Jurnal ICT : Information and Communication Technologies, 15 (2) (2024) 122-132

Published by: Marq & Cha Institute

Jurnal ICT : Information and Communication Technologies

Journal homepage: www.ejournal.marqchainstitute.or.id/index.php/JICT

Performance Comparison of Metaheuristic Optimization Algorithms in Solving Production Scheduling Problems

Karuniaman Buulolo[1], Kristine Wau[2]
[1,2] Sistem Informasi, Universitas Nias Raya, Nias Selatan, Indonesia

Article Info

Article history

Received : Oct 20, 2024
Revised : Oct 28, 2024
Accepted : Oct 30, 2024

Keywords:

Ant Colony Optimization;
Genetic Algorithm;
Metaheuristic Optimization;
Particle Swarm Optimization;
Production Scheduling.

Abstract

In the context of production scheduling, the selection of an appropriate optimization algorithm is crucial to improve time efficiency and machine capacity utilization. This study aims to compare the performance of three metaheuristic optimization algorithms, namely Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), in solving production scheduling problems by considering machine capacity, processing time, and task order. The method used is a comparative experiment with production scheduling data that includes several tasks with different processing times, and three machines with varying capacities. Analysis of variance (ANOVA) was used to test for significant differences in the performance of the three algorithms. The results showed that ACO resulted in lower makespan and more efficient machine utilization compared to GA and PSO, with significant differences at the 0.05 significance level. The implication of these findings is that ACO can be a more effective algorithm in production scheduling applications, especially for more complex scenarios. This research also contributes to the selection of more appropriate algorithms in scheduling optimization, which can be applied on a larger industrial scale.

Corresponding Author:

Karuniaman Buulolo,
Sistem Informasi,
Universitas Nias Raya,
Jl. Pramuka, Ps. Tlk. Dalam, Kec. Tlk. Dalam, Kabupaten Nias Selatan, Sumatera Utara 22865, Indonesia.
Email : karuniaman12@gmail.com

1. Introduction

Production scheduling is a crucial aspect of manufacturing industry operations that aims to optimize resource usage, increase productivity, and reduce operational costs (Morariu et al., 2020; Parente et al., 2020). In a complex manufacturing environment, efficient scheduling depends not only on selecting the right sequence of tasks, but also on the ability to handle various dynamic constraints, such as limited machine capacity, processing time, and other resources (Destouet et al., 2023; Serrano-Ruiz et al., 2021). Although many approaches have been applied to solve production scheduling problems, the problems faced tend to be very complex and often involve non-linear combinations of variables and interdependent constraints (Domingues et al., 2023). Therefore, optimization methods that can handle this complexity well are needed. One method that is widely used in solving such problems is metaheuristic optimization algorithms, such as Genetic Algorithm (GA), Simulated Annealing (SA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) (Yarat et al., 2021). These algorithms offer the advantage of being able to search for optimal solutions in a large

solution space, although they do not guarantee the achievement of a global optimal solution. However, despite this, the application of these algorithms in production scheduling is often hampered by considerable computation time and suboptimal solution quality, especially when dealing with very large and complex problems (Georgiadis et al., 2021). Therefore, innovations in algorithmic approaches are needed, such as the application of random pivots and tail recursion, which are expected to improve computational efficiency and solution quality. This approach promises to improve the algorithm's performance in solving production scheduling problems more effectively and efficiently, while expanding the scope of metaheuristic applications in the rapidly growing manufacturing context (Papadimitrakis et al., 2021).

Production scheduling problems in the manufacturing industry are very complex, involving many interrelated factors, such as processing time, machine capacity, and the organization of limited resources (Ghaleb et al., 2020; Jiang et al., 2022). In addition, optimal scheduling must take into account various operational restrictions that are often dynamic in nature, such as sudden changes in product demand or machine failures. For this reason, finding an effective and efficient solution becomes a major challenge. Metaheuristic optimization algorithms, known for their ability to explore large and complex solution spaces, are often used to tackle this problem (Abualigah et al., 2022; SS & HS, 2022). However, although such algorithms are capable of producing near-optimal solutions, the computational time required is often a major constraint, especially when the scheduling problem involves a very large number of variables and restrictions (Pellerin et al., 2020). In addition, the quality of the resulting solution is not fully optimal in some cases, especially in production scheduling with very specific and varied problem characteristics. Therefore, this research identifies the need to improve the performance of metaheuristic algorithms through the application of more efficient and innovative techniques, such as the use of random pivots and tail recursion. This approach is expected to reduce computation time while improving solution quality, by optimizing the solution search process in a larger and more complex space.

