Priority Scheduling Implementation for Exam Schedule

M I Yanwari[*1], A S Prabuwono[2], T R Yudantoro[3], N B Aji[4], Wiktasari[5], S Handoko[6]

[1,3-5]Computer Engineering Technology, Politeknik Negeri Semarang

[2]King Abdulaziz University, Saudi Arabia

[6]Informatics Engineering, Politeknik Negeri Semarang

E-mail: irwan.yanwari@polines.ac.id[1], antonsatria@eu4m.eu[2], tryudan@polines.ac.id[3], bayu.nurseno@polines.ac.id[4], wiktasari@polines.ac.id [5], kang.handoko@polines.ac.id[6]

Submitted: 5 November 2022, revised: 2 February 2023, accepted: 6 February 2023

Abstract. Scheduling is a common problem that has been raised for a long time. Many algorithms have been created for this problem. Some algorithms offer flexibility in terms of constraints and complex operations. Because of that complexity, many algorithms need huge computation resources and execution time. A platform like a web application has many restrictions such as execution time and computation resources. A complex algorithm is not suited for the web application platform. Priority scheduling is a scheduling algorithm based on a priority queue. Every schedule slot will produce a queue based on the constraints. Each constraint will have a different weight. The weight in the queue represents their priority. This algorithm provides a light algorithm that only needs a few computations and execution times. The exam schedule is one of many problems in educational institutions. A web application is a popular platform that can be accessed from everywhere. Many educational institutions use web platforms as their main system platform. Web platforms have some restrictions such as execution time. Due to web platform restrictions, priority scheduling is a suitable algorithm for this platform. In this study, the author implemented a priority scheduling algorithm in scheduling cases with a website platform and the study showed that this algorithm solution could be an alternative for solving scheduling cases with low computational resources.

Keywords: exam schedule; web platform; scheduling algorithm; priority schedule.

1. Introduction

Scheduling is one of the classic problems that has been raised for a long time. There are many algorithms created to solve scheduling problems. Each algorithm is created specifically for a certain condition. In cloud-based scheduling, conventional job scheduling algorithms cannot be used in cloud environment problems [1]. In other cases, the cloud-based algorithm is difficult to be implemented in a conventional case. Each algorithm will also have a different goal to accomplish. In a company or industry's scheduling, the goal is to improve efficiency, decision-making and process control, and better customer service [2]. This article discusses several algorithms that can be implemented for scheduling cases. Besides that, there is also an implementation of a priority scheduling algorithm as a solution that the author considers efficient and suitable for web platforms.

2. Scheduling Algorithm

There are several algorithms created to solve scheduling problems. Each algorithm is driven to solve a specific problem. One of the scheduling problems is finding the optimal schedule [2]. Finding an optimal schedule is very challenging because each schedule is different according to its domain and constraints.

2.1. Failure Aware Workflow Scheduling (FAWS / FLAWS)

Sometimes, a schedulling algorithm can fail to alocate new job in a new resource. If this kind of failure hapenned, the algorithm needs to alocate that job to another available resource. FAWS / FLAWS algorithm is designed to reduce the lost time caused by placing jobs on a resource to fail in the middle of job execution [3]. By avoiding that failure, FLAWS also reduces the number of job rescheduling. FLAWS is designed to avoid job assignment failure. However, it does not mean it is 100% reliable. Some research is made to solve that problem by enhancing the algorithm [4]. Since the algorithm is robust and fault-tolerant, this algorithm is suitable for cloud platforms. Moreover, some studies are also trying to improve this algorithm specifically for cloud-based systems [6].

2.2. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) algorithm was introduced by Kennedy and Eberhart in 1995 [7]. PSO is an evolutionary algorithm that imitated the social behavior of a bird flock [8]. PSO starts with early solutions and updates the solutions from iteration to iteration [8]. The PSO algorithm has been used in different search stages, and these stages are close to the optimum stage with their pbest and gbest values. PSO is suitable for continuous and discrete problems and PSO is also good for global searches in the problem space [9]. PSO is weak for local searches, with a large possibility of being trapped in a local optimum in the last iteration [9]. PSO converges globally because it searches globally. It always tries to find the solutions that have better fitness functions in a stochastic search problem space. Some research is trying to improve this algorithm [10]. The particle swarm optimization (PSO) algorithm can be split into three steps: the first step is to initialize each particle’s position and velocity, the next is to update the velocity, and lastly is to update the particle’s position [11].

