View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by TELKOMNIKA (Telecommunication Computing Electronics and Control)

TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 3, June 2021, pp. 762~769
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i3.18792

Particle swarm optimization for solving thesis defense timetabling problem

Gilbert Christopher, Arya Wicaksana
Department of Informatics, Universitas Multimedia Nusantara, Indonesia

Article Info

Article history:

Received Aug 13, 2020
Revised Nov 15, 2020
Accepted Nov 25, 2020

Keywords:

Optimization
Particle swarm optimization
Scheduling
Timetabling

ABSTRACT

The thesis defense timetabling problem is a fascinating and original NP-hard optimization problem. The problem involves assigning the participants to defense sessions, composing the relevant committees, satisfying the constraints, and optimizing the objectives. This study defines the problem formulation that applies to Universitas Multimedia Nusantara (UMN) and use the particle swarm optimization (PSO) algorithm to solve it. As a demonstration of concept and viability, the proposed method is implemented in a web-based platform using Python and Flask. The implementation is tested and evaluated using real-world instances. The results show that the fastest timetable generation is 0.18 seconds, and the slowest is 21.88 minutes for 25 students and 18 department members, without any violation of the hard constraints. The overall score of the EUCS evaluation for the application is 4.3 out of 6.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Arya Wicaksana
Department of Informatics
Universitas Multimedia Nusantara
Scientia Boulevard St., Gading Serpong, Tangerang-15810, Banten, Indonesia
Email: arya.wicaksana@umn.ac.id

1. INTRODUCTION

The thesis defense is a mandatory activity to be taken by students in Universitas Multimedia Nusantara (UMN). Indonesian ministry of education and culture regulates Indonesian universities to have this activity in their 4 years undergraduate curriculum (bachelor). In general, there exists at least three participants in a defense session: the student, the supervisor, and the examiner. In UMN, specifically in the Department of Informatics, there are four participants, including the moderator of the session. Thus, the size of the department and the number of students determine the time needed to create the timetable for the thesis defense sessions. Related work in [1] uses local search, integer programming (IP), and constraint programming (CP) for comparison in solving the thesis defense timetabling problem that applies to some Italian universities. Based on the experimental analysis, it is also found that the problem is solvable in NP-complete, and according to the personnel involved, it has reduced the time spent in the overall procedure by several worker-days for each graduation period.

In this work, the particle swarm optimization (PSO) algorithm introduced by J. Kennedy and R. Eberhardt is used [2]. In the PSO algorithm, the population is called a swarm, and each individual is called particle [3]. In [4, 5], the PSO algorithm successfully optimizes and solves the scheduling problems with multiple constraints. The PSO algorithm has excellent robustness and useful in different application environments with little modification [6]. The PSO algorithm also delivers the same optimal solution than other algorithms with faster computing time and a faster convergence rate than other algorithms, such as the genetic

algorithm [7]. PSO algorithm also successfully implemented in some computer science problem, such as knapsack problem [8, 9] and job-shop problem [10, 11] and some real-life cases, such as optimization of reservoir operation [12], task scheduling in grid [13, 14] resource-constrained project scheduling [15], cloud computing [7, 16, 17], and employee scheduling [18].

Scheduling is allocating resources in a specific time to produce or finish a task. The scheduling problem is a complex combinatorial problem because there is more than one solution and is locally optimal. Scheduling problem is classified as a non-deterministic polynomial-time hard (NP-Hard) problem. In scheduling problem, there are two types of constraint: a hard constraint and soft constraint. The hard constraint is a constraint that cannot be violated, and soft constraint is a constraint that can be violated. However, the violation must be minimized to get the optimal solution [19].

