui,ac,id, komarudin74@ui.ac.id

Abstract

Scheduling involves determining the start and the end time of resource assignments, events, or tasks to achieve a particular goal. A good schedule can increase the efficiency and the utilization of resources. The University Examination Timetabling Problem (UETP) is an NP-hard problem that determines the schedule of exams given students' exam lists and limited period and room. This study proposes using a Great Deluge Algorithm (GD) with local search operators to solve the UETP. The method is used to generate solutions for the uncapacitated Toronto benchmark datasets. The proposed method shows promising results compared to the original Great Deluge Algorithm and the Modified Great Deluge Algorithm available in previous literature.

Keywords
great deluge algorithm, uncapacitated, examination timetabling problem, local search, scheduling

1. Introduction

Scheduling is a decision-making process used regularly in various manufacturing and service industries. Scheduling involves allocating resources to activities in a specific timeframe to optimize objectives (Pinedo 2008). The scheduling process involves setting limited resources to meet a particular goal while satisfying constraints (Aldeeb et al. 2019). Schmidt and Stroehlein (1980) define scheduling as the control of participants, meetings, and periods so that all meetings can be done in available periods for all participants. There are many different scheduling applications, such as nurse scheduling, transportation scheduling, disaster management scheduling, production scheduling, employee scheduling, educational timetabling, etc. Scheduling provides benefits such as increasing time efficiency and resource utilization.

Highly complex scheduling problems have a non-deterministic polynomial solving time, better known as *NP-hard* (Abayomi-Alli et al. 2020). The high complexity of scheduling problems makes it a compelling topic for research. Educational timetabling is among the most frequently researched and developed from various scheduling problems. One of the reasons is that educational timetabling is needed each semester and each year in all academic institutions (Qu et al. 2009).

Burke et al. (2007) state that educational timetabling generally deals with a set of activities (classes, exams, etc.) that needs to be allocated in a specific period. These allocations need to be organized while satisfying two types of constraints: hard and soft constraints. Hard constraints cannot be violated, usually because of physical constraints such as classes taken by the same group of students. On the other hand, soft constraints need to be satisfied as much as possible.

Educational timetabling problems are divided into several sub-problems, with the most common being the Course Timetabling Problem (CTP) and the Examination Timetabling Problem (ETP). ETP is also further divided into other sub-problems: the University Examination Timetabling Problem (UETP), where the subject matters are specific to universities. Scheduling in universities is naturally a complex problem, with variables such as subjects, lecturers, students, rooms, time, and preference. With universities still scheduling manually, aligning all those variables generally takes a long time.

A survey conducted by Khair et al. (2018a) shows that many different methods and approaches have been used in the UETP. Heuristics and the Genetic Algorithm (GA) is the most common method used to solve the UETP. Based on a survey conducted by Aldeeb et al. (2019), the best solutions use a hybrid metaheuristic version that combines local-search and population-based methods. Great Deluge (GD) is a method that has been used to try and solve the UETP and has gained traction in the scheduling literature (Mandal et al., 2020). GD has its origins in solving the Travelling Salesman Problem (TSP) and, in some cases, outperforms other algorithms like Simulated Annealing (SA) and the Hill Climbing Algorithm (HC) (Dueck, 1993). In ETP, GD is a candidate that can generate good quality solutions (Mandal and Kahar, 2020).

1.1 Objectives

UETP is still a highly discussed subject, with methods still being developed. One of these methods, namely the Great Deluge Algorithm, has become the motivation for this research. This research proposes a new modified Great Deluge algorithm with local search operators to solve UETP with a benchmark dataset. This research also compares results between other GD methods and the best-known solution for UETP.

2. Literature Review

Dueck first introduced GD as an algorithm similar to the Threshold Accepting Hill Algorithm (TA) and SA (Dueck, 1993). In the study, the algorithms were compared in solving the TSP. Like SA, GD allows worse configurations to escape the local optimum. The difference is rather than using probabilities to accept/reject solutions based on the temperature like in SA, GD uses an upper bound that decreases over iterations.