2.3. Genetic Algorithm (GA)

Genetic algorithm (GA) is an algorithm that attempts to simulate natural evolution to problem-solving tasks [12]. GA was first introduced by Holland in 1970 [13]. This algorithm randomly combines various optimal solutions in a collection to get the next best solution generation, namely in a condition that maximizes its suitability and conforms to predetermined limits [12]. GA utilizes a representation based on groups to model a schedule [13]. Gens in GA are used to model time spaces in which jobs can be scheduled [13]. Each gen has a limited length that cannot be exceeded. A chromosome is an instance that contains a group of genes, and each chromosome represents a possible schedule of activities. GA is a popular metaheuristic algorithm for solving the job-shop scheduling problem (JSP) and flexible job-shop scheduling problem (FJSP) in recent years [14]. To apply GA as an algorithm, a solution to a specific problem must be encoded in the form of a string, known as a chromosome [14].

2.4. Ant Colony Optimization (ACO)

Ant colony algorithm (ACO) is an algorithm for finding optimal paths that are based on the behavior of ants searching for food [15]. Ant behavior when searching for food is the ants initially wander randomly. When an ant finds food, it walks back to the colony while spreading "markers" (pheromones) that show the path to the food [15]. When the other ants come across the markers, they will follow the path with a certain probability. Then they will populate the path with their markers as they bring the food back to the colony [15]. As more ants find the path, it gets stronger until there is a stream of ants traveling to various food sources near the colony [15]. To use an ant algorithm for scheduling, it is necessary to define the problem in a graph [16]. The main advantage of the Ant System is that it easily deals with combinatorial optimization problems defined on a non-symmetric graph [16]. The main advantage of the Ant System is that it easily deals with combinatorial optimization problems defined on a non-symmetric graph [16]. The only adaptation to be made for dealing with non-symmetry is the expansion of the pheromone table. This increases its spatial complexity (memory) but does not necessarily require extra computational power [16].

2.5. Graph Coloring Approach

A scheduling problem can be modeled by a graph where the vertices and edges represent the courses and the common students, respectively [17]. The graph coloring algorithm gives rise to variations that have all proved to be NP-complete, such as the saturation algorithm, the Recursive Largest First (RFL) algorithm, degree of saturation algorithm, simulated annealing algorithm, and greedy algorithm [17]. There are orders in which exams are selected are based on the graph coloring approach [18]. In the graph coloring approach, each item being scheduled is represented by a different vertex where the edges between vertices represent conflicting items [18]. Coloring the graph is the process of allocating the different colors to each vertex so that two adjacent vertices will have different colors and each color is equivalent to one period in the exam timetable [18].

2.6. Priority Scheduling

Priority scheduling is a scheduling algorithm derived from the CPU scheduling problem [19]. When several tasks are scheduled by the CPU, every task can have different priorities. Priority scheduling will execute the schedule with higher priority tasks first. In Priority scheduling, each task is given a priority number [19]. The priority of the processes that occur on the processor is something that determines when the process will be done [20].

3. Comparison between Algorithms

Every algorithm has its advantage and disadvantage. For example, Genetic Algorithm is well known for providing precise results and low error rates. On the other hand, this algorithm requires considerable computational resources. The level of resource usage by each algorithm can be seen in Table 1.

Table 1. Scheduling algorithms comparison

No	Algorithms	Computation Resource
1	Failure Aware Workflow Scheduling	Low - Medium
2	Particle Swarm Optimization (PSO)	Medium - High
3	Genetic Algorithm (GA)	Medium - High
4	Ant Colony Optimization (ACO)	High
5	Graph Colouring Approach	Low - Medium
6	Priority Scheduling	Low - Medium

4. Exam Schedule Constraints

The constraints used in scheduling problems developed for the educational institution exam can be divided into two groups [21] :

- Hard constraints, constraints that express the conditions that must be met while obtaining a solution.

- Soft constraints, conditions that are desired to be met even if they do not contain any obligation.