This paper defines the problem formulation that applies to the Department of Informatics at Universitas Multimedia Nusantara. The fitness functions tailored to the problem formulation are developed with both hard and soft constraints. The goal is to try to optimize the timetabling process by minimizing the soft constraints violations. The proposed approach is implemented in a web-based platform using the Python programming language and the Flask framework. The application is tested using real-world instances and evaluated using the end-user computing satisfaction (EUCS) with a 6-point Likert scale [20, 21]. The rest of this paper is organized as follows; section 2 briefly describes the research method used for this study, including problem formulation, particle swarm optimization (PSO), and the design and implementation work. Section 3 describes the results of the study and the performance evaluation. Section 4 concludes this paper with some discussions on future work.

2. RESEARCH METHOD

2.1. Problem formulation

In Universitas Multimedia Nusantara (UMN), thesis defense sessions are allocated into two weeks per batch. Within a single batch, the number of students going for their thesis defense is not limited by the department. In the Department of Informatics at UMN, 18 department members are all eligible to be the committee of the sessions. Sessions are meetings where students defend their thesis in front of a committee, and some sessions might overlap in time [1]. Faculty members are characterized by their academic level and research areas. Students are allowed to have at least one supervisor and at most two supervisors. The department assigns the examiner and moderator of the sessions. Another consideration is the quota for each department member to become a committee of a session. Department member that holds a position in the university is limited to a lower number of sessions.

As customary, constraints are divided into hard and soft ones. The former must always be satisfied, whereas the latter compose the objective function that is optimized (minimized) during each iteration in the PSO. There is only one hard constraint that applies to all, which is the time availability of each participant. There must not exist overlapping sessions for any of the participants. The soft constraints are:

- Quota: The maximum number of sessions that are allocated for each of the department members.
- Academic Level: The academic level of the department member that is regulated by the government.
- Experience: The previous experience in moderating the sessions.
- ResearchArea: The conformity of the examiner's research areas with the thesis.

The objective function is obtained by summing up the violations of all soft constraints. In practical cases, the separation in hard and soft constraints can be modified by the user, who could relax some of the hard constraints by turning them into soft ones and assigning them a weight. It is also possible to add weight for each of the soft constraints chosen by the users. For the sake of simplicity, this work sticks to the classification provided above.

2.2. Particle swarm optimization

In general, the particle swarm optimization (PSO) algorithm consists of three steps: first, to initialize each particle's position and velocity, second is to update the velocity, and third is to update the position. These three steps are repeated until the stop condition is met, or the maximum iteration is reached. The initial position and velocity of each particle are generated randomly using (1) and (2) where x represents position and v represents velocity [22-25].

$$x_0^i = x_{min} + rand(x_{max} - x_{min}) \quad (1)$$

$$v_0^i = v_{min} + rand(v_{max} - v_{min}) \quad (2)$$

The velocity is updated by using (3). The inertia factor (w), cognitive learning rate (c_1 and c_2), and random numbers (r_1 and r_2) are the parameters that influence the update of the velocity [22].

$$v_{k+1}^{i} = w * v_k^i + c_1 + r_1 * \left(p^i - x_k^i\right) + c_2 * r_2 * (p_k^g - x_k^i) \quad (3)$$

The final step in each iteration is to update each particle's position using (4) [22].

$$x_{k+1}^{i} = x_k^i + v_{k+1}^i \quad (4)$$

2.3. Design and implementation

Figure 1 shows the application workflow. First, the user must input the data that is used to the application. After the data entered, the application will start scheduling using the PSO algorithm, beginning with the schedule for the thesis defense, then the examiner, and the moderator of the thesis defense. After the scheduling process is done, the optimized schedule will be shown by the application. There are three fitness functions developed and used in this research. The first fitness function (fSupervisor) defined by (5) is to set the initial schedule consisting of the student and the supervisor. The second fitness function (fExaminer) defined by (6) is for scheduling the examiner. The third fitness function (fModerator) defined by (7) is for scheduling the moderator. The goal is to find the global minimum for each of these fitness functions.