The usage of GD in UETP was first proposed by E. Burke et al. (2004), which suggests that GD is an effective algorithm for solving the UETP. This usage is developed further to solve the problem as a flex-deluge algorithm. The algorithm proved to successfully get some of the best solutions at that time (E. Burke and Bykov 2006). Kahar and Kendall (2015) modified the Dueck GD algorithm to receive better solutions in real-world cases at University Malaysia Pahang.

Currently, the best-known solutions of most of the UETP are held by using hybrid metaheuristics (Aldeeb et al. 2019). The most used method for hybridization is Graph Heuristics, which has been combined with GD (Mandal and Kahar, 2020; Muklason et al. 2019), Tabu Search (TS) and HC (Dewi et al. 2021), Ant Colony Optimization (ACO) (Khair et al. 2018b), Fast-SA (Leite et al. 2019), etc. In (Bellio et al. 2021), a novel combination of Local Search Operators is used along with SA with a principled tuning procedure, which improves upon previous literature works.

Mandal and Kahar (2020) proposed a hybridization of Graph Heuristics with other metaheuristics such as GD, TS, SA, Late Acceptance Hill-Climbing (LAHC), and Variable Neighborhood Search (VNS) to solve UETP problems. In the study, the performance of GD is shown to be better than other methods, which is caused by the ability of GD to escape from the local optimum.

2.1 Uncapacitated Examination Timetabling Problem

UETP is divided into two sub-problems based on the hard constraint of class capacity, which are Capacitated UETP and Uncapacitated UETP (UUETP) (Leite et al. 2014). A benchmark dataset that is commonly used for UUETP is the Toronto benchmark dataset (Carter et al., 1996). The formulations proposed by Carter et al. (1996) did not consider the room capacity. Therefore, it is classified as a UUETP. In those formulations, the considered parameters and variables are the time between exams, time periods, and exams with the same students (Bellio et al. 2021).

The UUETP mathematical model based on Carter et al. (1996) has the parameters exam as $E = \{1, \dots, E\}$, time periods as $P = \{1, \dots, P\}$, and a total number of students as M. A conflict matrix of $(c_{ij})_{E\times E}$, represents the number of students two exams have in common, where i and j are the exam pair $(i, j \in \{1, \dots, E\})$ and c_{ij} is equal to the number of common students in i and j. The decision variable for the UUETP is p_k, which is the period assigned to exam k, $(k \in \{1, \dots, P\})$.

$$\text{minimize } \frac{\sum_{i-1}^{E-1} \sum_{j=i+1}^{E} \{c_{ij} \times proximity(p_i,\} p_j)}{M} \qquad (1)$$

$$proximity(p_i, p_j) = \begin{cases} \frac{2^6}{2^{|p_i - p_j|}}, if 1 \le |p_i - p_j| \le 5 \\ 0, \qquad\qquad otherwise \end{cases} \tag{2}$$

$$\sum_{i=1}^{E-1} \sum_{j=i+1}^{E} c_{ij} \times \lambda\,(p_i - p_j) = 0 \tag{3}$$

$$\lambda(p_i - p_j) = \begin{cases} 1, & if\ p_i = p_j \\ 0, & otherwise \end{cases} \tag{4}$$

The objective function in equation (1) is to minimize the proximity cost, which is a soft constraint of the time students have between exams. Equation (2) shows the proximity penalty between two exams. If two conflicted exams are allocated within 2-5 periods, the proximity penalty can have values of 16, 8, 4, 2, and 1 for period distances of 2, 3, 4, and 5, respectively. If conflicted exams are not allocated within these 2-5 periods, they are given a penalty value of 0. Constraints (3) and (4) show the hard constraints where exams with conflicting students cannot be allocated to the same period.

3. Methods

Our solution method works in two stages, that are based on Bellio et al. (2021). The difference with our study lies in the first stage and the algorithm used. In the first stage, a feasible solution that satisfies the constraint of no conflicted students is generated. The second stage is the improvement stage, where the objective is to minimize the penalty cost while still satisfying the constraints.