Each constraint type has a different treatment. Unlike soft constraints which only affect queue weight in positive weight points, hard constraint weight affects queue weight in positive and negative weight points depending on queue item whether it satisfies the hard constraints or not. If the queue item satisfies a hard constraint, a positive point will be added to the item weight. If the queue item does not satisfy a hard constraint, a negative point will be added to the item weight.

In this case, all constraints were from Electrical Engineering Department in Politeknik Negeri Semarang. It can be different for other cases. The hard constraints in this study were:

- General courses with similar content would be held at the same time.
- Every class would only have one exam a day.
- Every tutor would only do one supervising exam course at a time.

Meanwhile, the soft constraints in this study were:

- Every tutor would supervise his/her class.
- If the tutor had more than one class for a subject, he/she would supervise one class and the other free tutor would supervise the other class.
- Every tutor would supervise a subject from his/her Homebase.
- Every tutor needed to have a similar workload.

5. Method

Based on previous work in scheduling problem-solving and their result, this study proposed a model suited for exam scheduling with given constraints. At the beginning of the implementation, all constraints were defined based on the Study Case. The Constraints consist of Hard Constraints and Soft Constraints as described in point 4 (Exam Schedule Constraints). Then the preparation of sessions for each exam scheduled is carried out. After the algorithm calculated the priority based on the course queue, the course is distributed to available sessions. The same step is carried out to determine the supervisor for every exam scheduled. After that, The distribution of supervisors is calculated for every supervisor. The work steps carried out are depicted in Figure 1.

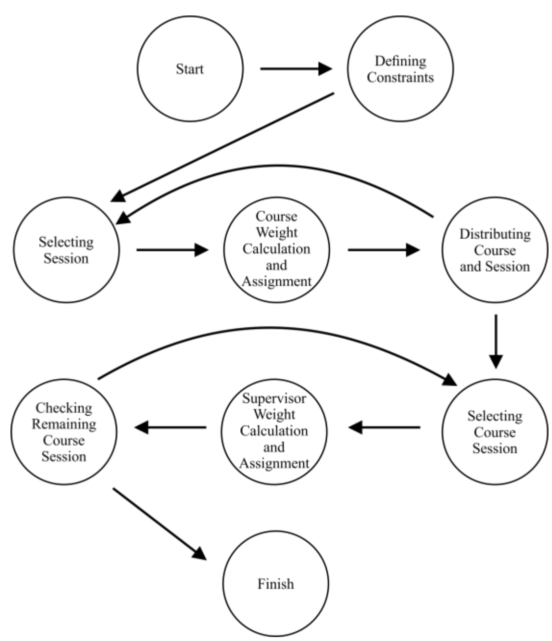

Figure 1. Exam scheduling model

5.1. Defining Constraints

Defining constraints is a stage where all constraints are sorted from the most important to less important and then divided into hard and soft constraints. The constraints are also divided into Constraints for Course and Constraints For Supervisor. The Constraints for Course is a list of constraints that describes what kind of courses will take priority over other courses. The Constraints for Course can be seen in Table 2. The Constraints for Supervisor is a list of constraints that describes which supervisor will take priority over other supervisors. The Constraints for Supervisor can be seen in Table 3.

Table 2. Constraints for Course

Code	Constraints	Type	Weight
C11	General Courses with similar content will be held at the same time.	Hard Constraint	2
C12	Every class will only have one exam a day.	Hard Constraint	1

Table 3. Constraints for Supervisor

No	Constraints	Type	Weight
C21	Every tutor will only do one supervising exam course at a time	Hard Constraint	5
C22	Every tutor will supervise his/her class.	Soft Constraint	4
C23	If the tutor has more than one class for a subject, he/she will supervise one class and the other free tutor will supervise the other class	Soft Constraint	3
C24	Every tutor will supervise a subject from his/her home base.	Soft Constraint	2
C25	Every tutor needs to have a similar workload.	Soft Constraint	1

5.2. Selecting Session

The session shows when the exam is carried out and the day and time of the exam. Each session will bring up a queue of courses to be scheduled. Each queue will be calculated in the next step. Table 4 illustrates how session selection is carried out.

