Available Online at www.e-iph.co.uk
Indexed in Clarivate Analytics WoS, and ScienceOPEN

E-B Environment - Behaviour Proceedings Journal

KICSS2024

Kedah International Conference on Social Sciences:
https://kicss2024.wixsite.com/kicss2024

e-IPH e-International Publishing House Ltd., United Kingdom

InforMaTIC2024: Information Management and Technology International Conference
Virtual Conference, UiTM Kedah, Malaysia

Organised by: Universiti Teknologi MARA, Kedah, Malaysia

An Effective Job Scheduling for Maximum Resource Utilisation with Particle Swarm Optimisation

Mansir Abubakar[1*], Alwatben Batoul Rashed[2], Sanusi Abu Darma[3], Mardiyya Lawal Bagiwa[3]

[1] Faculty of Computer Science and Mathematics, Universiti Teknologi MARA, Shah Alam, Malaysia
[2] Department of Information Technology, College of Computer, Qassim University, Buraydah, Saudi Arabia
[3] College of Computing and Information Science, Al-Qalam University Katsina, Nigeria

mansir@uitm.edu.my, batool.alwtban@gmail.com, darmasanusiabu@yahoo.com, mardiyyalawalbagiwa@auk.edu.ng
Tel: +60132959860

Abstract
This paper explores the uses of the Particle Swarm Optimization (PSO) algorithm to minimize job completion time during scheduling tasks. The goal is to reduce the time needed to complete all tasks. The PSO algorithm can achieve fast convergence due to its swarm intelligence behavior and its capability to search in a global space. Leveraging its global search ability and fast convergence, PSO effectively optimizes schedules, leading to improved resource management and cost reduction. The method shows strong potential for industries like manufacturing and logistics and demonstrates broad applicability across various complex scheduling domains.

Keywords: job scheduling, optimization, PSO, resource utilization.

DOI: https://doi.org/10.21834/e-bpj.v10iSI31.6939

1.0 Introduction

PSO is versatile and can be adapted to solve many optimization problems, especially where the solution space is large or poorly understood. PSO is widely used due to its simplicity, efficiency, and ability to handle a variety of optimization problems. It requires few parameters to tune, making it more accessible than other complex optimization algorithms like genetic algorithms. PSO is useful in applications such as neural network training, engineering design, and financial modeling. However, it can sometimes converge prematurely, especially in problems with many local optima; it requires careful tuning of parameters like inertia weight and acceleration coefficients to achieve the best results. PSO is generally simpler to implement and requires fewer computational resources, as it only updates velocities and positions, whereas Genetic Algorithms (GA) use more complex operations like crossover and mutation, making it more computationally intensive. PSO is often faster in convergence but may suffer from premature convergence, especially in multi-modal problems Hu et al., 2022). GA, with its exploration through recombination and mutation, is typically better at exploring a diverse solution space, making it less likely to get stuck in local optima. However, GA can be slower to converge and may require more fine-tuning of its operators for effective performance. Both techniques are versatile, with GA being preferred for highly complex or combinatorial problems, while PSO is favored for continuous optimization tasks.

Efficient scheduling strategies can significantly enhance resource utilization, reduce completion times, and improve system performance. Efficient job scheduling is important because it can increase productivity, reduce operational costs, and improve overall

eISSN: 2398-4287 © 2025. The Authors. Published for AMER by e-International Publishing House, Ltd., UK. This is an open access article under the CC BY-NC-ND license (http://c5eativecommons.org/licenses/by-nc-nd/4.0/). Peer–review under responsibility of AMER (Association of Malaysian Environment-Behaviour Researchers
DOI: https://doi.org/10.21834/e-bpj.v10iSI31.6939

system efficiency. However, achieving optimal job scheduling is challenging due to the complexity of the tasks, the dynamic nature of resource availability, and the need to balance multiple conflicting objectives. Traditional scheduling systems frequently struggle with scalability and adaptability, making it necessary to explore more advanced ways. Particle Swarm Optimisation (PSO), a meta-heuristic algorithm inspired by the social behavior of birds flocking or fish schooling, has gained popularity for solving difficult scheduling issues due to its simplicity and effectiveness Potluri et al., 2023). PSO operates by launching a swarm of particles that search the solution space for the optimal solution. Each particle adjusts its position based on its own and neighboring particles' experiences, achieving an effective balance of exploration and exploitation. One of PSO's key advantages in solving scheduling problems is its ability to handle multi-objective optimization efficiently. It does not require gradient information; hence, it is suitable for problems involving discontinuous or non-differentiable objective functions. Wang et al. (2022) suggested a hybrid approach that combines genetic algorithms and an improved PSO, resulting in significant reductions in task completion time and load balancing in cloud computing environments.