The beginning of the first stage sorts the exams based on the Largest Degree (LD.) Graph Heuristics, where exams with the largest number of conflicts are sorted to the top (Mandal and Kahar 2015). Next, the initial solution is generated by assigning exams to the first period without conflicts. If there are no available periods, the exam is allocated to the first period. If the solution still has conflicts, local search operators are used to changing the solution until there are no further conflicts. After a feasible solution is generated, stage 2 begins by using our modified GD algorithm.

3.1 Local Search Operators

Local search operators that we used in this paper are based on Bellio et al. (2021), which we modified according to the research model. Table 1 shows the move operators, their description, and preconditions.

Table 1. Move operators

Move operator	Description	Stage	Preconditions
MoveExam (ME.)	assigns exam e to period p	1	$\varphi(e) \neq p$; exam e is involved in a violation
SwapExam (SE.)	swaps the period assigned to e_1 and e_2	1	$\varphi(e_1) \neq \varphi(e_2)$; have students in common; either e_1 or e_2 is involved in a violation
KickExam1 (KE1)	assigns exam e_1 to $\varphi(e_2)$, and assigns $e_1\ to\ p$	1	e_1 and e_2 have students in common; e_1 is involved in a conflict
MoveExam2 (ME2)	assigns exam e to period p	2	$\varphi(e) \neq p$; exam e is involved in a distance cost; a move does not create conflict violations
KickExam2 (KE2)	assigns exam e_1 to $\varphi(e_2)$, and assigns $e_1\ to\ p$	2	e_1 and e_2 have students in common; e_1 is involved in a distance cost; both movements do not create conflict violations

Move operator	Description	Stage	Preconditions
DoubleKickExam (DKE)	assigns e_1 to $\varphi(e_2)$, e_2 to $\varphi(e_3)$, and e_3 to p	2	e_1 and e_2 have students in common; e_2 and e_3 have students in common; e_1 is involved in a distance cost; all three movements do not create conflict violations
KempeChain (KC.)	assigns exam e to period p; all exams in p in conflict with e are moved to $\varphi(e)$, all exams in conflict with them are moved to p, loop until there are no more conflicts	2	$\varphi(e) \neq p$; exam e is involved in a distance cost; an exam in p conflicts with e
SwapPeriods (SP.)	assigns all exams in period $p_1$to period p_2, and all exams in period $p_2$to period p_1	2	$p_1 \neq p_2$

3.2 Our Modified Great Deluge Algorithm

Our algorithm is based on the modified GD proposed by Kahar and Kendall (2015), with the main difference being in the step of increasing the water level. The algorithm begins by initializing the configurations, which are the desired value D to 80% of the current objective value, maximum iteration I, and the initial water level B to 1% of the initial objective value. Maximum iterations are set depending on the size of the instance. In the algorithm proposed by Kahar and Kendall (2015), rain speed ΔB is calculated from the difference between the initial water level and the desired value, divided by maximum iteration. However, we observed that the water level decreased too fast, so we further divided the rain speed by 0.25 to decrease the speed.

After the parameters have been initialized, a local search operator is chosen based on probability to generate a new solution. The new solution is accepted if it is better than the current solution or has a lower value than the current water level. If the new solution is better than the current best solution, then the current best solution is updated. A modification from the algorithm provided by Kahar and Kendall (2015) also happens after updating the current best solution, where B will be updated to the minimum value between the B or $1.01f(s)$. This is to increase the rain speed if there is a significant leap in the best solution. After updating solutions, the water level is lowered by the rain speed.

If there are no improvements within a certain number of iterations or the current solution reaches the desired solution, the current solution will be set to the best solution to try and search from that point. Another modification from Kahar and Kendall (2015) is also done here, where B will be set to $1.01f(s)$ means that if there are still enough iterations, the algorithm will continue to accept new search spaces since the water level won't dip below the current solution for too long. This will help the algorithm to escape from a local optimum.

