International Journal of Innovative Computing, Information and Control
Volume **21**, Number **3**, June **2025**

ICIC International ©2025 ISSN 1349-4198
pp. **781–790**

OPTIMIZING GENETIC PROCESS MINING WITH HEURISTIC AND HONEY BEE APPROACHES FOR ENHANCED INITIAL POPULATION DESIGN AND NEIGHBORHOOD SEARCH

Yutika Amelia Effendi[1] and Minsoo Kim[2,*]

[1]Robotics and Artificial Intelligence Engineering
Faculty of Advanced Technology and Multidiscipline
Airlangga University
Kampus C UNAIR, Jl. Dr. Ir. H. Soekarno, Mulyorejo, Surabaya, Jawa Timur 60115, Indonesia
yutika.effendi@ftmm.unair.ac.id

[2]Department of Industrial and Data Engineering
College of Engineering
Pukyong National University
45 Yongso-ro, Namgu, Busan 48513, Korea
*Corresponding author: minsky@pknu.ac.kr

Received September 2024; revised January 2025

Abstract. *Genetic process mining is a method used to identify, monitor, and improve processes by extracting insights from event logs within modern information systems. While traditional Genetic Algorithms (GAs) can mine models with all structural constructs, except for duplicate tasks, and are resilient to noise, their significant computational time remains a major drawback. This study enhances the Honey Bee Genetic Algorithm (HBGA) by incorporating the Heuristic Miner (HM)'s dependency measure, addressing challenges related to random population generation and excessive parameter tuning while accelerating the initial stages of evolution. By leveraging HM for initial population design and HBGA for neighborhood search, the proposed approach reduces the number of fitness function evaluations and computational time without compromising the accuracy of the final process model. Experimental results demonstrate that the proposed method outperforms traditional genetic process mining in terms of execution time and fitness function evaluations, while achieving superior fitness. Consequently, this process mining algorithm offers a more efficient and effective solution for discovering workflow models.*

Keywords: Genetic process mining, Dependency relation, Heuristic miner, Honey bee algorithm, Process mining, Process discovery, Process enhancement

1. **Introduction.** Process mining holds significant potential for enhancing process mapping, which can lead to improved process performance [1]. Among the widely recognized algorithms in process mining is genetic process mining, a robust method for identifying, monitoring, and optimizing real-world processes by analyzing event logs commonly found in modern information systems [2]. Its primary objective is to derive models that accurately represent actual process behavior, thereby facilitating better decision-making and process optimization [3]. Traditional Genetic Algorithms (GAs) have been widely applied in process mining due to their strong search capabilities and flexibility in handling complex optimization problems [4]. However, the effectiveness of GAs relies heavily on the quality of the initial population and the efficiency of the neighborhood search mechanism [5].

DOI: 10.24507/ijicic.21.03.781

A challenge in genetic process mining is generating an initial population that is both diverse and representative of high-quality solutions. An effective initial population can significantly enhance the convergence speed and solution quality of the genetic algorithm [6]. Similarly, refining the neighborhood search mechanism can lead to more efficient exploration and exploitation of the search space, preventing the algorithm from getting trapped in local optima and aiding in the discovery of globally optimal solutions [7]. Recent advances in bio-inspired algorithms, such as the Honey Bee Algorithm (HBA) [8], provide promising solutions to these challenges. The HBA, inspired by the foraging behavior of honey bees, excels at balancing exploration and exploitation [9], making it well-suited to enhance both the initial population and neighborhood search in genetic process mining. Additionally, in specific domains, GA performance can be boosted by incorporating established heuristics to construct the initial population [10,11]. Research shows that while heuristics do not alter the final outcome if the GA runs indefinitely, they can accelerate early evolution stages.