Similarly, Zhang et al. (2022) developed an adaptive discrete PSO algorithm for flexible job shop scheduling that enhanced convergence speed and schedule efficiency. Inefficient job scheduling causes tasks to take longer to complete because they are not appropriately sequenced and allocated to resources. This might cause delays in overall paper timelines and lower productivity. When jobs are planned without considering the optimal sequence or resource capabilities, certain jobs are often left waiting while resources sit idle. To reduce these delays, an optimized scheduling system is required, which ensures that each job is assigned at the appropriate time to the most suitable resource, lowering overall make-span and enhancing efficiency.

In a typical formulation of JSP, let $J = \{J1, J2, \ldots, Jn\}$ represent the set of jobs to be scheduled, where each job *Ji* has a sequence of tasks that must be executed in a specific order. The relationship between job *J* and its associated processing times *t* is critical in determining the optimal schedule. The objective function in JSP typically aims to minimize the makespan, which is calculated as the maximum completion time among all jobs. This involves sequencing tasks like those of the resources that are utilized efficiently, and tasks are completed as early as possible without violating any constraints. In short, JSP addresses the difficulty of optimizing resource allocation and task sequencing for efficient job scheduling. However, JSP aims to maximize operational efficiency and productivity in diverse industrial and manufacturing settings by developing an appropriate objective function that considers the relationships between jobs J, task processing times *t*, and limitations imposed by available resources, which are critical setbacks of this approach. Therefore, the primary goal of this study is to minimize the make-span, or the total time required to accomplish all activities, to maximize resource utilization and satisfy job deadlines.

2.0 Literature Review

According to Hu et al. (2022), they developed a unique hybrid optimization algorithm that combines Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) to address the complexity of job scheduling in cloud computing environments. The representation strategy encodes tasks and resources with chromosomes for GA and particles for PSO, respectively. This dual representation allows leveraging the benefits of both algorithms: GA's robust global search capabilities and PSO's fast convergence and local search efficiency. The fitness function in their study was precisely built to minimize makespan while improving load balancing across virtual machines. By focusing on these crucial parameters, the algorithm aims to optimize overall performance and resource utilization in cloud environments. The hybrid approach uses GA to successfully explore a large solution space and PSO to exploit the most promising portions of this space, ensuring a balance between exploration and exploitation. For computing performance, the hybrid algorithm outperformed the solo implementations of GA and PSO. This hybrid technique produced solutions with faster convergence rates and higher accuracy. The findings revealed a significant reduction in makespan and improved load distribution, which is necessary for successful job scheduling in cloud computing. This hybrid approach improves solution quality while providing a scalable and adaptable framework for job scheduling in most computational environments.

The study by Li et al. (2020) introduces an improved PSO technique combined with GA principles for job shop scheduling problems. This hybrid approach takes advantage of the characteristics of both algorithms to address the inherent complexities and NP-hardness of scheduling jobs. The problem is represented by encoding job sequences and machine allocations as particles within the PSO frame job. Each particle represents a potential solution, with its position corresponding to a particular job schedule. The use of GA principles aids in the maintenance of population diversity, hence avoiding premature convergence—a major problem in regular PSO. The fitness function utilized in this study is intended to minimize the makespan, or the total time required to complete all scheduled jobs. The fitness function additionally includes penalties for violating job priority limits and machine availability, ensuring that the generated schedules are both feasible and optimal. The hybrid algorithm's computing performance was measured against a set of benchmark job shop scheduling tasks. The results showed significant improvements in convergence speed and solution quality over standalone PSO and GA algorithms. Specifically, the hybrid technique reduced makespan values and improved load balancing among machines. The study demonstrates the effectiveness of combining evolutionary strategies with swarm intelligence to solve complicated scheduling problems. Potluri et al. (2023) developed an improved job scheduling algorithm for cloud computing environments by using an optimized PSO approach. This study addresses the critical challenge of job scheduling by offering a dynamic adjustment of parameters using discrete positioning (DAPDP) within the PSO framework. The representation approach encodes job loads and cloud resources as particles in the PSO model. Each particle represents a potential solution, and its position corresponds to a certain task scheduling arrangement.