Table 4. Sessions

No	Day	Session	
1	1	1	Course Queue to be Assigned
2	1	2	Course Queue to be Assigned
3	2	1	Course Queue to be Assigned
4	2	2	Course Queue to be Assigned
…	…	…	Course Queue to be Assigned
	n	1	Course Queue to be Assigned
	n	2	Course Queue to be Assigned

5.3. Course Weight Calculation and Assignment

Weight calculation in the priority queue is a summation operation. The Summation is executed for each constraint weight. If in the first loop one course has been assigned to one session, then in the next loop, that course will not be calculated again. The result of that assignment will affect the other course weight. Table 5 illustrates how Course Weight Calculation and Assignment is carried out.

Table 5. Weight Calculation for Day 1 Session 1 Room 2

No	Student Entry Year	Course	Session	Room	Constraints	Weight
1	2020	Math	…		- C12	-1
2	2020	English	Day 1 Session 1	Room 1		
3	2020	Algorithm and Programming	…		- C12	-1
4	2021	English	…		C11 + C12	**3**
…	2021	IoT	…		C12	1
	2021	Embedded System	…		C12	1
	…	…	…		…	

5.4. Distributing Course and Sessions

After calculating the weights for each course, the course with the highest weight points will represent the item with the highest priority to be assigned to that session. For example, in Table 5 the English Course for students with the entry year 2021 has a higher point weight compared to other courses. From this calculation, on Day 1 session 1 room 2 will be allocated for English language course for students with the entry year 2021.

5.5. Selecting Course Session

Just like how a course is assigned to an exam session. When a supervisor needs to be assigned to an exam course session, a calculation for the supervisor queue is also needed in the process. Before the weight calculation, a slot for the supervisor queue is important. Table 6 illustrates how Course Session Selection is carried out.

Table 6. Supervisor Queue

Code	Student Entry Year	Course	Session	Room	
S1	2020	Math	Day 2 Session 1	Room 1	Supervisor Queue to be Assigned
S2	2020	English	Day 1 Session 1	Room 1	Supervisor Queue to be Assigned
S3	2020	Algorithm and Programming	Day 3 Session 1	Room 1	Supervisor Queue to be Assigned
S4	2021	English	Day 1 Session 1	Room 2	Supervisor Queue to be Assigned
…	2021	IoT	…	…	Supervisor Queue to be Assigned
	2021	Embedded System	…	….	Supervisor Queue to be Assigned
	…	…	…	…	…

5.6. Weight Calculation and Assignment

To assign a supervisor, the total weight of each supervisor needs to be calculated. In a case where a course session code is S3 with the course lecturer is “Irwan”, the calculation result will be similar to Table 7. Table 7 illustrates how Weight Calculation and Assignment is carried out.

Table 7. Supervisor Weight Calculations

Code	Lecturer	Supervisor	Workload	Constraints	Weight
S3	Muhammad Irwan Yanwari	Muhammad Irwan Yanwari	2	C21 + C22 + C24	11
S3	Muhammad Irwan Yanwari	Nurseno Bayu Aji	1	C21 + C24 + C25	8
S3	Muhammad Irwan Yanwari	Wiktasari	3	C21 + C24	7
S3	Muhammad Irwan Yanwari	Afandi Nur Aziz Thohari	2	C21 + C24	7
…	Muhammad Irwan Yanwari	Kuwat Santoso	3	C21 + C24	7
…	Muhammad Irwan Yanwari	Amran Yobi Oktabera	1	C21 + C24 + C25	8
…	…	…	…	…	…

Table 7 shows the highest weight to be assigned as supervisor is “Irwan”, therefore “Irwan” will be assigned to supervise the Algorithm exam for the 2020-year students in day 3 session 1 room 1.

5.7. Checking Remaining Course Session

After one assignment, it is important to check if there is still a course with no assignment yet. If there are still queue items in waiting, then another calculation will be performed once more. If the queue is empty, it means the scheduling is finished.