Efforts to reduce genetic processing time have been extensively studied. Parallel and Distributed Genetic Algorithms (PGAs) distribute computation across multiple processors or machines, significantly reducing runtime by parallelizing fitness evaluations [12]. Memetic algorithms, which integrate local search techniques with GAs, focus on refining high-quality solutions to minimize the number of generations required [13]. Adaptive Genetic Algorithms (AGAs) dynamically adjust parameters, such as mutation rates and crossover probabilities, optimizing the search process and preventing stagnation [14]. Hybrid approaches combine GAs with other optimization methods, such as Particle Swarm Optimization (PSO) or Ant Colony Optimization (ACO), leveraging their strengths to improve convergence speed and computational efficiency [15]. Additionally, surrogate modeling and fitness caching techniques reduce the computational burden of fitness evaluations by approximating or reusing results [16]. Problem-specific representations and operators tailored to the domain also enhance efficiency by avoiding unnecessary computations, while clustering techniques reduce population size without compromising diversity.

This study seeks to reduce GA execution time, which is a critical improvement, by enhancing the Honey Bee Genetic Algorithm (HBGA) [17] through the integration of the Heuristic Miner (HM)'s dependency measure approach [2,18] for initial population construction. The HM algorithm accelerates early evolutionary stages without altering the discovered process models, in contrast to the typical random initial population generation in GAs. Our approach involves designing techniques that capture event log variations and represent them in the initial population through dependency measurements and pattern identification, such as frequent activity sequences, parallel branches, and loops. Subsequently, we implement a neighborhood search mechanism based on the HBA to optimize search space exploration and exploitation, enhancing the HBGA's ability to find high-fitness solutions. Using HM to construct the initial population enables the HBGA to overcome traditional limitations like random population generation and excessive parameter tuning. This improvement accelerates the evolutionary process's early stages, reduces the number of fitness function executions, and minimizes computational time, ultimately leading to the discovery of accurate process models.

For evaluation, this research will comprehensively compare the proposed approach with GA and HBGA in a yarn manufacturing process that includes sequential and parallel processes. Evaluation metrics will include fitness value, the number of fitness calculations, and computational runtime. Overall, this approach offers a more efficient and effective solution for mining workflow models, marking a valuable advancement in the field.

This paper is structured as follows. Chapter 2 discusses related works. Chapter 3 presents our proposed approach. Chapter 4 provides the experimental results and discussion. Finally, Chapter 5 offers conclusions and future work.

2. **Related Works.** Many researchers have significantly advanced process mining [2,19], with most methods beginning with an event log to uncover underlying processes. This section provides a brief overview of related works relevant to our study.

2.1. **Genetic process mining.** GAs, inspired by Darwin's "survival of the fittest" principle [17], aim to find optimal solutions by evolving successive generations from an initially random population [2]. These generations are refined through genetic operators like crossover and mutation, with a fitness function evaluating each generation's quality to determine selection probabilities for the next generation. The algorithm stops once a termination condition is met [3,10], as shown in Figure 1.

0. Initialize population P with process models
1. For each process model p in P:
   Evaluate fitness $F(p)$ based on event logs
2. Repeat until stopping criteria (time limit, sufficient fitness, etc.) are met:
   a. Select a subset S of P based on fitness (best-fit process models for reproduction)
   b. Generate new process models N through crossover and mutation on S
   c. For each new process model n in N:
      Evaluate fitness $F(n)$
   d. Replace the least-fit process models in P with the new process models N
3. Terminate the algorithm

FIGURE 1. Pseudocode of the GA

A key challenge in workflow model mining is identifying dependencies between activities. Existing methods struggle with issues such as duplicate and hidden activities, non-free-choice constructs, noise, and incompleteness [20-22]. While GAs can handle most structural constructs and are resistant to noise, they face a significant drawback: high computational time [3,4,23].

2.2. **Honey bee genetic algorithm.** The bee colony algorithm is a widely used method for finding optimal solutions, popular for its unique approach that mimics honey bee swarm intelligence [8,9]. [24-26] have shown its effectiveness in various optimization fields. This paper applies the HBA to process mining by combining it with the GA in a new method called the HBGA [17]. The HBGA leverages the HBA's neighborhood search for identifying the best individuals, paired with GA's diversity to achieve global optimization, thus reducing the number of fitness function evaluations compared to random parent selection [17]. Figure 2 outlines HBGA's steps, where n is the number of scout bees, m the sites chosen for search, and e the elite sites. Despite its strengths, HBGA still requires extensive parameter tuning and improved execution time for large event logs [17].