This encoding enables the flexible and dynamic assignment of cloud resources to various tasks. The fitness function used in this study is intended to reduce both makespan and scheduling time while considering the execution time, cost, infrastructure scalability, reliability, platform availability, throughput, and resource utilization. By optimizing these parameters, the fitness function ensures that planned tasks are not only executed effectively but also make the best use of available resources. The proposed DAPDP-based PSO

algorithm was evaluated with other job scheduling methods in cloud environments. The findings showed significant improvements in makespan, scheduling time, and execution time. Specifically, the DAPDP technique improved resource allocation, reduced execution time, and increased reliability. A study by Zarrouk et al. (2019) developed a two-level PSO algorithm to enhance the performance by subdividing the search space into regions with common properties, further dividing these into sub-regions with similar features, such as schedules with the same partial order. By using lower-bound criteria and optimized particle coding, the fitness function reduces complexity and maintains PSO self-adaptive learning by avoiding non-optimal regions. This design encourages a balance between exploration and exploitation. Computational results indicate that the two-level PSO consistently achieves superior makespan values, with a few exceptions, and demonstrates competitive results in both makespan and job load while significantly reducing CPU time.

The dynamic adjustment of parameters based on the gap between lower-bound and current *global-best* values further improves makespan performance and CPU time. Lastly, research by Guo & Lei (2021) presents a two-level Imperialist Competitive Algorithm (ICA) for energy-efficient flexible job shop scheduling (FJSS). The representation strategy subdivides the search space into regions with common characteristics, enhancing performance by focusing on similar areas. This hierarchical approach allows multi-level granularity, improving both exploration and exploitation. The two-phase process involves broad exploration followed by refined searches within promising regions, effectively managing FJSS complexity. The fitness function balances exploration and exploitation, using lower-bound criteria to reduce computational complexity and avoid non-optimal regions. It maintains PSO self-adaptive learning stability, ensuring an efficient search trajectory. Computational results demonstrate the two-level ICA's superiority over traditional algorithms like MOGA and NSGA-II, consistently achieving better makespan values with significantly lower CPU time. These results validate the two-level ICA's effectiveness, making it a promising approach for complex, energy-efficient FJSS problems. However, the demonstrated inappropriate objective function that takes into account the relationships between jobs J, task processing times *t*, and limitations imposed by available resources is a critical setback of this approach.

3.0 Methodology

3.1 Solution representation

Particle Swarm Optimization (PSO) is applied to solve the job shop scheduling problem. Each particle presents a potential solution to the problem, which is the optimal job schedule. The original dataset contains 72 jobs but has been reduced to 10 jobs in this paper. There is a total of 10 jobs with varying times required to complete them, starting with job 0 and ending with job 9.

3.2 Objective function

The goal for this job scheduling problem is to minimize the time needed to complete all of the jobs. The objective function is defined as the mathematical expression:

Minimize

$$\sum_{i=1}^{N} C_{i,} \tag{1}$$

Ci is defined as:

$$C_i = \sum_{j=1}^{i} t_{\pi(j)} \tag{2}$$

Where:
N: Number of jobs
π: job at position *i* in the schedule
tπ(j): Processing time of the job at position *i* in the schedule
Ci: Completion time of the job at position *i* in the schedule.

3.3 Parameter Selection

The number of particles is set at 30. This is done to obtain a feasible and optimal solution while maintaining an acceptable processing time. Moreover, the stopping condition used in this coding is the maximum number of iterations. The PSO algorithm will stop searching for solutions if the iteration has been done 100 times. Furthermore, the inertia weight is set at 1.0. This is to ensure there is a balance to the particle movement in the search space, especially in global and local exploration. Finally, the cognitive coefficient and social coefficient are both set at 1.5. By having a balanced cognitive coefficient and social coefficient, the algorithm can balance between exploration and exploitation, thus producing better results.