$$fSupervisor = s_1 + s_2 \quad (5)$$

- $s_1 \in \{0,1\} : 1\ if\ the\ supervisor\ is\ not\ available\ for\ the\ session, 0\ otherwise$
- $s_2 \in \{0,1\} : 1\ if\ the\ session\ overlaps, 0\ otherwise$

$$fExaminer = e_1 + e_2 + e_3 + e_4 + e_5 + e_6 \quad (6)$$

- $e_1 \in \{0,1\} : 1\ if\ the\ examiner\ is\ not\ available\ for\ the\ session, 0\ otherwise$
- $e_2 \in \{0,1\} : 1\ if\ the\ examiner's\ academic\ level\ is\ not\ suitable, 0\ otherwise$
- $e_3 \in \{0,1\} : 1\ if\ the\ session\ already\ has\ an\ examiner\ assigned\ to\ it, 0\ otherwise$
- $e_4 \in \{0,1\} : 1\ if\ the\ assigned\ examiner\ is\ also\ the\ supervisor, 0\ otherwise$
- $e_5 \in \{0,1\} : 1\ if\ the\ examiner's\ quota\ is\ up, 0\ otherwise$
- $e_6 \in \{0,1\} : 1\ if\ the\ examiner's\ research\ area\ is\ not\ suitable\ for\ the\ session, 0\ otherwise$

$$fModerator = m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7 \quad (7)$$

- $m_1 \in \{0,1\} : 1\ if\ the\ moderator\ is\ not\ available\ for\ the\ session, 0\ otherwise$
- $m_2 \in \{0,1\} : 1\ if\ the\ moderator's\ academic\ level\ is\ not\ suitable, 0\ otherwise$
- $m_3 \in \{0,1\} : 1\ if\ the\ moderator\ has\ no\ previous\ experience\ as\ moderator, 0\ otherwise$
- $m_4 \in \{0,1\} : 1\ if\ the\ session\ already\ has\ a\ moderator\ assigned\ to\ it, 0\ otherwise$
- $m_5 \in \{0,1\} : 1\ if\ the\ moderator\ is\ also\ the\ supervisor, 0\ otherwise$
- $m_6 \in \{0,1\} : 1\ if\ the\ moderator\ is\ also\ the\ examiner, 0\ otherwise$
- $m_7 \in \{0,1\} : 1\ if\ the\ moderator's\ quota\ is\ up, 0\ otherwise$

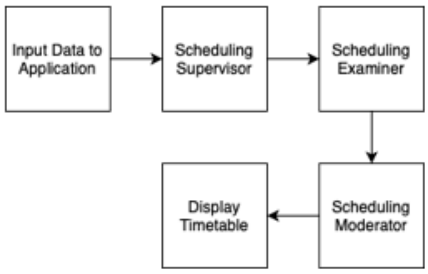

Figure 1. Brief process of application workflow

Figure 2 shows the implementation flowchart of the PSO algorithm for the application. There are three phases of the PSO module, as described in the implementation flowchart. The first PSO module is to initialize the session consisting of student and supervisor. The second is to assign an examiner to the session, and the last is to assign a moderator to the session. In this implementation, v_{max} is not limited to allow the particle to fly through an excellent solution. In addition to that, the position is normalized to be in the range between 0 and 39. The normalization is to follow with the nature of the problem, where there are 40 time slots within two

weeks with five working days and eight hours a day. The maximum number of iteration for each fitness function is set to 20 in this work, and the user can alter this. The implementation is targeted to run until it finds the solution; it will restart the whole process when no solutions to be found. Violations on the hard constraint is not allowed.