2.3. **Heuristic miner algorithm.** The HM algorithm is widely used in process mining to derive process models from event logs [2,18]. It calculates a dependency measure that quantifies the strength of relationships between activities based on their co-occurrence frequency in the log [10,27-29]. The primary steps are as follows.

1) Co-occurrence Frequency: For each activity pair (A, B), count how often they appear together in a trace.

0. Initialize parameters: max_generations, n, m, e, crossover_rate, mutation_rate
1. Generate initial population P with n random solutions
2. Evaluate fitness $F(p)$ for each p in P
3. Select m sites for neighborhood exploration from P
4. While stopping criteria are not met:
    5. Identify top e elite bees E and $(m - e)$ remaining bees R from the m sites
    6. For each site in E:
        7. Apply crossover and mutation to generating new solutions
    8. For each site in R:
        9. Apply crossover and mutation to generating new solutions
    10. Evaluate fitness F(new_solution) for each new solution
    11. Select the best solution from each of the m sites to form new population P
12. Return the best solution from P

FIGURE 2. Pseudocode of the HBGA

2) Individual Frequency: Count total occurrences of A and B in the log, regardless of co-occurrence.
3) Dependency Measure Calculation: Compute the dependency measure as the ratio of the co-occurrence frequency to the minimum individual frequency of A or B.

$$\textit{Dependency Measure } (A, B) = \frac{\textit{Frequency of Co-occurrence } (A, B)}{\min(\textit{Frequency of } A, \textit{Frequency of } B)} \tag{1}$$

4) Normalization (*Optional*): Adjust dependency measures to a set range, like 0 to 1.
5) Thresholding (*Optional*): Apply a threshold to filtering weaker dependencies.
6) Visualization: Use the dependency measures to visualize the process model as a graph, with activities as nodes and dependencies as edges.

3. **Proposed Approach.** The proposed approach integrates the HM algorithm with the HBGA to enhance process discovery from event logs, leveraging the strengths of both methods. The workflow, as shown in Figure 3, begins with initializing parameters, such as the maximum number of generations, population size, and rates for crossover and mutation, which are essential for controlling the behavior and convergence of the algorithm.

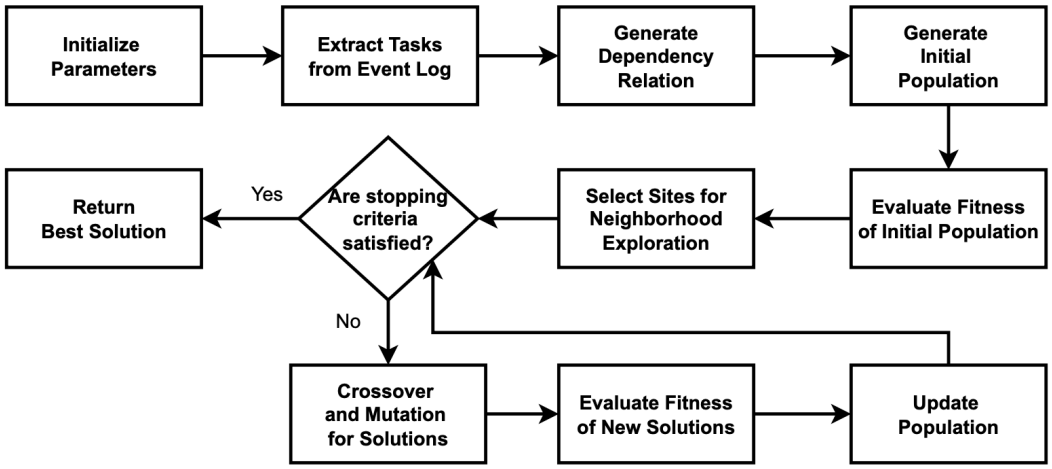

FIGURE 3. Flowchart of the proposed approach

The process starts by extracting tasks from the event log and generating a dependency relation using the HM algorithm. This relation quantifies the strength of relationships between tasks [10,27], distinguishing between loops and parallel structures. The HM-generated dependency measures provide a solid foundation for constructing the initial population by defining causality relations for each individual. This dependency relation also guides the evaluation of fitness during the algorithm, ensuring solutions align with key properties of process models, such as completeness (covering all event traces) and preciseness (minimizing extra behaviors not found in the log) [3,10].