3.3 Parameter Selection

The number of particles is set at 30. This is done to obtain a feasible and optimal solution while maintaining an acceptable processing time. Moreover, the stopping condition used in this coding is the maximum number of iterations. The PSO algorithm will stop searching for solutions if the iteration has been done 100 times. Furthermore, the inertia weight is set at 1.0. This is to ensure there is a balance to the particle movement in the search space, especially in global and local exploration. Finally, the cognitive coefficient and social

Table 1. Solution Representation of the jobs

4	1	7	6	0	9	3	5	2	8

The solution representation example shows the schedule for the job sequentially. From this solution example, the schedule will start with job 4. Then, the scheduling continues with job 1. Then, it will be followed by job number 7 and so on. This sequence of jobs will continue until all jobs have been scheduled. The PSO process flow is shown in Figure 1.

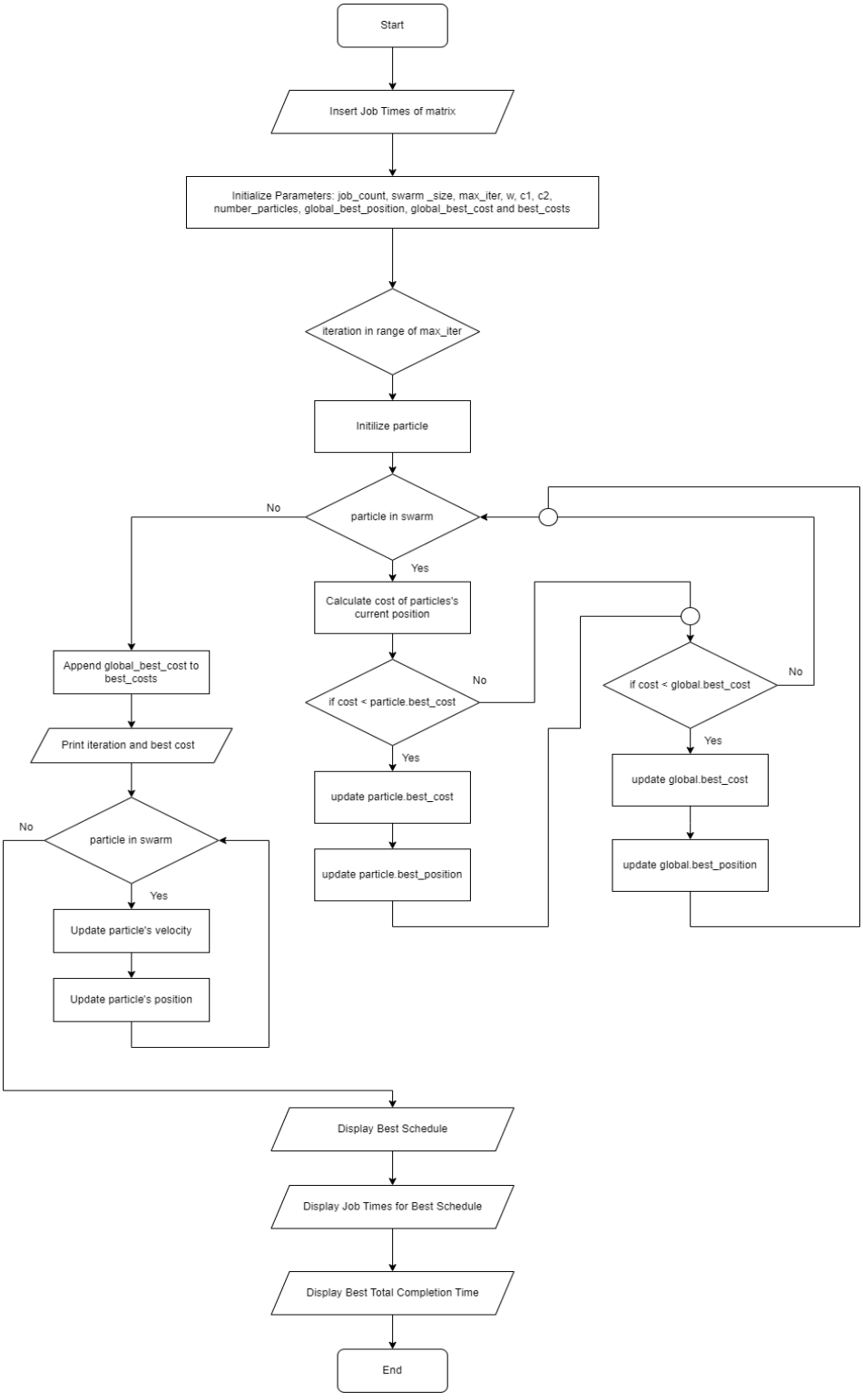

Figure 1. PSO Flow Diagram

coefficient are both set at 1.5. By having a balanced cognitive coefficient and social coefficient, the algorithm can balance between exploration and exploitation, thus producing better results. *3.3 Parameter Selection*

The number of particles is set at 30. This is done to obtain a feasible and optimal solution while maintaining an acceptable processing time. Moreover, the stopping condition used in this coding is the maximum number of iterations. The PSO algorithm will stop searching for solutions if the iteration has been done 100 times. Furthermore, the inertia weight is set at 1.0. This is to ensure there is a balance to the particle movement in the search space, especially in global and local exploration. Finally, the cognitive coefficient and social coefficient are both set at 1.5. By having a balanced cognitive coefficient and social coefficient, the algorithm can balance between exploration and exploitation, thus producing better results.

3.4 *Data Collection and Analysis*