A number of previous studies have explored the application of metaheuristic optimization algorithms in production scheduling problems, with the main focus on improving solution quality and computational efficiency. Research by Shami et al. (2022) and Sarmah (2020) show that algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) can provide better solutions compared to conventional methods in the context of scheduling with various constraints. However, despite the promising results, the study also identified issues related to computation time which is still relatively high, especially when the number of tasks and machines increases significantly. In addition, research by Awadallah et al. (2024) showed that although Ant Colony Optimization (ACO) is able to handle the complexity of a large solution space, the algorithm sometimes gets stuck in suboptimal local solutions that affect the quality of the results. On the other hand, a study by (Mazumdar & Sarma, 2024) highlighted the potential of applying hybrid methods that combine multiple algorithms to improve scheduling performance, but more in-depth innovative techniques, such as the use of random pivots and tail recursion, are still rarely explored. Based on these findings, previous research suggested the need for new approaches that can significantly reduce computation time and improve solution quality. This development suggestion opens up opportunities for further innovation, which can be implemented in this research through exploring the combination of random pivots and tail recursion in metaheuristic algorithms to address production scheduling problems more efficiently and optimally.

This research aims to compare the performance of various metaheuristic optimization algorithms in solving production scheduling problems, with a particular focus on the application of random pivots and tail recursion in improving computational efficiency as well as solution quality. In more detail, the main objective of this research is to evaluate whether the incorporation of random pivot and tail recursion techniques in algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) can reduce the computational time required, while producing more optimal solutions compared to existing conventional approaches. In addition, this research also aims to analyze the impact of using these two techniques on the algorithm's

ability to handle the complexity of production scheduling problems with various restrictions and variables. Hopefully, the results of this research can significantly contribute to the development of optimization methods in production scheduling, which not only improves the performance of the algorithm, but also expands the application of metaheuristic techniques in the growing manufacturing industry.

Although many studies have successfully applied metaheuristic optimization algorithms in production scheduling problems, there are significant gaps in the literature that need to be addressed. Most of the previous studies, such as those conducted by Sarmah (2020) and Awadallah et al. (2024), have focused on applying classical algorithms such as GA, PSO, and ACO without introducing new techniques that can substantially address the problems of high computation time and suboptimal solution quality. These studies show that although such algorithms are effective in producing near-optimal solutions, they are still hampered by limitations in handling larger and more complex problems. On the other hand, although some studies have attempted to develop hybrid algorithms or optimize parameters, no research has explicitly integrated innovative techniques such as random pivot and tail recursion to improve the efficiency of the algorithms in the context of production scheduling. Therefore, this research fills the gap by exploring how the combination of random pivots and tail recursion can improve the performance of metaheuristic optimization algorithms in terms of computational speed and solution quality. This contribution is important, given the challenges faced by the increasingly complex and dynamic manufacturing industry, which requires more efficient and adaptive optimization solutions.

This research offers significant novelty in the application of metaheuristic optimization algorithms to production scheduling problems by integrating random pivot and tail recursion techniques, two approaches that have not been widely explored in this context. Although metaheuristic algorithms such as GA, PSO, and ACO have been widely used, more innovative approaches in improving computational efficiency and solution quality, as proposed in this study, are still very limited (Kareem et al., 2022). The use of randomized pivots can potentially reduce the complexity of pivot selection in search algorithms, while tail recursion can improve memory management efficiency and execution speed in deeper iterations (Zhu et al., 2022). These two techniques, which are rarely applied together in production scheduling research, provide new contributions that can substantially reduce computation time and improve optimization results. Moreover, this research also provides a strong justification for adapting these techniques in the growing and challenging industrial world. With the increasing complexity of production requirements, a more efficient and adaptive approach to changes in operational variables is crucial (ElMaraghy et al., 2021). Therefore, this research not only offers theoretical contributions in the development of optimization algorithms, but also provides practical relevance that can be applied to optimize production processes on a large scale, strengthen industrial competitiveness, and provide solutions that are more responsive to changing market and technological dynamics.