ALGORITHM 1: Our Modified Great Deluge Algorithm

```
Initialize the configurations;
Set the initial solution $S$;
Calculate initial cost function $f(s)$;
Set the initial desired value $D$;
Set the number of iterations $I$;
Initial water level $B = 0.01f(s) + f(s)$;
Initialize rain speed $\Delta B = (B - D)/(0.25I)$;
Set $s_{best} = s$;
while not some stopping conditions do
        Choose local search operator based on probability;
        Calculate new solution $f(s*)$;
        If $f(s*) \leq f(s)$ $or$ $f(s*) \leq B$ then;
```

```
            Accept $s = s*$;
        If $f(s*) \leq f(s_{best})$ then;
            Accept $s_{best} = s*$;
            Set $B = \min(B,\ 1.01f(s))$;
        Lower the level B = B – ΔB;
        If no improvement in W iterations or $B \leq f(s_{best})$ or $f(s) \leq D$ then;
            Set $s = s_{best}$;
            Set $B = 1.01f(s)$;
        If $f(s) \leq D$ then;
            $D = 0.8f(s)$;
            Set new decay rate $\Delta B = (f(s) - D)/(0.25I_{remaining})$;
end
```

4. Results and Discussion

This paper's computation was implemented using Visual Studio Code (v. 1.64) using C++. The experiments were run on a computer with an AMD Ryzen 5 3600X Processor with a clock speed of 2.8GHz and 16 GB of RAM with a speed of 3200MHz.

4.1 Problem Instances

In a problem instance for a UUETP model, there need to be several parameters, which are the exams, students, and periods. Information is also needed about which students are entering which exams. The benchmark dataset used in this study is the Toronto Dataset taken from website created by Bellio et al. (2020). The dataset contains 13 examples UUETP instances formulated by Carter et al. (1996). The overview of the Toronto Dataset can be seen in Table 2.

Table 2. Carter's instances

Instances	Exams	Students	Periods	Conflict Matrix Density
car91	682	16925	35	0.13
car92	543	18419	32	0.14
ear83	190	1125	24	0.27
hec92	81	2823	18	0.42
kfu93	461	5349	20	0.06
lse91	381	2726	18	0.06
pur93	2627	30032	42	0.03
rye93	486	11483	23	0.08
sta83	139	611	13	0.14
tre92	261	4360	23	0.18
uta92	622	21266	35	0.13
ute92	184	2750	10	0.03
yor83	181	941	21	0.29

4.2 Parameter Configurations

The configuration and parameter initialization are the desired value D, maximum iteration I, rain speed ΔB, the initial water level B, and last improved iteration W. Each parameter's value has been explained in the previous section. As stated previously, the number of maximum iterations used varies between instances based on the size of the instance, and the probability of the move operators are based on Bellio et al. (2021) can be seen in Table 3.

Table 3. Probability of move operators at stage 2

Parameter	Definition	Probability
P_{ME2}	probability of *ME2* moves	0.5
P_{KC}	probability of *KC* moves	0.15
P_{KE}	probability of *KE2* moves	0.15
P_{DKE}	probability of *DKE* moves	0.05
P_{SP}	probability of *SP* moves	0.15

4.3 Comparison Results

The computational results can be seen in Tables 4, and 5. Instances are run through our modified GD algorithm ten times each. The solution and model constraints were verified and validated using a Microsoft Excel spreadsheet. We compared our algorithm with the Dueck GD and the modified GD (Dueck 1993; Kahar and Kendall 2015). These three algorithms are all programmed in C++ and run in the same machine to get a fair comparison. The average value results from the three algorithms and their gaps can be seen in Table 4, and the running time results can be seen in Table 5. Table 6 compares the solutions with results from best-known solution for UUETP.