The initial population of solutions is generated, and fitness values are computed for each solution based on how well they adhere to the dependency relation. Using these evaluations, sites for neighborhood exploration are selected, prioritizing promising areas. At this stage, the algorithm evaluates whether the stopping criteria are met. These criteria include reaching the maximum number of generations or observing no improvement in the top solution for a specified number of iterations [10,17].

If the stopping criteria are not met, the algorithm enters an iterative loop, where crossover and mutation operations are applied to elite and non-elite sites using a neighborhood search strategy. This iterative refinement balances exploration (searching new areas of the solution space) and exploitation (refining high-quality solutions). The fitness of the newly generated solutions is evaluated, and the population is updated by retaining the best-performing solutions. This process repeats until the stopping criteria are satisfied.

The integration of HM into HBGA enhances the traditional algorithm by replacing random initialization with a dependency-based initial population. This approach ensures faster convergence and better-quality solutions, as the HM provides meaningful starting points. Genetic operators like elitism, crossover, and mutation refine the solutions, and the neighborhood search strategy further optimizes them. The final solution, returned upon meeting the stopping criteria, represents the discovered process model that is both accurate and efficient.

The pseudocode is illustrated in Figure 4. In this representation, L denotes the event log, p represents the power value, D is the dependency function, and C is the dependency relation. T refers to the set of tasks or activities extracted from the event log L. Additionally, r represents a randomly generated number between 0 and 1, used to probabilistically determine whether a dependency relation between two tasks should be added to C, based on the dependency strength. Furthermore, n indicates the number of scout bees, m represents the number of sites selected for neighborhood search, and e denotes the number of elite sites among the chosen ones. This integrated approach highlights how the combined strengths of the HM and HBGA contribute to robust and efficient process discovery.

4. **Results and Discussion.** In this section, we evaluate the proposed approach by comparing its performance with the GA and the HBGA algorithms. We focus on the original GA and HBGA, as there are no other process discovery algorithms in process mining that operate similarly to GA, and the GA used here differs from the typical GA in artificial intelligence [30]. For the experiments, we used an event log from a yarn manufacturing process at a factory in Jakarta, Indonesia. Collected in June 2023, the log includes 100 cases, 14 activities, and 4 originators. Preprocessing revealed both sequential and parallel processes. The experiments were conducted on a computer with an Intel® Core™ i7 CPU at 3.1 GHz, 8 GB memory, a 320 GB hard disk (5400 RPM), and Windows 10 Ultimate 64-bit OS. Table 1 summarizes the parameters used for testing the dataset with GA, HBGA, and our proposed approach.

0. Initialize parameters: max_generations, n, m, e, crossover_rate, mutation_rate
1. $T \leftarrow$ Extract tasks from L
2. $C \leftarrow \emptyset$
3. For each task pair $(t1, t2)$ in $T \times T$:
    a. $r \leftarrow$ Random(0, 1)
    b. If $r < (D(t1, t2, L))^p$ then:
        i. $C \leftarrow C \cup \{(t1, t2)\}$
4. $P \leftarrow$ Generate initial population with n random solutions
5. For each p in P:
    Evaluate fitness $F(p)$ using dependency relation C
6. $S \leftarrow$ Select m sites for neighborhood exploration
7. While stopping criteria are not met:
    a. $E \leftarrow$ Identify top e elite bees from S
    b. $R \leftarrow$ Identify $(m - e)$ remaining bees from S
    c. For each site in E:
        i. Apply crossover and mutation to generating new solutions
    d. For each site in R:
        i. Apply crossover and mutation to generating new solutions
    e. For each new solution n:
        Evaluate fitness $F(n)$ using dependency relation C
    f. $P \leftarrow$ Update the population with the best new solutions
8. Return the best solution from P as the discovered process model