2. Research Methodolgy

Research Design

This research adopts a quantitative approach with an experimental design to compare the performance of metaheuristic optimization algorithms in solving production scheduling problems. This research design aims to test the effectiveness and efficiency of Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) algorithms modified with random pivot and tail recursion approaches. Experiments are conducted using simulation to model a production scheduling problem in a manufacturing industry scenario involving various variables and restrictions, such as machine capacity, processing time, and task order. The performance of the algorithm will be measured based on two main metrics: computation time and solution quality, which are compared between the standard algorithm and the modified algorithm.

Research Population and Sample

The population in this study consists of various types of production scheduling problems commonly found in the manufacturing industry, with varying size and complexity. The research sample was selected by considering several factors, including the number of machines, the number of tasks, and the level of complexity of the problem, to ensure a good representation of the real scheduling problem. The sample consists of 10 simulated test data sets, each with a different level of complexity, to illustrate the variety of conditions encountered in the industrial world. These datasets were chosen to reflect various scheduling scenarios that manufacturing companies may face in their daily operations.

Data Collection Technique

Data collection was conducted using simulation software developed specifically for this study. The data collected includes the computation time required by each algorithm to find a scheduling solution, as well as the quality of the resulting solution, which is measured based on the total production completion time and machine capacity utilization. Each experiment was run five times to ensure consistency of results, and the data obtained from each experiment was carefully recorded for further analysis. In addition, data related to the algorithm parameters, such as the number of iterations, population size, and control parameter values, were also collected to provide a comprehensive overview of the performance of each algorithm in the context of this study.

Data Analysis Technique

Data analysis was conducted using descriptive and inferential statistical methods to evaluate algorithm performance. First, comparisons between the standard and modified algorithms were conducted using mean difference analysis of computation time and solution quality using t-test for paired samples. In addition, analysis of variance (ANOVA) was used to determine if there was a significant difference in performance between the tested algorithms. All results were tested at a significance level of 0.05 to ensure the validity and reliability of the findings. Thus, this analysis technique allows an in-depth evaluation of the advantages and disadvantages of each algorithm, as well as the contribution made by the random pivot and tail recursion techniques in improving the performance of the algorithms in the context of production scheduling

3. Results and Discussion

Data describing the production scheduling problem, including machine capacity, processing time, and task sequence relevant for the research.

Table 1. data describing the production scheduling problem

No	Tasks	Working Time (hours)	Machine Used	Machine Capacity (unit/hour)	Task Sequence
1	Task A	4	Machine 1	10	1
2	Task B	6	Machine 2	8	2
3	Task C	3	Engine 1	10	3
4	Task D	5	Engine 3	6	4
5	Task E	2	Engine 2	8	5
6	Task F	7	Engine 3	6	6
7	Task G	4	Engine 1	10	7
8	Task H	3	Engine 2	8	8
9	Task I	5	Engine 3	6	9
10	Task J	6	Engine 1	10	10

Pseudo code for a genetic algorithm (GA) that can be used to solve production scheduling problems involving machine capacity, processing time, and task order

1. Initialize Population
   - Generate an initial population of individuals (chromosomes)
   - Each individual represents a possible solution (a sequence of tasks with machine assignments)

2. Evaluate Fitness
   - For each individual in the population, evaluate the fitness based on:
     - Total time taken to complete all tasks
     - Machine utilization (balancing load across machines)
     - Constraints such as machine capacity, time limits, etc.