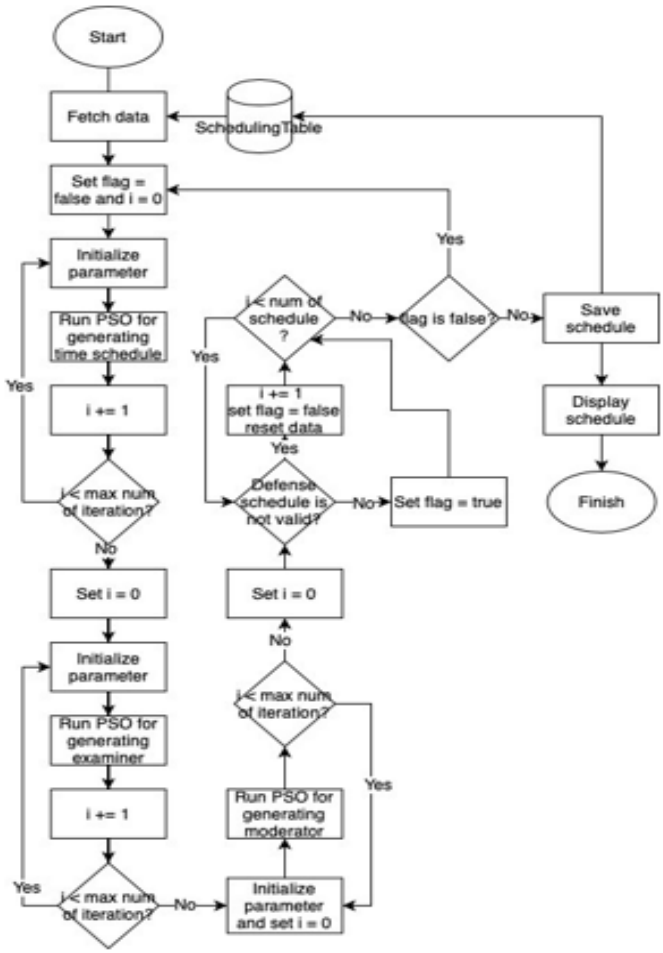

Figure 2. Implementation flowchart

2.4. End-user computing satisfaction (EUCS)

The dimension of EUCS are content, accuracy, format, ease of use, and timeliness. This model is mainly based on the end-user's computing satisfaction model of Doll and Torkzadeh. The model is shown in Figure 3 [26].

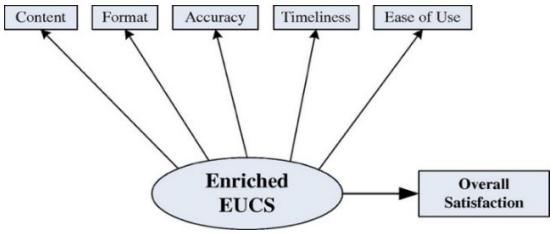

Figure 3. The enriched end-user computing satisfaction model

3. RESULTS AND ANALYSIS

The application is tested using four test-cases specifics for the application. The variables for the test are given in Table 1. Each of the test-cases is executed ten times to measure the overall performance of the

implementation. The first batch of thesis defense consists of two weeks that are in the teaching weeks. The second batch is not during the teaching weeks.

The results of the first test-case are shown in Table 2. The results show that the average number of iterations required to generate the schedule in this test is 160.6 ≅161 iterations. The average execution time for test-case number one is 27.27 seconds with the fastest run on the eight iterations with 7.21 seconds, and the latest is on the second iteration with 69.17 seconds. The average number of violations on the hard constraints is 0, and on the soft constraints is 4.

Table 1. Test-cases and parameter configuration

Test-case No.	Number of students	Number of department members	Maximum quota for lecturer with a position	Maximum quota for lecturer without a position	Thesis defense batch	w	c_1	c_2
1	25	18	5	10	first	0.6	1.5	1.5
2			5	10	second			
3			4	9	first			
4			4	9	second			

Table 2. Results of test-case number one

Repetition no.	Number of iterations	Execution time (sec)	Violated soft constraints
1	256	44.28	4
2	415	69.17	5
3	133	22.6	4
4	108	18.4	4
5	189	32.19	4
6	74	12.72	4
7	98	16.57	3
8	41	7.21	5
9	82	14.17	3
10	210	35.36	4

The results for test-case number two is shown in Table 3. The average number of iterations required to generate the schedule is 18.2 ≅ 19 iterations. The average execution time is 3.19 secs with the fastest is 0.18 seconds on the first iteration, and the latest is 6.56 seconds on the seventh iteration. The average number of violations on the soft constraints is 3.2 ≅ 4. The results for test-case number three is shown in Table 4. The average number of iterations required to generate the schedule is 2,823.6 ≅ 2,824 iterations. The average execution time is 450.19 secs with the fastest is on the ninth iteration with 41.04 secs, and the latest is on the forth iteration with 1,312.97 seconds. The average number of violations on the soft constraints is 3.9 ≅ 4.