FIGURE 4. Pseudocode of the proposed approach

TABLE 1. Parameters used in the experiments

Algorithm	**Maximum generations limit**	**Size of population (N)**	m	e
The GA	200	100	–	–
The HBGA	10	50	50	30
Proposed approach	10	50	50	30

To assess the performance of the algorithms tested, we focused on two main factors: 1) the number of executions of the fitness functions and 2) the computational time required to achieve the target fitness value. In this paper, computational time was not used as a primary measure because it can vary across different operating systems and code implementations. Consequently, our primary comparison factor is the number of executions of the fitness function. Each algorithm was executed 10 times with different parameter values to evaluate the results.

Table 2 displays the number of fitness calculations and computational execution times for each algorithm, based on the highest fitness value obtained during the runs. The highest fitness value achieved by each algorithm reflects its ability to accurately represent the event log, with the proposed approach attaining the highest fitness at 0.981309, followed closely by the HBGA at 0.980450, and the GA at 0.972893. In terms of efficiency, the proposed approach required the fewest fitness calculations (6230) and the shortest computational time (4411 ms), while the HBGA performed 10458 fitness calculations in 8293 ms. The GA, with the lowest performance, required 24917 fitness calculations and 30530 ms of computational time. Overall, the proposed approach outperforms both the GA and the HBGA in accuracy and computational efficiency.

TABLE 2. Final results

Algorithm	Highest fitness value	Number of fitness calculations	Computational time (ms)
The GA	0.972893	24917	30530
The HBGA	0.980450	10458	8293
Proposed approach	0.981309	6230	4411

Furthermore, to measure the improvement of the proposed approach, we calculated the percentage enhancement in terms of fitness function executions and execution time using the formulas in Equations (2) and (3). The findings are summarized in Table 3.

$$Pfitness = \frac{Number\ of\ fitness\ calculations\ of\ the\ proposed\ approach}{Number\ of\ fitness\ calculations\ of\ the\ GA/HBGA} \times 100\% \quad (2)$$

$$Ptime = \frac{Execution\ time\ of\ the\ proposed\ approach}{Execution\ time\ of\ the\ GA/HBGA} \times 100\% \quad (3)$$

TABLE 3. Performance improvement results for the proposed approach

Algorithm	Number of fitness calculations	Computational time
GA	25%	15%
HBGA	59%	53%

Table 3 summarizes the performance improvements of the proposed approach compared to the GA and the HBGA algorithms in terms of the number of fitness calculations and computational time. When compared to the GA, the proposed approach required only 25% of the number of fitness calculations and 15% of the computational time, effectively reducing these metrics by 75% and 85%, respectively. Similarly, compared to the HBGA, it used 59% of the fitness calculations and 53% of the computational time, representing reductions of 41% and 47%. These findings demonstrate that the proposed approach significantly enhances computational efficiency over both the GA and the HBGA, particularly in reducing the number of fitness evaluations and overall execution time while still presenting the correct process model. Finally, Figure 5 shows the visual representation of the process model created using YAWL (Yet Another Workflow Language) after running all the algorithms. The process models obtained from all three algorithms are identical,