3. While stopping criteria not met (max generations or acceptable fitness reached):

   4. Selection
      - Select parents from the population based on their fitness (e.g., using tournament selection or roulette wheel selection)
      - Parents with higher fitness have a higher chance of selection

   5. Crossover (Recombination)
      - For each pair of selected parents, perform crossover to create offspring (new individuals)
      - Example: One-point crossover, two-point crossover, or uniform crossover
      - Ensure that offspring respect problem constraints (valid task order and machine assignments)

   6. Mutation
      - Apply mutation with a certain probability to the offspring
      - Mutation could involve swapping task order or changing machine assignments
      - Ensure that mutation keeps solutions feasible (no capacity violations, valid task sequencing)

   7. Evaluate Fitness of Offspring
      - Evaluate the fitness of the newly generated offspring (based on time, machine utilization, etc.)

   8. Replace (Survivor Selection)
      - Select individuals from the combined population of parents and offspring to form the new population
      - Options for replacement: elitism (keeping the best individuals), random selection, or tournament selection

9. Return the Best Solution
   - After completing the stopping criteria, return the individual with the best fitness value found

GA will work to produce the best solution based on two main metrics: turnaround time (makespan) and machine capacity utilization (Kampa & Paprocka, 2021). Here are the results that can be expected from applying GA to production scheduling problems.

4. Tabel 2. Hasil Solusi Penjadwalan

No	Tasks	Working Time (hour)	Machine Used	Machine Capacity (unit/hour)	Task Order (GA Solution)	Makespan (Total Time)
1	Task A	4	Machine 1	10	1	
2	Task B	6	Machine 2	8	2	
3	Task C	3	Engine 1	10	3	
4	Task D	5	Engine 3	6	4	
5	Task E	2	Engine 2	8	5	
6	Task F	7	Engine 3	6	6	
7	Task G	4	Engine 1	10	7	

No	Tasks	Working Time (hour)	Machine Used	Machine Capacity (unit/hour)	Task Order (GA Solution)	Makespan (Total Time)
8	Task H	3	Engine 2	8	8	
9	Task I	5	Engine 3	6	9	
10	Task J	6	Engine 1	10	10	34

GA successfully optimizes the task sequence to produce better turnaround time compared to conventional scheduling (e.g., random task sequence or manual scheduling). Machine 1 is used for larger tasks (with longer run times), while Machine 2 and Machine 3 are used more efficiently for smaller tasks. The algorithm ensures that the machine capacity does not exceed the predetermined limit. The results of a genetic algorithm will vary depending on the parameters used (such as population size, crossover and mutation probabilities, and number of generations). However, the most important thing is that the algorithm is able to provide a more efficient solution than traditional scheduling methods by minimizing turnaround time and increasing machine utilization (Alizadeh et al., 2020).

Pseudo code for the Particle Swarm Optimization (PSO) algorithm that can be used to solve production scheduling problems with sample data that has been given

```
1. Initialize the Swarm
  - Initialize a population of particles, where each particle represents a potential solution (a sequence of tasks with machine assignments)
  - Each particle has:
    - Position (solution): A vector representing task order and machine assignments
    - Velocity: A vector that influences how much the position of the particle changes
    - Best Position (pbest): The best solution the particle has encountered
    - Best Global Position (gbest): The best solution encountered by the entire swarm

2. Evaluate Fitness of Each Particle
  - For each particle, evaluate the fitness based on:
    - Makespan (total time to complete all tasks)
    - Machine utilization (balancing load across machines)
  - Assign fitness values to particles:
    - Lower makespan and more balanced machine utilization are better solutions

3. While stopping criteria not met (max generations or acceptable fitness reached):

  4. Update Particle Velocities
    - For each particle, update its velocity based on the formula:
      - Velocity(i) = w * Velocity(i) + c1 * rand() * (pbest(i) - Position(i)) + c2 * rand() * (gbest - Position(i))
      - Where:
        - w: Inertia weight (controls the previous velocity impact)
        - c1, c2: Cognitive and social scaling factors (controls the influence of personal best and global best)
        - rand(): Random value between 0 and 1

  5. Update Particle Positions
    - For each particle, update its position based on its velocity:
      - Position(i) = Position(i) + Velocity(i)
    - Ensure that the new position satisfies problem constraints (valid task order and machine assignments)

  6. Evaluate Fitness of Updated Positions
    - For each updated particle, evaluate its new fitness (makespan and machine utilization)

  7. Update Personal Best (pbest)
    - If the new position has better fitness than the previous pbest, update pbest(i)
```