Table 4. Results comparison (average) with other Great Deluge Algorithm

Instances	Our Modified GD	Dueck (1993)	% gap between (Dueck, 1993)	Kahar and Kendall (2015)	% gap between (Kahar and Kendall, 2015)
	Avg	Avg		Avg	
car91	6.80	8.13	-16.28%	7.42	-8.35%
car92	5.60	6.85	-18.22%	6.00	-6.76%
ear83	42.07	47.06	-10.61%	45.03	-6.58%
hec92	12.52	14.77	-15.28%	13.26	-5.57%
kfu93	16.61	22.34	-25.68%	18.52	-10.36%
lse91	14.85	17.59	-15.59%	15.85	-6.28%
pur93	7.79	9.29	-16.11%	8.59	-9.29%
rye93	13.21	17.52	-24.62%	14.25	-7.30%
sta83	158.59	161.92	-2.06%	161.58	-1.85%
tre92	10.22	11.87	-13.93%	10.49	-2.56%
uta92	4.48	4.90	-8.58%	4.79	-6.43%
ute92	29.78	35.88	-17.00%	32.94	-9.57%
yor83	42.56	47.66	-10.70%	44.76	-4.93%

Table 5. Results comparison (running time) with other Great Deluge Algorithm

Instances	Our Modified GD	Dueck (1993)	% gap between (Dueck, 1993)	Kahar and Kendall (2015)	% gap between (Kahar and Kendall, 2015)
	Time (s)	Time (s)		Time (s)	
car91	1371.09	1319.99	3.87%	1349.76	1.58%
car92	886.80	855.96	3.60%	868.39	2.12%
ear83	482.10	467.63	3.09%	469.62	2.66%
hec92	174.20	173.74	0.26%	174.06	0.08%
kfu93	644.04	630.38	2.17%	637.89	0.96%
lse91	487.96	473.97	2.95%	482.35	1.16%
pur93	1175.80	1180.60	-0.41%	1209.09	-2.75%
rye93	732.06	697.83	4.91%	716.27	2.20%

Instances	Our Modified GD	Dueck (1993)	% gap between (Dueck, 1993)	Kahar and Kendall (2015)	% gap between (Kahar and Kendall, 2015)
	Time (s)	Time (s)		Time (s)	
sta83	151.22	150.22	0.67%	149.83	0.93%
tre92	505.26	470.68	7.35%	476.13	6.12%
uta92	1135.58	1088.51	4.32%	1118.55	1.52%
ute92	243.84	238.07	2.42%	242.04	0.74%
yor83	235.12	229.09	2.64%	232.76	1.01%

From the results shown above, it is shown that our modified GD algorithm can generate solutions with no conflicts. When compared to the Dueck (1993) and the modified Kahar and Kendall (2015), our proposed method generates better solutions with competitive solve times, with an average objective value of 10.79% lower and computation time of 2.16% higher.

Table 6. Results comparison with best-known solution

Instances	Best Known Solution		references	Us		% gap of min value
	Min	Time (s)		Min	Time (s)	
car91	4.237932		Bellio et al. (2021)	6.78	134.17	59.96%
car92	3.642109		Bellio et al. (2021)	5.51	850.50	51.18%
ear83	32.420444		Bellio et al. (2021)	41.40	466.19	27.69%
hec92	10.033652	18000	Burke and Bykov (2016)	12.13	173.40	20.91%
kfu93	12.80	18360	Burke and Bykov (2016)	16.59	627.18	29.60%
lse91	9.773661		Bellio et al. (2021)	14.60	470.70	49.34%
pur93	3.88	20520	Burke and Bykov (2016)	7.62	1173.14	96.43%
rye93	7.837586		Bellio et al. (2021)	12.70	695.72	61.99%
sta83	156.86		Burke et al. (2010)	158.18	148.55	0.84%
tre92	7.590367		Bellio et al. (2021)	9.99	508.80	31.57%
uta92	2.947193		Bellio et al. (2021)	4.44	1073.68	50.79%
ute92	24.76		Bellio et al. (2021)	28.818	244.06	16.34%
yor83	34.404888		Bellio et al. (2021)	41.66	231.40	21.09%

However, the solutions given are still far from the current best-known solution for UUETP, where the gap of objective values varies between the instances. The smallest gap is in instance sta83 with a 0.84% difference, where sta83 is one of the smallest instances. The biggest gap is for instance pur93 with a 96.43% difference, where pur93 is one of the largest instances. This aligns with the fact that larger instances are more complex. Results also suggest that improvements to the solution could be done with better parameter tuning and a more efficient code.