6. Implementation Result

In the implementation of the priority scheduling algorithm, the research constraint for the implemented case is embedded in the system code. The final exam was successfully scheduled as it should be (the scheduling result can be seen in figure 4). To represent the problem discussed in the case study, resource and code execution time restrictions were applied, such as the following:

- Web platform is used in this study
- Max execution time for the server is 30 seconds
- The processor is 4 core processor AMD Ryzen 7 3750H with base speed 2.30GHz
- 16GB of RAM

The data sample used in this research is 30 courses from the Computer Technology Engineering study program and Informatics study program in Polines which is input via the dashboard (the input form can be seen in figure 2). The scheduling algorithm is executed on a page dedicated to the scheduling purpose (the page can be seen in figure 3). To lower the resource consumption even further the data is executed in groups consisting of data from each study program. With implementation restricted to one study program executed at a time, the server workload becomes lighter because each study program only had 3 student batches with courses numbered from 8 to 10 (less than 30 courses scheduled at a time). By executing the process in groups, resource consumption is reduced significantly. By doing that, the problem of resource limitations is optimized even further.

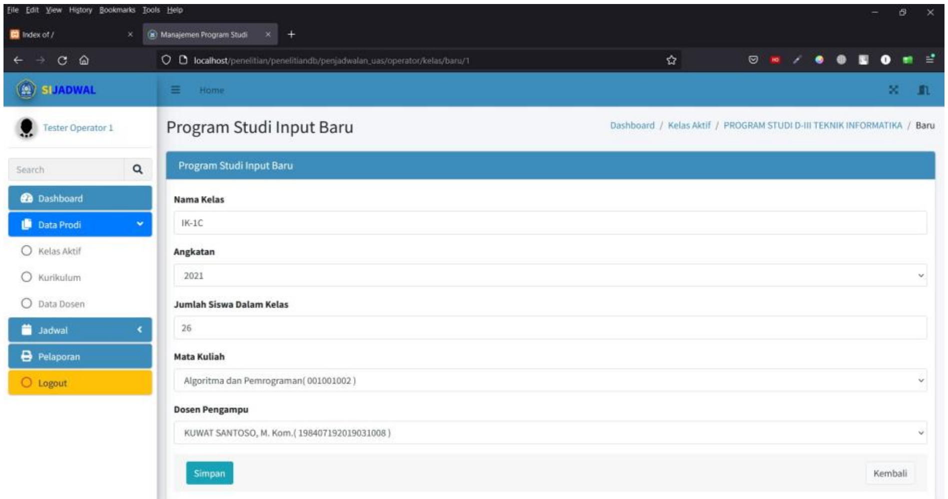

Figure 2. Active Class Form

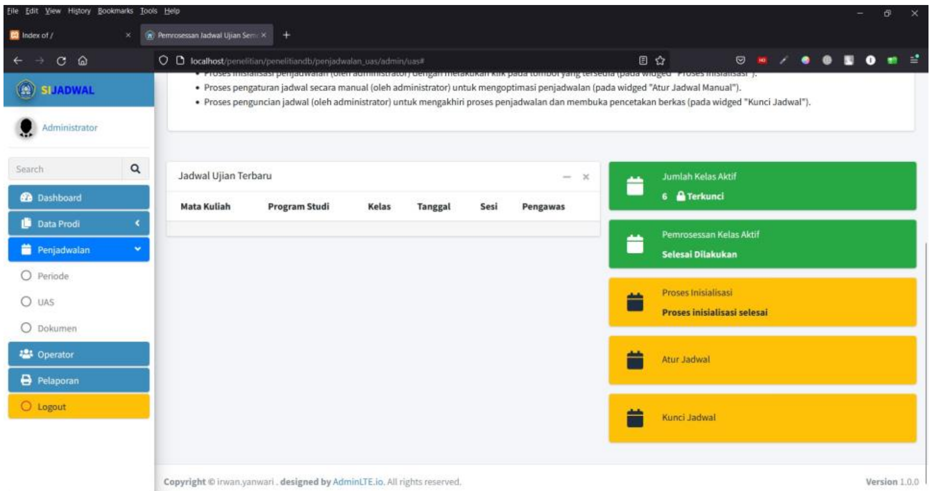

Figure 3. Scheduling Page

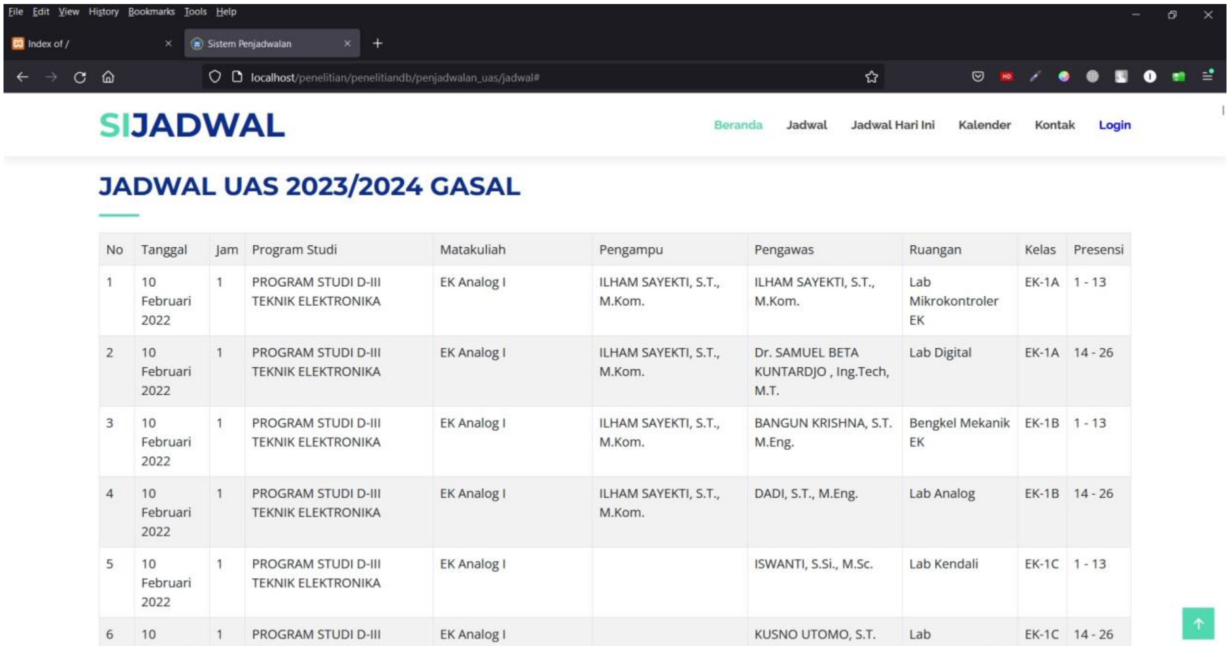

Figure 4. Scheduling result

7. Conclusions