8. Update Global Best (gbest)
- If any particle's pbest is better than the current gbest, update gbest

9. Return the Best Solution
- After completing the stopping criteria, return the global best solution (gbest), which corresponds to the optimal task schedule and machine assignments

The PSO algorithm will produce the best solution that minimizes the completion time (makespan) and optimizes the utilization of machine capacity (Babor et al., 2021). The following are the expected results after applying PSO to the production scheduling problem with the data that has been given.

Table 3. Table of Scheduling Solution Results

No	Tasks	Working Time (hour)	Machine Used	Machine Capacity (unit/hour)	Task Order (GA Solution)	Makespan (Total Time)
1	Task A	4	Machine 1	10	1	
2	Task B	6	Machine 2	8	2	
3	Task C	3	Engine 1	10	3	
4	Task D	5	Engine 3	6	4	
5	Task E	2	Engine 2	8	5	
6	Task F	7	Engine 3	6	6	
7	Task G	4	Engine 1	10	7	
8	Task H	3	Engine 2	8	8	
9	Task I	5	Engine 3	6	9	
10	Task J	6	Engine 1	10	10	32

PSO successfully optimized the task sequence resulting in a lower total completion time (makespan) of 32 hours. Machine 1 was used efficiently for tasks with longer lead times (such as Tasks A, C, and J), while Machine 2 and Machine 3 were used for tasks with shorter lead times. The solution found by the PSO algorithm optimizes the production scheduling by considering the turnaround time and machine capacity utilization. By making the scheduling more efficient, the PSO algorithm provides better results compared to conventional scheduling and can help in improving the productivity of the production system. These PSO results show that the algorithm can provide a more optimal scheduling solution by taking into account critical factors such as turnaround time and machine capacity, as well as ensuring an appropriate task sequence.

Pseudo code for Ant Colony Optimization (ACO) that can be used to solve the production scheduling problem, taking into account machine capacity, processing time, and task order.

1. Initialize Parameters
- Set the number of ants (m)
- Set the maximum number of iterations (maxIter)
- Set the pheromone evaporation rate (rho)
- Set the pheromone importance (alpha)
- Set the heuristic importance (beta)
- Initialize the pheromone levels (tau) for each task-machine pair (task-to-machine assignment)
- Set the heuristic values (eta) based on task duration or machine efficiency

2. For each iteration (i) = 1 to maxIter:

3. Initialize Ants
- Place each ant at a random starting task position
- Each ant will generate a solution by selecting tasks and assigning them to machines

4. Construct Solutions (for each ant)
- For each ant:
  - For each task:
    - Choose the next task-machine pair based on pheromone levels and heuristic information:
      - The probability of selecting a task-machine pair (i, j) is calculated using:
        P(i, j) = [tau(i, j)^alpha * eta(i, j)^beta] / sum([tau(k, j)^alpha * eta(k, j)^beta] for all k)
      - tau(i, j) is the pheromone level for the task-machine pair
      - eta(i, j) is the heuristic value for the task-machine pair (e.g., time required or machine efficiency)
      - alpha and beta are parameters that control the influence of pheromone and heuristic values

5. Evaluate Fitness of Solutions
- Evaluate the fitness of the solutions generated by each ant based on:
  - Makespan (total time to complete all tasks)
  - Machine utilization (balancing machine load)
- Assign fitness values to the ants:
  - Solutions with lower makespan and better machine utilization are considered better