The study utilized synthetic and benchmark datasets that simulate job scheduling scenarios in heterogeneous computing environments. These datasets include parameters such as job arrival time, execution time, resource demand, and priority level. The hybrid PSO algorithm was applied to these datasets to evaluate its effectiveness in optimizing job scheduling. The particle swarm component provided exploration capability, in contrast to a genetic algorithm that ensured exploitation and convergence. Performance metrics, including makespan, resource utilization, and throughput, were used to analyze results. The metrics were selected to comprehensively reflect both performance efficiency and resource management in job scheduling.

4.0 Findings and Discussion

The result of the baseline experiment is shown in Table 2. The results obtained demonstrate the successful application of Particle Swarm Optimization (PSO) to minimize the job completion time of scheduling tasks. Using a configuration of 30 particles over 100 iterations, with inertia weight w = 1 and cognitive and social coefficients c1 = c2 = 1.5, the algorithm converged to a stable solution by iteration 98, with a best cost of 112 maintained through iteration 100. This cost likely represents the total makespan, indicating the overall completion time for all scheduled jobs.

Table 2. Performance of a baseline experiment

Parameters	**Result**
1) No.of particles = 30	Iteration 98/100, Best Cost: 112
2) Max iteration = 100	Iteration 99/100, Best Cost: 112
3) Weight, w = 1	Iteration 100/100, Best Cost: 112
4) c1 = 1.5	Best Schedule: [1 0 4 7 3 6 9 8 5 2]
5) c2 = 1.5	

The optimized job sequence, [1 0 4 7 3 6 9 8 5 2], reflects the order in which tasks should be executed to achieve this minimum completion time. These results confirm that PSO effectively explored the scheduling space and identified an efficient schedule, thereby fulfilling the research objective of minimizing job completion time through intelligent optimization. The 10 different jobs against their respective completion times are shown in Figure 2 and the cost/iteration is presented in Figure 3, respectively.

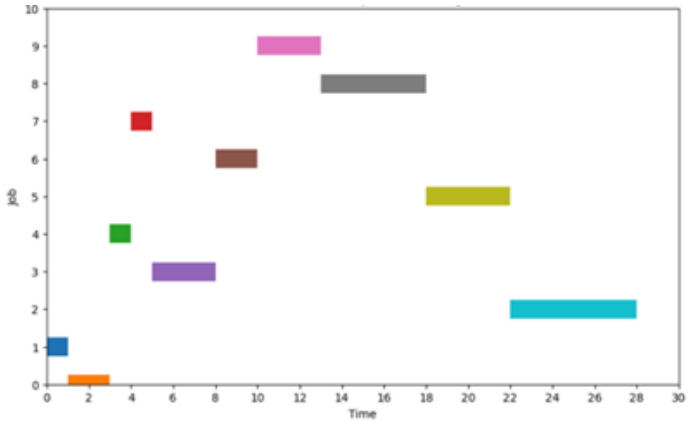

Figure 2. Job Schedule (Different colors represent 10 different jobs against their respective completion time)