In this study, the priority queue basic concept is used to schedule exam sessions. With the maximum execution time for the server-side script of 30 seconds, this algorithm successfully scheduled 30 courses with given constraints. This research used a computer with 4 core processor with a base speed 2.30GHz to represent the limit of computation resource and execution time. By standard, this computation power can be considered low because high-end computers used 16 core processors with a speed of 3.6 to 5.00 GHz. In this study, it was found that:

- Scheduling of Course Examinations and Examination Supervisors need to be calculated separately.
- The order of the constraints affects the calculation result drastically.
- To schedule a huge number of courses, the data need to be executed in the group.

8. Acknowledgement

Special thanks are given to the Politeknik Negeri Semarang which has funded this research and encouraged the publication of this article in IJIS.

References

[1] A. Bhatt, P. Dimri, and A. Agarwal, “A study on various job scheduling algorithms in Multi-Cloud Environment,” vol. 8, no. 12, pp. 77–89, 2017.

[2] M. A. Aziz and I. H. Ninggal, “Scalable workflow scheduling algorithm for minimizing makespan and failure probability,” *Bull. Electr. Eng. Informatics*, vol. 8, no. 1, pp. 283–290, 2019, doi: 10.11591/eei.v8i1.1436.

[3] Z. Yu, C. Wang, and W. Shi, “FLAW : FaiLure-Aware Workflow Scheduling in High Performance Computing Systems,” *J. Clust. Comput.*, vol. 13, no. 4, pp. 421–434, 2010.

[4] M. A. Aziz, J. H. Abawajy, and M. Chowdhury, “Scheduling workflow applications with Makespan and reliability constraints,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 12, no. 2, pp. 482–488, 2018, doi: 10.11591/ijeecs.v12.i2.pp482-488.

[5] S. Chitra, “Compensatory Aggregation based Failure Aware Cloud Workflow Scheduling,” *2nd Int.*

Conf. Innov. Mech. Ind. Appl. ICIMIA 2020 - Conf. Proc., no. Icimia, pp. 41–46, 2020, doi: 10.1109/ICIMIA48430.2020.9074964.