Table 3. Results of test-case number two

Repetition no.	Number of iterations	Execution time (sec)	Violated soft constraints
1	1	0.18	3
2	25	4.14	4
3	34	5.57	2
4	23	3.86	4
5	8	1.45	3
6	6	1.15	4
7	37	6.56	3
8	32	5.77	3
9	7	1.44	3
10	9	1.79	3

The results for test-case number four is shown in Table 5. The average number of iterations required to generate the schedule is 76 iterations. The average execution time is 18.69 seconds with the fastest is 4.47 seconds on the sixth iteration, and the latest is 36.94 seconds on the fifth iteration. The average number of violations on the soft constraints is 3.6 ≅ 4. Based on the comparison of test-case number one and test-case number two shown in Figure 4, the application's overall performance is best in the second batch. In this period, the optimization process runs faster due to no overlapping sessions between thesis defense sessions and department members' teaching schedules. This same characteristic is also embodied in another comparison shown in Figure 5, where the application performs less in test-case number 3 compare to test-case number 4.

The evaluation of the application is measured using EUCS in an interview with the end-user of this application. The EUCS questionnaire aims to measure the content, accuracy, format, ease of use, and application timeliness. Figure 6 displays the EUCS result with the overall score is 4.3 out of 6. The highest score is 5 out of 6 for the ease of use factor. Content scores the lowest with 3.8 out of 6. The content could be improved in future work to add more explanation and information regarding the application and the whole process.

Table 4. Results of test-case number three

Repetition no.	Number of iterations	Execution time (sec)	Violated soft constraints
1	349	54.97	3
2	1,128	176.46	6
3	5,736	915.2	4
4	8,144	1,312.97	4
5	2,544	403.94	4
6	4,600	733.36	4
7	948	150.19	3
8	1,071	169.42	4
9	259	41.04	3
10	3,454	543.87	4

Table 5. Results of test-case number four

Repetition no.	Number of iterations	Execution time (sec)	Violated soft constraints
1	110	21.91	3
2	62	14.18	4
3	38	10.63	3
4	159	36.09	3
5	160	36.94	4
6	11	4.47	4
7	72	16.3	4
8	57	19.75	4
9	57	14.6	3
10	34	7.53	4