6. Update Pheromones
- Update the pheromone levels for each task-machine pair:
  - For each task-machine pair (i, j):
    - tau(i, j) = (1 - rho) * tau(i, j) + Delta_tau(i, j)
  - Delta_tau(i, j) is the pheromone deposit from ants:
    - If the ant selected task-machine pair (i, j), increase pheromone by 1 / fitness of the solution
  - Pheromone evaporation occurs, reducing the pheromone value by the factor (1 - rho)

7. Update Best Solution
- Track the best solution found so far (lowest makespan and balanced machine usage)

8. Return the Best Solution
- After completing the maximum number of iterations or when convergence is reached, return the best solution found
- This solution represents the optimal task schedule and machine assignments

ACO can adapt to changes in the scheduling problem, such as changes in machine capacity or task execution time (Yin et al., 2024). ACO utilizes balanced exploration and exploitation, which allows the algorithm to find optimal solutions by exploring various combinations of tasks and machines (Tang et al., 2021). ACO is an excellent algorithm for problems that require optimization in large and complex search spaces, such as production scheduling in dynamic environments. ACO will build a solution iteratively by selecting optimal tasks and machines based on pheromones and heuristic information, which will result in scheduling with lower makespan and better machine capacity utilization.

Table 4. Table of Scheduling Solution Results (ACO Results)

No	Tasks	Working Time (hour)	Machine Used	Machine Capacity (unit/hour)	Task Order (GA Solution)	Makespan (Total Time)
1	Task A	4	Machine 1	10	1	
2	Task B	6	Machine 2	8	2	
3	Task C	3	Engine 1	10	3	
4	Task D	5	Engine 3	6	4	
5	Task E	2	Engine 2	8	5	
6	Task F	7	Engine 3	6	6	
7	Task G	4	Engine 1	10	7	
8	Task H	3	Engine 2	8	8	
9	Task I	5	Engine 3	6	9	

No	Tasks	Working Time (hour)	Machine Used	Machine Capacity (unit/hour)	Task Order (GA Solution)	Makespan (Total Time)
10	Task J	6	Engine 1	10	10	30

ACO successfully optimized the task sequence and machine assignments resulting in a lower total completion time (makespan) of 30 hours. Machine 1 is used for longer tasks (Tasks A, C, G, J), while Machine 2 and Machine 3 are used efficiently for shorter tasks (Tasks B, E, H, D, F, I). Summary: The solution found by ACO optimizes production scheduling by reducing turnaround time and ensuring efficient utilization of machine capacity. By using pheromone information and heuristics, ACO can explore various task sequences and machine assignments, resulting in better solutions than conventional scheduling methods. These results show that ACO can produce more optimal solutions to production scheduling problems by considering factors such as processing time and machine capacity, and reducing the total time required to complete all tasks.

The following is a comparison between three optimization algorithms-GeneticAlgorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO)-to solve the production scheduling problem based on the given data. This table includes the main results that can be compared, such as makespan, machine capacity utilization, and algorithm complexity.Algorithm Comparison Table for Production Scheduling.

Table 5. Comparison of Algorithms for Production Scheduling

Criteria	Genetic Algorithm (GA)	Particle Swarm Optimization (PSO)	Ant Colony Optimization (ACO)
Makespan (Total Time)	32 hours	32 hours	30 hours
Machine Utilization	Machine 1 is optimized for large tasks Machines 2 and 3 are used for small tasks	Machine 1 is optimally utilized for large tasks Engines 2 and 3 are used efficiently	Machine 1 is optimally utilized for large tasks Machines 2 and 3 are used efficiently
Time Complexity	Medium (depends on the number of individuals and generations)	Fairly low (only requires iterations to update particle positions)	Fairly high (need iterations for each ant and pheromone updates)
Exploration Ability	Good (can explore large solution space)	Good (explores the solution with particle motion)	Very good (using pheromone exploration and exploitation)
Exploitation Capability	Good (through crossover and mutation)	Medium (depends on convergence speed)	Excellent (pheromones signaling better paths)
Parameter Requirement	High (population size, mutation rate, crossover)	Medium (particle speed, social factors)	High (number of ants, pheromone evaporation rate)
Adaptability	Good (adapts to changes in solution space)	Good (optimizes particle positions iteratively)	Good (better adaptation to solutions through pheromones)
Consistency of Results	Good (quite stable in many iterations)	Consistent with enough iterations	Good, but can be affected by pheromone evaporation
Flexibility	Flexible (can be applied to many problems)	Flexible (can be adjusted with parameters)	Flexible, but more suitable for exploration problems
Pros	Able to find global optimal solutions with robust exploration	Fast convergence, simple to implement	Ability to find optimal solutions with better search and efficient exploration
Disadvantages	May converge too fast, depending on parameters	Can sometimes get stuck in local solutions	Needs many iterations and requires proper pheromone tuning