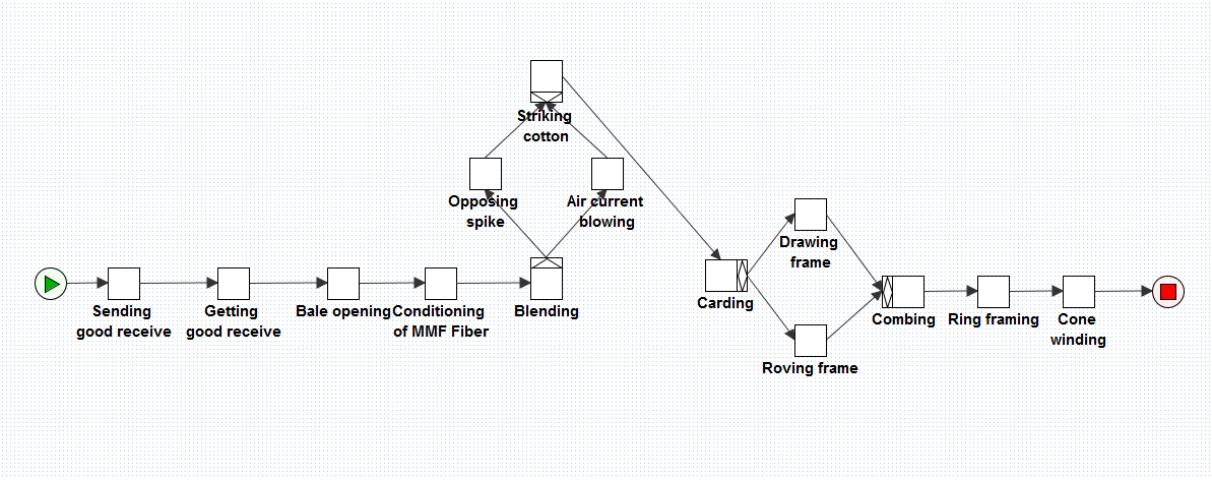

FIGURE 5. Discovered process model for all three algorithms

as this research focuses solely on reducing the number of fitness evaluations and overall execution time.

5. **Conclusions.** This paper introduced an enhanced approach for process mining that improves the basic GA by integrating the HM and the HBGA algorithms. This integration optimizes the initial population design and neighborhood search, enabling the discovery of process models with higher fitness values, fewer fitness calculations, and reduced computational time. At the start of the proposed approach, the HM algorithm's dependency measure is used to design the initial population, capturing direct and indirect dependencies and identifying patterns like frequent sequences and parallel branches. The HBGA's neighborhood search concept is also applied to improving individual selection, moving away from the traditional random parent selection.

Experimental results show that the proposed approach outperforms both the GA and the HBGA in terms of execution time and the number of fitness evaluations. Therefore, we conclude that this approach is effective for discovering process models with faster computational time. Future work will focus on enhancing the genetic process mining algorithm to handle both complete and incomplete trace cases.

Acknowledgment. This work was supported by the National Research Foundation of Korea (NRF) grant, funded by the government of the Republic of Korea (MSIT) (RS-2023-00242528).

REFERENCES

[1] Y. A. Effendi and M. Kim, Refining process mining in port container terminals through clarification of activity boundaries with double-point timestamps, *ICIC Express Letters, Part B: Applications*, vol.15, no.1, pp.61-70, DOI: 10.24507/icicelb.15.01.61, 2024.

[2] W. M. P. van der Aalst, *Process Mining: Data Science in Action*, Springer, DOI: 10.1007/978-3-662-49851-4, 2016.

[3] W. M. P. van der Aalst, A. K. Alves De Medeiros and A. J. M. M. Weijters, Genetic process mining, in *Applications and Theory of Petri Nets 2005. ICATPN 2005. Lecture Notes in Computer Science*, G. Ciardo and P. Darondeau (eds.), Berlin, Heidelberg, Springer, 2005.

[4] A. K. Alves De Medeiros, *Genetic Process Mining*, Ph.D. Thesis, Industrial Engineering and Innovation Sciences, Technische Universiteit Eindhoven, DOI: 10.6100/IR614016, 2006.

[5] V. S. Jorapur, V. S. Puranik, A. S. Deshpande and M. Sharma, A promising initial population based genetic algorithm for job shop scheduling problem, *Journal of Software Engineering and Applications*, vol.9, pp.208-214, DOI: 10.4236/jsea.2016.95017, 2016.