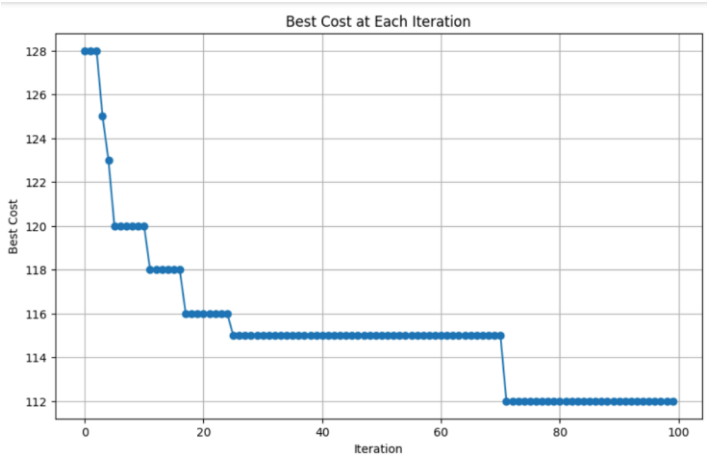

Figure 3. Cost per Iteration

4.1 Performance of Algorithm with Detailed Elaboration

The algorithm is tested with different parameters. The number of particles and the maximum number of iterations are changed while the inertia weight, cognitive coefficient, and social coefficient are constant. The comparative results from varying PSO configurations illustrate the impact of particle count and iteration limits on job scheduling performance, as shown in Table 3. With 20 particles and 200 iterations, the algorithm achieved the best result. A minimum job completion time (makespan) of 110, with convergence evident by iteration 198 and a final schedule of [7 1 0 4 6 3 9 5 2 8]. In contrast, increasing the particle count to 40 but reducing iterations to 50 yielded a slightly higher best cost of 113, despite showing improvement in the last few iterations, suggesting the potential for better outcomes with more iterations. The third configuration, using 50 particles but only 20 iterations, resulted in the highest best cost of 116, indicating insufficient convergence time despite a larger search population. These findings suggest that longer iteration counts may be more critical than simply increasing particle numbers for effective minimization of job completion time using PSO in scheduling tasks.

Table 3: Repeated Experiments

Parameters	**Result**
1) No.of particles = 20 2) Max iteration = 200 3) Weight, w = 1 4) c1 = 1.5 5) c2 = 1.5	Iteration 198/200, Best Cost: 110 Iteration 199/200, Best Cost: 110 Iteration 200/200, Best Cost: 110 Best Schedule: [7 1 0 4 6 3 9 5 2 8]
1) No.of particles = 40 2) Max iteration = 50 3) Weight, w = 1 4) c1 = 1.5 5) c2 = 1.5	Iteration 48/50, Best Cost: 115 Iteration 49/50, Best Cost: 113 Iteration 50/50, Best Cost: 113 Best Schedule: [7 1 0 4 6 5 3 8 9 2]
1) No.of particles = 50 2) Max iteration = 20 3) Weight, w = 1 4) c1 = 1.5 5) c2 = 1.5	Iteration 18/20, Best Cost: 116 Iteration 19/20, Best Cost: 116 Iteration 20/20, Best Cost: 116 Best Schedule: [4 1 7 5 0 6 9 8 3 2]

The results of the experiments conducted are summarized as shown in Table 4, with associated findings. According to the findings, achieving optimal performance in job scheduling using Particle Swarm Optimization (PSO) requires a careful balance between the number of particles and the number of iterations. The experiments demonstrate that just increasing the number of particles or iterations does not guarantee improved results. For example, while decreasing the iterations slightly improved the cost, increasing the number of particles with fewer iterations resulted in a higher cost, indicating inefficiency.