4.4 Parameter Testing

Parameter testing is also done using analysis of variance to check parameters that have significant effects on the proposed algorithm. There are four parameters that are tested, each having three levels. The initial normality test indicates that the residuals are normally distributed, thus fulfilling the normality assumption. Table 7 shows the levels and p-values of the tested parameters. The analysis suggests that the tested parameters have no statistical significance towards the means of the objective value. Furthermore, interactions between factors also show p-values larger than 0.05, suggesting no statistical significance. Therefore, it can be concluded that the suggested model can be used with minimal parameter tuning.

Table 7. Parameter testing using analysis of variance

Source	Level	DF	Adj SS	Adj MS	F-Value	P-Value
Water Level Multiplier	[1%, 3%, 5%]	2	0.01922	0.009612	0.82	0.441
Desired Level Multiplier	[65%, 80%, 95%]	2	0.01147	0.005733	0.49	0.613
Last Improved Iteration	[30, 50, 70]	2	0.02228	0.011140	0.96	0.387
Rain Speed Multiplier	[1, 0.25, 0.125]	2	0.00236	0.001182	0.10	0.904
Error		126	1.46946	0.011662		
Lack-of-fit		16	0.16556	0.010347	0.87	0.601
Pure Error		110	1.30391	0.011854		
Total		134	1.52480			

6. Conclusion

This research proposes a great deluge algorithm to solve the Uncapacitated Examination Timetabling problem. Results suggest that in its current state, the algorithm can generate valid solutions. We also compared the solution of our proposed algorithm with the solutions from previous works. We believe our proposed algorithm can be further improved in future research. One way of improvement would be to write a more efficient code for the algorithm. Another avenue of research could be parameter tuning for each specific instance. We would also like to compare this algorithm directly with a SA algorithm in the future. Applying the algorithm with a real-world dataset would also be a possible future research topic.

References

Abayomi-Alli, A., Misra, S., Fernández-Sanz, L., Abayomi-Alli, O., & Edun, A. R., Genetic algorithm and tabu search memory with course sandwiching (Gats_cs) for university examination timetabling, *Intelligent Automation and Soft Computing*, vol. 26, no. 3, 2020

Aldeeb, B. A., Al-betar, M. A., Abdelmajeed, A. O., & Younes, M. J., A Comprehensive Review of Uncapacitated University Examination Timetabling Problem, *International Journal of Applied Engineering Research*, vol.14, no. 24, 2019

Bellio, R., Ceschia, S., di Gaspero, L., & Schaerf, A, Uncapacitated Examination Timetabling, https://opthub.uniud.it/problem/timetabling/edutt/ett/uett, Accessed on February, 2022

Bellio, R., Ceschia, S., di Gaspero, L., & Schaerf, A., Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling, *Computers and Operations Research*, vol. 132, 2021

Burke, E., & Bykov, Y., *Solving exam timetabling problems with the flex-deluge algorithm*, 370–372, 2006

Burke, E., Bykov, Y., Newall, J., & Petrovic, S., A time-predefined local search approach to exam timetabling problems, *IIE Transactions (Institute of Industrial Engineers)*, vol. 36, no. 6, 2004

Burke, E. K., & Bykov, Y., An adaptive flex-deluge approach to university exam timetabling, *INFORMS Journal on Computing*, vol. 28, no. 4, 2016

Burke, E. K., Eckersley, A. J., McCollum, B., Petrovic, S., & Qu, R., Hybrid variable neighbourhood approaches to university exam timetabling, *European Journal of Operational Research*, vol. 206, no. 1, 2010

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R., A graph-based hyper-heuristic for educational timetabling problems, *European Journal of Operational Research*, vol. 176, no. 1, 2007.