[6] I. Vlašić, M. Durasević and D. Jakobović, Improving genetic algorithm performance by population initialisation with dispatching rules, *Computers & Industrial Engineering*, vol.137, DOI: 10.1016/j.cie.2019.106030, 2019.

[7] J. Yang, H. Liu, K. Liang, M. Shan, L. Kong and L. Zhou, A genetic algorithm with lower neighborhood search for the three-dimensional multiorder open-size rectangular packing problem, *International Journal of Intelligent Systems*, vol.2024, 4456261, DOI: 10.1155/2024/4456261, 2024.

[8] B. Yuce, M. S. Packianather, E. Mastrocinque, D. T. Pham and A. Lambiase, Honey bees inspired optimization method: The bees algorithm, *Insects*, vol.4, no.4, pp.646-662, DOI: 10.3390/insects 4040646, 2013.

[9] K. Diwold, M. Beekman and M. Middendorf, Honeybee optimisation – An overview and a new bee inspired optimisation scheme, in *Handbook of Swarm Intelligence. Adaptation, Learning, and Optimization*, B. K. Panigrahi, Y. Shi and M. H. Lim (eds.), Berlin, Heidelberg, Springer, 2011.

[10] A. K. Alves De Medeiros, A. J. M. M. Weijters and W. M. P. van der Aalst, Genetic process mining: An experimental evaluation, *Data Mining and Knowledge Discovery*, vol.14, pp.245-304, DOI: 10.1007/s10618-006-0061-7, 2007.

[11] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Natural Computing, Springer, DOI: 10.1007/978-3-662-05094-1, 2003.

[12] T. Harada and E. Alba, Parallel genetic algorithms: A useful survey, *ACM Computing Survey*, vol.53, no.4, Article no.86, pp.1-39, DOI: 10.1145/3400031, 2020.