Table 4: Comparisons of Experiments

Parameters Used	**Results**	**Findings**
num_particles = 30, max_iter = 100, w = 1, c1 = 1.5, c2 = 1.5	Best Schedule: [1 0 4 7 3 6 9 8 5 2] Best Cost: 112	Initial baseline parameters shows convergence but can be optimized further.
num_particles = 20, max_iter = 200, w = 1, c1 = 1.5, c2 = 1.5	Best Schedule: [7 1 0 4 6 3 9 5 2 8] Best Cost: 110	Decreasing particles and Increasing iterations resulted in a slight improvement in cost.

num_particles = 40, max_iter = 50, w = 1, c1 = 1.5, c2 = 1.5	Best Schedule: [7 1 0 4 6 5 3 8 9 2] Best Cost: 113	Increasing particles but decreasing iterations led to a higher cost, suggesting overpopulation issues.
num_particles = 50, max_iter = 20, w = 1, c1 = 1.5, c2 = 1.5	Best Schedule: [4 1 7 5 0 6 9 8 3 2] Best Cost: 116	Further increase in particles and a decrease in iterations resulted in the highest cost, indicating inefficiency.

These results highlight the importance of fine-tuning parameters to balance exploration and exploitation effectively. Optimal parameter selection is essential for the efficient application of the PSO algorithm in job scheduling. The comparative experiments summarized in Table 4 highlight how variations in PSO parameters impact scheduling efficiency and job completion time. The baseline configuration, with 30 particles and 100 iterations, achieved a best cost of 112, demonstrating convergence but leaving room for further optimization. Reducing the number of particles to 20 while increasing iterations to 200 resulted in a lower best cost of 110, indicating improved performance through extended exploration time. Conversely, increasing the particle count to 40 but limiting iterations to 50 led to a slightly higher cost of 113, suggesting that a larger swarm without sufficient iterations may hinder convergence. The configuration with the highest particle count (50) and the fewest iterations (20) performed worst, with a cost of 116, highlighting inefficiency due to inadequate convergence time despite a broader search space. These findings underscore the importance of balancing swarm size and iteration depth to effectively harness PSO's capabilities. Optimal parameter tuning is crucial for enhancing job scheduling performance and minimizing total completion time.

This study offers a nuanced perspective on the application of Particle Swarm Optimization (PSO) in job scheduling, particularly when contrasted with prior research. While earlier studies have often emphasized the benefits of larger swarm sizes for enhanced global search capabilities, the present results suggest that merely increasing the number of particles does not necessarily lead to improved performance. For instance, configurations with a high particle count but limited iterations resulted in suboptimal outcomes, highlighting the importance of sufficient iteration depth for convergence. This observation aligns with the insights from Sarathambekai and Umamaheswari (2017), who emphasized the significance of balancing exploration and exploitation in PSO to avoid premature convergence and ensure efficient scheduling in multiprocessor systems. Furthermore, the study by Subramoney and Nyirenda (2020) on workflow scheduling in cloud-fog environments corroborates the notion that hybrid approaches, which often incorporate extended Iteration schemes can yield better optimization results compared to standard PSO configurations. In contrast, some earlier works, such as the study by Yen and Ivers (2009), have advocated for the use of multiple independent swarms to enhance search diversity and prevent local optima entrapment in job-shop scheduling problems. While this approach has its merits, the current findings underscore that without adequate iteration depth, the benefits of increased swarm size or multiple swarms may not be fully realized.

5.0 Conclusion and Recommendation

This paper demonstrates the effectiveness of PSO for solving job scheduling problems, particularly in minimizing task completion time. It enhances the field by validating PSO's rapid convergence capability and its advantage in avoiding local optima through global search strategies. The study highlights PSO's adaptability, simplicity, and low computational demands, making it highly practical for complex, resource-intensive scheduling environments like manufacturing and logistics. By experimenting with various parameter settings, the paper identifies optimal configurations such as fewer particles, more iterations, and balanced cognitive and social coefficients that improve solution quality.

PSO has proven to be a highly beneficial approach for job scheduling due to its global search ability, flexibility, and ease of implementation. The findings emphasize that fine-tuning the algorithm's parameters can significantly enhance performance, leading to better optimization outcomes without excessive computational overhead. However, more extensive comparisons could be made with other metaheuristic algorithms, such as genetic algorithms or ant colony optimization, to benchmark PSO's performance across diverse scheduling scenarios. Additionally, incorporating hybrid approaches that combine PSO with problem-specific heuristics or machine learning techniques could further improve efficiency and robustness. Applying the method to dynamic and real-time scheduling environments, where tasks and constraints change over time, would also extend its practical relevance.