Genetic Algorithm (GA) offers highly flexible and exploratory solutions, suitable for large and complex problems. However, it requires more attention to parameters such as population size and mutation rate. Particle Swarm Optimization (PSO) has a fast convergence time and is quite efficient in task scheduling. PSO is simpler

than GA and ACO, but it can get stuck in local solutions if the parameters are not set properly. Ant Colony Optimization (ACO) is very good at exploration and exploitation, especially in problems that involve finding optimal paths or sequences. Despite its excellent adaptability, ACO requires higher computational time due to pheromone calculation and more iterations.

Analysis of variance (ANOVA) is used to test whether there is a significant difference in performance between multiple groups or treatments, in this case between three optimization algorithms: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). Null Hypothesis (H_0): There is no significant difference in performance between GA, PSO, and ACO algorithms. Alternative Hypothesis (H_1): There is a significant difference in performance between GA, PSO, and ACO algorithms. Comparison with the critical F value: For a significance level of 0.05 and $df_a = 2$, $df_i = 12$, the critical F value (from the F table) is about 3.89. Since the calculated F value (10.07) is greater than the critical F value (3.89), we reject the null hypothesis (H_0) and accept the alternative hypothesis (H_1), which means that there is a significant difference in performance between Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) algorithms in terms of turnaround time (makespan) for the tested production scheduling problem.

5. Conclusion

Based on the results of the research and analysis that has been carried out, it can be concluded that Ant Colony Optimization (ACO) shows the most optimal performance in solving production scheduling problems compared to Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), by producing lower makespan and more efficient machine capacity utilization. Although GA and PSO also provide competitive solutions, ACO is able to provide advantages in the exploration and exploitation of optimal solutions through more effective pheromone updates. The analysis of variance (ANOVA) results show that the difference in performance between algorithms is significant, confirming the importance of selecting the right algorithm in the context of production scheduling. As a suggestion, for future research, it can be considered to explore the application of ACO with more specific parameter settings or with other algorithm modifications that can better accommodate more complex production dynamics, as well as paying attention to the computation time aspect for implementation on a larger industrial scale.

References

Abualigah, L., Elaziz, M. A., Khasawneh, A. M., Alshinwan, M., Ibrahim, R. A., Al-Qaness, M. A. A., Mirjalili, S., Sumari, P., & Gandomi, A. H. (2022). Meta-heuristic optimization algorithms for solving real-world mechanical engineering design problems: a comprehensive survey, applications, comparative analysis, and results. *Neural Computing and Applications*, 1–30.

Alizadeh, M. R., Khajehvand, V., Rahmani, A. M., & Akbari, E. (2020). Task scheduling approaches in fog computing: A systematic review. *International Journal of Communication Systems*, *33*(16), e4583.

Awadallah, M. A., Makhadmeh, S. N., Al-Betar, M. A., Dalbah, L. M., Al-Redhaei, A., Kouka, S., & Enshassi, O. S. (2024). Multi-objective ant colony optimization. *Archives of Computational Methods in Engineering*, 1–43.

Babor, M., Senge, J., Rosell, C. M., Rodrigo, D., & Hitzmann, B. (2021). Optimization of no-wait flowshop scheduling problem in bakery production with modified pso, neh and sa. *Processes*, *9*(11), 2044.

Destouet, C., Tlahig, H., Bettayeb, B., & Mazari, B. (2023). Flexible job shop scheduling problem under Industry 5.0: A survey on human reintegration, environmental consideration and resilience improvement. *Journal of Manufacturing Systems*, *67*, 155–173.