[13] V. Kralev and R. Kraleva, Combining genetic algorithm with local search method in solving optimization problems, *Electronics*, vol.13, no.20, DOI: 10.3390/electronics13204126, 2024.
[14] C. H. Dai, Y. F. Zhu and W. R. Chen, Adaptive probabilities of crossover and mutation in genetic algorithms based on cloud model, *IEEE Information Theory Workshop (ITW'06)*, Chengdu, China, pp.710-713, DOI: 10.1109/ITW2.2006.323754, 2006.
[15] K. Menghour and L. Souici-Meslati, Hybrid ACO-PSO based approaches for feature selection, *International Journal of Intelligent Engineering and Systems*, vol.9, no.3, pp.65-79, DOI: 10.22266/ijies2016.0930.07, 2016.
[16] A. Kattan, A. Agapitos, Y.-S. Ong, A. A. Alghamedi and M. O'Neill, GP made faster with semantic surrogate modelling, *Information Sciences*, vols.355-356, pp.169-185, DOI: 10.1016/j.ins.2016.03.030, 2016.
[17] Y. Z. Seleem, M. H. Mohamed and K. F. Hussain, Improving genetic process mining using Honey Bee algorithm, *2013 2nd International Conference on Informatics & Applications (ICIA)*, pp.59-65, DOI: 10.1109/ICoIA.2013.6650230, 2013.
[18] A. J. M. M. Weijters, W. M. P. van der Aalst and A. K. Alves De Medeiros, *Process Mining with the Heuristics-Miner Algorithm*, Technical Report, Technische Universiteit Eindhoven, 166, pp.1-34, 2006.
[19] W. M. P. van der Aalst, Foundations of process discovery, in *Process Mining Handbook. Lecture Notes in Business Information Processing*, W. M. P. van der Aalst and J. Carmona (eds.), Cham, Springer, 2022.
[20] W. M. P. van der Aalst, *Challenges in Business Process Mining*, http://bpmcenter.org/wp-content/uploads/reports/2010/BPM-10-01.pdf, Accessed on January 7, 2025.
[21] N. Martin, D. A. Fischer, G. D. Kerpedzhiev et al., Opportunities and challenges for process mining in organizations: Results of a Delphi study, *Business Information Systems Engineering*, vol.63, pp.511-527, DOI: 10.1007/s12599-021-00720-0, 2021.
[22] M. Faizan, M. F. Zuhairi, S. B. Ismail and R. Ahmed, Challenges and use cases of process discovery in process mining, *International Journal of Advanced Trends in Computer Science and Engineering*, vol.9, no.4, DOI: 10.30534/ijatcse/2020/141942020, 2020.
[23] F. G. Lobo, D. E. Goldberg and M. Pelikan, *Time Complexity of Genetic Algorithms on Exponentially Scaled Problems*, https://www.fernandolobo.info/p/gecco00.pdf, Accessed on January 7, 2025.
[24] D. Ferreira, M. Zacarias, M. Malheiros and P. Ferreira, Approaching process mining with sequence clustering: Experiments and findings, in *Business Process Management. BPM 2007. Lecture Notes in Computer Science*, G. Alonso, P. Dadam and M. Rosemann (eds.), Berlin, Heidelberg, Springer, 2007.
[25] M. A. M. Shukran, Y. Y. Chung, W. C. Yeh, N. Wahid and A. M. A. Zaidi, Artificial bee colony based data mining algorithms for classification tasks, *Modern Applied Science*, vol.5, no.4, pp.219-220, DOI: 10.5539/mas.v5n4p217, 2011.
[26] T. Dušan, Š. Milica and D. Tatjana, Bee colony optimization – Part II: The application survey, *Journal of Operations Research*, vol.25, no.2, pp.185-219, DOI: 10.2298/YJOR131029020T, 2015.
[27] R. Sarno, Y. A. Effendi and F. Haryadita, Modified time-based heuristics miner for parallel business processes, *International Review on Computers and Software*, vol.11, no.3, pp.249-260, DOI: 10.15866/irecos.v11i3.8717, 2016.
[28] P. Weber, B. Bordbar and P. Tiňo, A principled approach to mining from noisy logs using Heuristics Miner, *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Singapore, pp.119-126, DOI: 10.1109/CIDM.2013.6597226, 2013.
[29] A. J. M. M. Weijters and J. T. S. Ribeiro, Flexible Heuristics Miner (FHM), *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Paris, France, pp.310-317, DOI: 10.1109/CIDM.2011.5949453, 2011.
[30] Y. A. Effendi and M. Kim, Timed genetic process mining for robust tracking of processes under incomplete event log conditions, *Electronics*, vol.13, no.18, 3752, DOI: 10.3390/electronics13183752, 2024.

Author Biography

Yutika Amelia Effendi received her B.Comp.Sc. and M.Comp.Sc. degrees in Computer Science from Institut Teknologi Sepuluh Nopember, Indonesia, in 2016 and 2018, respectively. She earned her Ph.D. in Industrial and Data Engineering, a joint degree program between Pukyong National University and Pusan National University, South Korea, in 2025.

Dr. Effendi is currently a full-time lecturer in Robotics and Artificial Intelligence Engineering at Airlangga University, Indonesia. Her primary research interests include process mining, artificial intelligence, industrial data analytics, health informatics, and knowledge engineering. She has published over 50 works, including research papers, book chapters, and teaching materials, which are accessible through Scopus, ORCID, Google Scholar, WoS, and SINTA. She is currently working on a research project funded by the Airlangga Research Fund, focusing on smart scheduling and customer segmentation using machine learning.

Minsoo Kim received his B.S., M.S., and Ph.D. degrees in Industrial Engineering from Seoul National University, Republic of Korea, in 1996, 1998, and 2002, respectively. His research interests include technology innovation, process mining, big data, and deep learning. He is currently a full-time professor at the Department of Industrial and Data Engineering, Pukyong National University, Korea. Before joining Pukyong National University in 2004, he worked as a software developer and system architect in business process management systems at HandySoft Corp.

Dr. Kim is a member of the Korean Institute of Industrial Engineers, the Korea Technology Innovation Society, and the Korean Society for E-Business Studies. He is actively working several research projects funded from the National Research Foundation of Korea (NRF).