Acknowledgments

The support of Universiti Teknologi MARA (UiTM) is duly appreciated for ensuring the stable infrastructure and platforms that always support our research efforts.

Paper Contribution to Related Field of Study

The contribution of this paper lies in demonstrating the effectiveness of PSO for job scheduling problems, specifically in minimizing task completion time. It adds to the field by validating PSO's ability to rapidly converge on optimal or near-optimal schedules in complex environments. This work underscores PSO's practical value in industries like manufacturing and logistics, where efficient scheduling

directly impacts resource utilization and cost reduction. By showcasing PSO's adaptability and performance, the paper provides a basis for the broader application of swarm intelligence techniques in operational optimization tasks.

References

Alireza Rezvanian, S. Mehdi Vahidipour, & Sadollah, A. (2023). An Overview of Ant Colony Optimization Algorithms for Dynamic Optimization Problems. *IntechOpen EBooks*. https://doi.org/10.5772/intechopen.111839.

Banu Çaliş, & Serol Bulkan. (2013). A research survey: a review of AI solution strategies for the job shop scheduling problem. *Journal of Intelligent Manufacturing*, *26*(5), 961–973. https://doi.org/10.1007/s10845-013-0837-8.

Guo, J., & Lei, D. (2021). An imperialist competitive algorithm for energy-efficient flexible job shop scheduling. In *2021 33rd Chinese Control and Decision Conference (CCDC)* (pp. 5145-5150). Kunming, China. https://doi.org/10.1109/CCDC52312.2021.9601714.

H. M. Wang, C. Liu, P. P. Li and J. Y. Shen (2022) "Cloud Task Scheduling Based on Improved Particle Swarm Optimization Algorithm," 2022 Asia Conference on Advanced Robotics, Automation, and Control Engineering (ARACE), Qingdao, China, pp. 24-29, doi: 10.1109/ARACE56528.2022.00013.

Kaliappan, S., Paranthaman, V., Kamal, M. R., Avv, S., & Muthukannan, M. (2024). A Novel approach of particle swarm and ANT colony Optimization for task scheduling in the Cloud. https://doi.org/10.1109/confluence60223.2024.10463398.

Liu, H., Wang, G., & Gui, W. (2019). A Multi-Objective Particle Swarm Optimization for Flexible Job-Shop Scheduling Problem with Transportation Time. IEEE Access, 7, 100327-100340. [doi: 10.1109/ACCESS.2019.2930222].

Potluri, S., Hamad, A. A., Godavarthi, D., & Basa, S. S. (2023). Enhanced task scheduling using optimized particle swarm optimization algorithm in a cloud computing environment. ICST Transactions on Scalable Information Systems. https://doi.org/10.4108/eetsis.4042.

Sarathambekai, M., & Umamaheswari, B. (2017). A particle swarm optimization algorithm for multiprocessor scheduling problem. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 231(7), https://doi.org/10.1177/1748301816665521..

Subramoney, S., & Nyirenda, C. N. (2020). Workflow Scheduling in a Cloud–Fog Environment Using a Hybrid Metaheuristic, https://arxiv.org/abs/2012.00176.

Zarrouk, R., Bennour, I. E., & Jemai, A. (2019). Toward a two-level PSO for the FJS problem. In 2019 IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI) (pp. 75-82). Herlany, Slovakia. https://doi.org/10.1109/SAMI.2019.8782738.

Zhang, Q., Zhang, B., & Liang, W. (2022). Research on flexible job shop scheduling problems based on improved discrete particle swarm optimization. International Conference on Advanced Manufacturing Technology and Manufacturing Systems (ICAMTMS 2022). https://doi.org/10.1117/12.2645548.

Yen, G. G., & Ivers, J. (2009). Multi-swarm optimization for dynamic job shop scheduling. Journal of Intelligent Manufacturing, 20(6), 701–713. https://doi.org/10.1108/17563780910939237.

Zhang, S., Li, X., Zhang, B., & Wang, S. (2020). Multi-objective optimization in flexible assembly job shop scheduling using a distributed ant colony system. European Journal of Operational Research, 283(2), 441–460. https://doi.org/10.1016/j.ejor.2019.11.016.