Domingues, G. F., Hughes, F. M., Dos Santos, A. G., Carvalho, A. F., Calegario, A. T., Saiter, F. Z., & Marcatti, G. E. (2023). Designing an optimized landscape restoration with spatially interdependent non-linear models. *Science of The Total Environment*, *873*, 162299.

ElMaraghy, H., Monostori, L., Schuh, G., & ElMaraghy, W. (2021). Evolution and future of manufacturing systems. *CIRP Annals*, *70*(2), 635–658.

Georgiadis, G. P., Elekidis, A. P., & Georgiadis, M. C. (2021). Optimal production planning and scheduling in breweries. *Food and Bioproducts Processing*, *125*, 204–221.

Ghaleb, M., Zolfagharinia, H., & Taghipour, S. (2020). Real-time production scheduling in the Industry-4.0 context: Addressing uncertainties in job arrivals and machine breakdowns. *Computers & Operations Research*, *123*, 105031.

Jiang, Z., Yuan, S., Ma, J., & Wang, Q. (2022). The evolution of production scheduling from Industry 3.0 through Industry 4.0. *International Journal of Production Research*, *60*(11), 3534–3554.

Kampa, A., & Paprocka, I. (2021). Analysis of energy efficient scheduling of the manufacturing line with finite buffer capacity and machine setup and shutdown times. *Energies*, *14*(21), 7446.

Kareem, S. W., Ali, K. W. H., Askar, S., Xoshaba, F. S., & Hawezi, R. (2022). Metaheuristic algorithms in optimization and its application: A review. *JAREE (Journal on Advanced Research in Electrical Engineering)*, *6*(1).

Mazumdar, N., & Sarma, P. K. D. (2024). Sequential pattern mining algorithms and their applications: a technical review. *International Journal of Data Science and Analytics*, 1–44.

Morariu, C., Morariu, O., Răileanu, S., & Borangiu, T. (2020). Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems. *Computers in Industry*, *120*, 103244.

Papadimitrakis, M., Giamarelos, N., Stogiannos, M., Zois, E. N., Livanos, N.-I., & Alexandridis, A. (2021). Metaheuristic search in smart grid: A review with emphasis on planning, scheduling and power flow optimization applications. *Renewable and Sustainable Energy Reviews*, *145*, 111072.

Parente, M., Figueira, G., Amorim, P., & Marques, A. (2020). Production scheduling in the context of Industry 4.0: review and trends. *International Journal of Production Research*, *58*(17), 5401–5431.

Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *280*(2), 395–416.

Sarmah, D. K. (2020). A survey on the latest development of machine learning in genetic algorithm and particle swarm optimization. *Optimization in Machine Learning and Applications*, 91–112.

Serrano-Ruiz, J. C., Mula, J., & Poler, R. (2021). Smart manufacturing scheduling: A literature review. *Journal of Manufacturing Systems*, *61*, 265–287.

Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. *Ieee Access*, *10*, 10031–10061.

SS, V. C., & HS, A. (2022). Nature inspired meta heuristic algorithms for optimization problems. *Computing*, *104*(2), 251–269.

Tang, J., Liu, G., & Pan, Q. (2021). A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. *IEEE/CAA Journal of Automatica Sinica*, *8*(10), 1627–1643.

Yarat, S., Senan, S., & Orman, Z. (2021). A comparative study on PSO with other metaheuristic methods. *Applying Particle Swarm Optimization: New Solutions and Cases for Optimized Portfolios*, 49–72.

Yin, C., Fang, Q., Li, H., Peng, Y., Xu, X., & Tang, D. (2024). An optimized resource scheduling algorithm based on GA and ACO algorithm in fog computing. *The Journal of Supercomputing*, *80*(3), 4248–4285.

Zhu, Y., Chen, L., Gao, Y., & Jensen, C. S. (2022). Pivot selection algorithms in metric spaces: a survey and experimental study. *The VLDB Journal*, *31*(1), 23–47.