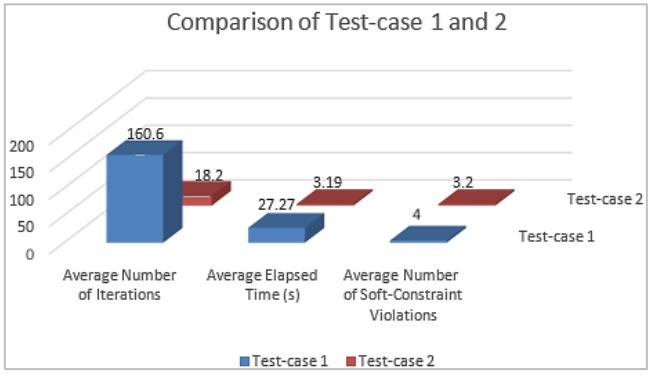

Figure 4. Comparison of test-case number one and number two

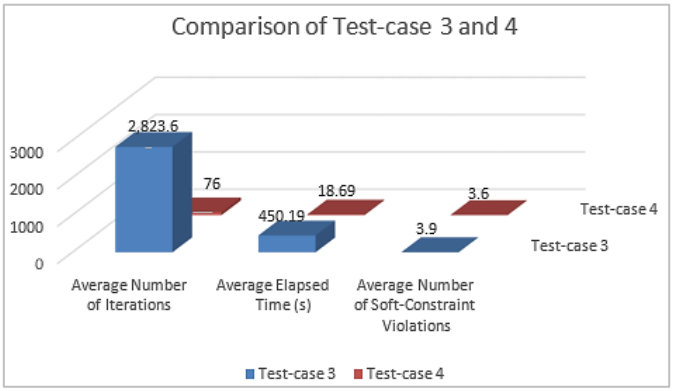

Figure 5. Comparison of test-case number three and number four

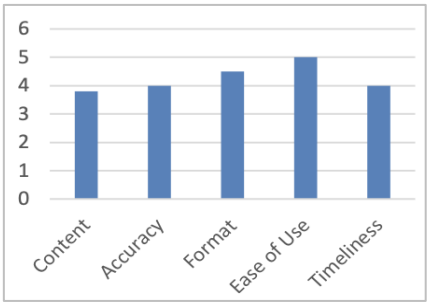

Figure 6. End-user computing satisfaction evaluation result

4. CONCLUSION

This work has successfully demonstrated the use of PSO for the thesis defense timetabling problem. The implementation successfully schedules 25 thesis defense exams without violating the hard constraint under different test-cases. The optimal inertia factor value is 0.6 for this application. The application is currently in use in the Informatics Department in UMN. The personnel involved have reduced the time spent in the overall procedure by several staff-hours for each graduation period (which consists of 4-6 thesis defense batch a year). In addition to that, the system has improved the solution's fairness in terms of the department members' multiple duties. The overall score of the EUCS evaluation for the application is 4.3 out of 6.

In the future, different fitness functions and linear decreasing inertia could be studied to yield better performance. Additional work on the user interface consists of adding a new menu that provides guidance and information on how-to-use the application. Features for sorting the application's data are also needed to increase the application's ease-of-use aspect further. In addition to the reporting menu, it is easier to archive schedules that have been made and modify the schedule in the thesis trial scheduling application. Another feature to be added is to allow constraints modification by end-user. This feature will allow end-user to add and to remove additional constraints.

REFERENCES

[1] M. Battistutta, S. Ceschia, F. De Cesco, L. Di Gaspero, and A. Schaerf, "Modelling and solving the thesis defense timetabling problem∗," *J. Oper. Res. Soc.*, vol. 70, no. 7, 2019.
[2] B. Chopard and M. Tomassini, "Particle swarm optimization," in: *An Introduction to Metaheuristics for Optimization. Natural Computing Series. Springer, Cham*, 2018.
[3] X. Cai and Z. Cui, "Hungry particle swarm optimization," *ICIC Express Lett.*, vol. 4, no. 3, 2010.
[4] S. M. Elsayed, R. A. Sarker, and E. Mezura-Montes, "Self-adaptive mix of particle swarm methodologies for constrained optimization," *Information Sciences,* 2014.
[5] K. E. Parsopoulos and M. N. Vrahatis, "Particle Swarm Optimization Method for Constrained Optimization Problems," *Frontiers in Artificial Intelligence and Applications,* vol. 76, pp. 214-220, 2002.
[6] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Comput.*, vol. 22, pp. 387-408, 2018.
[7] F. Nzanywayingoma and Y. Yang, "Analysis of Particle Swarm Optimization and Genetic Algorithm based on Task Scheduling in Cloud Computing Environment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 1, pp. 19-25, 2017.
[8] X. Ma, Y. Yan and Q. Liu, "A multi-objective particle swarm optimization for multiple knapsack problem with strong constraints," *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Wuhan, 2018, pp. 1201-1205, doi: 10.1109/ICIEA.2018.8397892.
[9] F. Hembecker, H. S. Lopes, and W. Godoy, "Particle swarm optimization for the multidimensional knapsack problem," *International Conference on Adaptive and Natural Computing Algorithms*, 2007.
[10] L. Gao, C. Peng, C. Zhou, and P. Li, "Solving flexible job-shop scheduling problem using general particle swarm optimization," *The 36th CIE Conference on Computers & Industrial Engineering,* 2006.
[11] P. S. Srinivas, R. V. Ramachandra, and C. S. Rao, "Particle Swarm Optimization Approach for Scheduling of Flexible Job Shops," *International Journal of Engineering Research & Technology (IJERT),* vol. 1, no. 5, pp. 1-6, 2012.
[12] K. Saber Chenari, H. Abghari, and H. Tabari, "Application of PSO algorithm in short-term optimization of reservoir operation," *Environ. Monit. Assess.*, vol. 188, 2016.
[13] T. Chen, B. Zhang, X. Hao, and Y. Dai, "Task scheduling in grid based on particle swarm optimization," *2006 Fifth International Symposium on Parallel and Distributed Computing,* 2006.
[14] L. Zhang, Y. Chen, R. Sun, and B. Yang, "A Task Scheduling Algorithm Based on PSO for Grid Computing," *Int. J. Comput. Intell. Res.*, vol. 4, no. 1, pp. 37-43, 2008.
[15] H. Zhang, H. Li, and C. M. Tam, "Particle swarm optimization for resource-constrained project scheduling," *Int. J. Proj. Manag.*, vol. 25, pp. 948-954, 2006.