Carter, M. W., Laporte, G., & Lee, S. Y., Examination timetabling: Algorithmic strategies and applications, *Journal of the Operational Research Society*, vol. 47, no. 3, 1996.

Dewi, S., Tyasnurita, R., & Pratiwi, F. S., Solving examination timetabling problem within a hyperheuristic framework, *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, 2021.

Dueck, G., New optimization heuristics; The great deluge algorithm and the record-to-record travel, *Journal of Computational Physics*, vol. 104, no. 1, 1993.

Kahar, M. N. M., & Kendall, G., A great deluge algorithm for a real-world examination timetabling problem, *Journal of the Operational Research Society*, vol. 66, no. 1, 2015.

Khair, A. F., Makhtar, M., Mazlan, M., Mohamed, M. A., & Rahman, M. N. A., A study on university course and exam timetabling problems and methods: An optimization survey, *International Journal of Engineering and Technology(UAE)*, vol. 7 ,no. 2.14, 2018a

Khair, A. F., Makhtar, M., Mazlan, M., Mohamed, M. A., & Rahman, M. N. A., Solving examination timetabling problem in UniSZA using ant colony optimization, *International Journal of Engineering and Technology(UAE)*, vol. 7, no. 2, 2018b.

Leite, N., Melício, F., & Rosa, A. C., A fast simulated annealing algorithm for the examination timetabling problem, *Expert Systems with Applications*, vol. 122, 2019.

Leite, N., Neves, R., Horta, N., Melício, F., & Rosa, A. C., Solving a capacitated exam timetabling problem instance using a bi-objective NSGA-II, *Studies in Computational Intelligence*, vol. 577, 2014.

Mandal, A. K., & Kahar, M. N. M., Solving examination timetabling problem using partial exam assignment with great deluge algorithm, *I4CT 2015 - 2015 2nd International Conference on Computer, Communications, and Control Technology, Art Proceeding*, Kuching, Malaysia, April 21-23, 2015.

Mandal, A. K., & Kahar, M. N. M., Performance analysis of graph heuristics and selected trajectory metaheuristics on examination timetable problem, *Indonesian Journal of Electrical Engineering and Informatics*, vol. 8, no. 1, pp. 163–177, 2020.

Mandal, A. K., Kahar, M. N. M., & Kendall, G., Addressing examination timetabling problem using a partial exams approach in constructive and improvement, *Computation*, vol. 8, no. 2, 2020.

Muklason, A., Syahrani, G. B., & Marom, A., Great deluge based hyper-heuristics for solving real-world university examination timetabling problem: New data set and approach, *Procedia Computer Science*, vol. 161, 2019.

Pinedo, M. L., *Scheduling: Theory, Algorithms, and Systems,* Springer, New York, 2008

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T. G., & Lee, S. Y., A survey of search methodologies and automated system development for examination timetabling, *Journal of Scheduling*, vol. 12, no. 1, 2009.

Schmidt, G., & Stroehlein, T., Timetable Construction - An Annotated Bibliography, *Computer Journal*, vol. 23, no. 4, 1980.

Biographies

Lnasya Syafitrie. S.T, is currently a master's student studying Systems, Design, and Management at Department of Industrial Engineering, Faculty of Engineering, Universitas Indonesia. Lnasya graduated from the same department for bachelor's degree in Industrial Engineering. Her research interests include optimization and scheduling.

Dr. Komarudin, S.T, M.Eng, is a Lecturer and Head of Department Industrial Engineering, Faculty of Engineering, Universitas Indonesia. Komarudin graduated from Vrije Universiteit Brussel. His research interest are optimization, mathematical programming, facility layout, scheduling, game theory, and many others. He currently researches the applications and developments of optimization methods and algorithms for solving industrial problems. He has published his research papers in, among others, European Journal of Operations Research, Annals of Operations Research and Journals of Operational Research Society, and many International Conferences.