[6] M. Farid, R. Latip, M. Hussin, and N. A. W. A. Hamid, "A fault-intrusion-tolerant system and deadline-aware algorithm for scheduling scientific workflow in the cloud," *PeerJ Comput. Sci.*, vol. 7, pp. 1–21, 2021, doi: 10.7717/PEERJ-CS.747.

[7] B. Niu, F. Zhang, L. Li, and L. Wu, "Particle Swarm Optimization for Yard Truck Scheduling in Container Terminal with a Cooperative Strategy," pp. 333–346, 2017, doi: 10.1007/978-3-319-49049-6_24.

[8] J. Joy, S. Rajeev, and V. Narayanan, "Particle Swarm Optimization for Resource Constrained-project Scheduling Problem with Varying Resource Levels," *Procedia Technol.*, vol. 25, no. Raerest, pp. 948–954, 2016, doi: 10.1016/j.protcy.2016.08.185.

[9] A. Al-Maamari and F. A. Omara, "Task scheduling using PSO algorithm in cloud computing environments," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 5, pp. 245–256, 2015, doi: 10.14257/ijgdc.2015.8.5.24.

[10] P. Pongchairerks, "Particle swarm optimization algorithm applied to scheduling problems," *ScienceAsia*, vol. 35, no. 1, pp. 89–94, 2009, doi: 10.2306/scienceasia1513-1874.2009.35.089.

[11] G. Christopher and A. Wicaksana, "Particle swarm optimization for solving thesis defense timetabling problem," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 19, no. 3, pp. 762–769, 2021, doi: 10.12928/TELKOMNIKA.v19i3.18792.

[12] P. Ulum and D. Fitriani, "Implementation of Genetic Algorithm in the Current Scheduling System," *J. Ris. Inform.*, vol. 3, no. 2, pp. 119–126, 2021, doi: 10.34288/jri.v3i2.167.

[13] G. Rivera, L. Cisneros, P. Sánchez-Solís, N. Rangel-Valdez, and J. Rodas-Osollo, "Genetic algorithm for scheduling optimization considering heterogeneous containers: A real-world case study," *Axioms*, vol. 9, no. 1, 2020, doi: 10.3390/axioms9010027.

[14] J. S. Park, H. Y. Ng, T. J. Chua, Y. T. Ng, and J. W. Kim, "Unified genetic algorithm approach for solving flexible job-shop scheduling problem," *Appl. Sci.*, vol. 11, no. 14, 2021, doi: 10.3390/app11146454.

[15] V. SELVI and R. UMARANI, "An Ant Colony Optimization For Job Scheduling To Minimize Makespan Time," *Int. J. Comput. Commun. Technol.*, vol. 6, no. 1, pp. 13–15, 2015, doi: 10.47893/ijcct.2015.1266.

[16] M. Seo and J. A. Ventura, "Ant colony optimization for job shop scheduling," *IIE Annu. Conf. Expo 2008*, pp. 1433–1438, 2008.

[17] A. Dandashi and M. Al-Mouhamed, "Graph coloring for class scheduling," *2010 ACS/IEEE Int. Conf. Comput. Syst. Appl. AICCSA 2010*, pp. 2–5, 2010, doi: 10.1109/AICCSA.2010.5586963.

[18] B. Hussin, A. S. H. Basari, A. S. Shibghatullah, S. A. Asmai, and N. S. Othman, "Exam timetabling using graph colouring approach," *2011 IEEE Conf. Open Syst. ICOS 2011*, pp. 133–138, 2011, doi: 10.1109/ICOS.2011.6079274.

[19] N. Parmar and A. Joshi, "Billing Queue Scheduling Using Priority Scheduling and Shortest Job First Scheduling Algorithm," pp. 64–67, 2017.

[20] A. P. U. Siahaan, "Comparison Analysis of CPU Scheduling FCFS, SJF and Round Robin," *Int. J. Eng. Dev. Res.*, vol. 4, no. 3, pp. 125–131, 2017.

[21] M. S. Başar and S. Kul, "A student-based central exam scheduling model using A* algorithm," *Open Comput. Sci.*, vol. 12, no. 1, pp. 181–190, 2022, doi: 10.1515/comp-2022-0237.