[16] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010.
[17] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," *Proc. - 2011 6th Annu. ChinaGrid Conf. ChinaGrid 2011*, 2011.
[18] S. Y., "Employee Scheduling based on Particle Swarm Optimization Algorithm and its Variation," *Int. J. Comput. Appl.*, 2016, doi: 10.5120/ijca2016910809.
[19] H. M. Sani and M. M. Yabo, "Solving Timetabling problems using Genetic Algorithm Technique," *Int. J. Comput. Appl.*, 2016.
[20] S. Wijaya and A. Wicaksana, "JACOB voice chatbot application using wit. Ai for providing information in UMN," *Int. J. Eng. Adv. Technol.*, vol. 8, no. 6, 2019.
[21] K. Filbert and S. Hansun, "Ticketing & CS system development for industrial needs," *Int. J. Sci. Technol. Res.*, 2019.
[22] M. A. El-Shorbagy and A. E. Hassanien, "Particle Swarm Optimization from Theory to Applications," *Int. J. Rough Sets Data Anal.*, vol. 5, no. 2, 2018.
[23] F. Marini and B. Walczak, "Particle swarm optimization (PSO). A tutorial," *Chemom. Intell. Lab. Syst.*, vol. 149, pp. 153-165, 2015.
[24] M. Clerc, "Particle Swarm Optimization," 2010. [Online]. Available: https://kamenpenkov.files.wordpress.com/2016/01/pso-m-clerc-2006.pdf
[25] M. E. H. Pedersen and A. J. Chipperfield, "Simplifying Particle Swarm Optimization," *Appl. Soft Comput. J.*, vol. 10, no. 2, pp. 618-628, 2010,
[26] V. P. Aggelidis and P. D. Chatzoglou, "Hospital information systems: Measuring end user computing satisfaction (EUCS)," *J. Biomed. Inform.*, vol. 45, no. 3, pp. 566-579, 2012.

BIOGRAPHIES OF AUTHORS

Gilbert Christopher is graduated from the Department of Informatics at UMN. He is currently working as Software Development Engineer at Garasi.id since 2019. His research interest is artificial intelligence, specifically optimization and scheduling.

Arya Wicaksana is a lecturer at the Department of Informatics at UMN. He received Master Degree in VLSI Engineering from Universitas Tunku Abdul Rahman. He successfully demonstrated the UTAR first-time success ASIC design methodology on a multi-processor system-on-chip project using 0.18um processing technology in 2015. His main research interests are quantum computing, hardware/software co-development, and computational intelligence. He has recently been working on a human-like voice chatbot system called Jacob and post-quantum cryptography for blockchain applications. He is affiliated with ACM and IEEE as a professional member. In IJNMT and IFERP, he has served as an invited reviewer and an invited author in IntechOpen and other scientific publications.