Tuijin Jishu/Journal of Propulsion Technology
ISSN:1001-4055
Vol. 44No. 4 (2023)

A Comparative Study of Evolutionary and Swarm Intelligence Algorithms for Job Scheduling on Identical Parallel Machines

[1]Rashmi Benni, [2]Shashikumar Totad, [3]Karibasappa K G, [4]Sachin Karadgi

School of Computer Science and Department of Automation and Robotics

KLE Technological University, Hubballi - 580031, Karnataka, India

***Abstract*—** In parallel computing systems, job scheduling plays a crucial role in enhancing system efficiency and minimizing the makespan. In recent years, evolutionary and swarm intelligence algorithms have gained prominence as effective approaches for solving combinatorial optimization problems. In the present work, we have considered genetic algorithm (GA) for evolutionary algorithms and particle swarm optimization (PSO) for swarm intelligence algorithms. Evolutionary algorithms (EA) and swarm intelligence algorithms (SIA) have shown promising results in solving job scheduling challenges. In this study, we collate the performance of EA and SIA approaches for job scheduling on parallel machines. We use different benchmark instances to evaluate the algorithms' makespan and computational time performance. The results show that SIA algorithms outperform EA algorithms regarding makespan and computational time for all benchmark instances. Furthermore, the study provides insights into the strengths and weaknesses of EA and SIA algorithms for job scheduling on parallel machines. Our findings provide useful insights for researchers and practitioners interested in applying optimization techniques to solve job scheduling problems on parallel machines.

***Index Terms*—** *evolutionary algorithms (EA), genetic algorithm (GA), job scheduling, meta-heuristics, optimization, parallel machines, particle swarm optimization (PSO), swarm intelligence algorithms (SIA).*

Introduction

In the era of intelligent computing systems efficient job scheduling on parallel machines is crucial to maximize the system throughput and reduce job completion times. Tasks are assigned to machines during job scheduling in order to optimize performance indicators like makespan and resource utilization. Manufacturing and production industries, High-Performance Computing (HPC), Cloud Computing, Energy Management, Telecommunications, and more are a few of the fields for which job scheduling is used [13]. Researchers have investigated several new computational intelligence techniques, such as swarm intelligence and evolutionary algorithms, as viable solutions for efficient work scheduling to meet this challenge [3]. Natural selection and genetics are the foundations of evolutionary algorithms. They employ genetic operators like selection, crossover, and mutation to iteratively develop a population of candidate solutions. Based on predetermined objectives, the fitness of each solution is assessed, and the more fit individuals are given preference for reproduction, giving rise to future generations of possibly better solutions. Evolutionary algorithms can traverse the search space and converge on optimal or nearly optimal solutions due to this iterative process. Swarm intelligence, on the other hand, draws its inspiration from the group behavior of social insect colonies, where individual agents work together and coordinate to solve challenging issues [5]. Particle swarm optimization (PSO) and other swarm intelligence algorithms model the behavior of these swarms to discover the best answers. Particles in PSO move around the solution space by modifying their positions

in accordance with their personal and social knowledge, which directs them toward better solutions [8]. Numerous optimization issues, such as work scheduling on identical parallel processors, have been tackled effectively using both evolutionary and swarm intelligence techniques. These methods offer benefits like flexibility, adaptability, and the capacity to manage complicated optimization environments. However, depending on the features of the problem and the algorithm setups, their performance may vary. This comparative study compares and contrasts the efficiency of swarm intelligence and evolutionary algorithms for work scheduling on identical parallel machines. We'll look into how well they perform in terms of makespan, resource use, and convergence speed. We will also investigate how various algorithmic parameters and problem scenarios affect the performance of the algorithms. Researchers and practitioners can choose the best algorithm for their unique scheduling scenarios by taking into consideration each of their respective features. In summary, this research promotes improved system performance and resource utilization by advancing effective work scheduling methods in parallel computing systems. The structure of this article is as follows: The preliminary research is provided in Section II. The proposed work and problem formulation employed are presented in Section III. The experimental setup, including the algorithms, benchmark instances, and performance measurements, is described in Section IV. The findings of the comparative study and results are presented in Section V along with a discussion of them. Section VI concludes by summarizing the results, and future scope for further research in this area.

Related Work

The employment of evolutionary algorithms (EAs) and swarm intelligence algorithms (SIAs) for job scheduling on identical parallel machines has been an area of various studies in recent years [1]. The summary of the important results and contributions of earlier studies are highlighted in this section. Solmaz Abdi carried out one of the early studies in this field and suggested a genetic algorithm (GA), PSO, and Modified PSO for job scheduling on parallel machines [6]. Their strategy centered on reducing the makespan, or the overall amount of time needed to accomplish all jobs. To produce novel solutions, the GA used mutation and crossover operators in addition to permutation encoding. Swarm intelligence was used by PSO and Modified PSO to determine the optimal solutions [9]. The algorithms were able to locate almost ideal solutions for a variety of problem scenarios, according to experimental results. To explore scheduling theory and techniques Michael L. Pinedo proposed various kinds of scheduling, its research challenges and applications [2]. It is a comprehensive article that provides a thorough introduction to the theory, algorithms, and systems related to scheduling. It covers a wide range of scheduling problems and techniques, offering valuable insights into the fundamental principles and practical applications of scheduling in various domains. The study by Wei-ChangYeh addresses the issues of scheduling tasks on uniform parallel machines with a constraint on resource consumption [4]. The authors propose a mathematical model and develop a heuristic algorithm to find efficient schedules while satisfying the resource consumption constraint, demonstrating the applicability of their approach through numerical experiments [7]. The efficiency of these metaheuristic algorithms has been demonstrated in earlier studies for job scheduling on parallel machines. The investigations have shown that they can reduce makespan, find solutions that are close to optimal, and increase computing efficiency [10]. The performance of these algorithms in the context of job scheduling on identical parallel machines may be further improved with additional research into novel algorithmic variations, tuning of hyper-parameters, and problem-specific adaptations.

Proposed Work

The research study proposed in this article examines the Job scheduling on identical parallel machines using a comparative evaluation of various meta-heuristic algorithms, such as particle swarm optimization (PSO) and genetic algorithm (GA) with the objective to design a fitness function that minimizes the makespan and considers the satisfaction of various precedence constraints and machine eligibility restrictions. Precedence constraint is the relationship between two jobs and machine eligibility restrictions is the job running capabilities of a machine. These constraints to which the problem is subjected depict the real world scenarios of job scheduling. To improve the solution fine tuning techniques, testing and validating a set of benchmark instances from the literature are carried out. Additionally, conclusions are derived from the findings of both algorithms after conducting a number of experiments in order to comprehend their abilities and weaknesses.

A. System Architecture

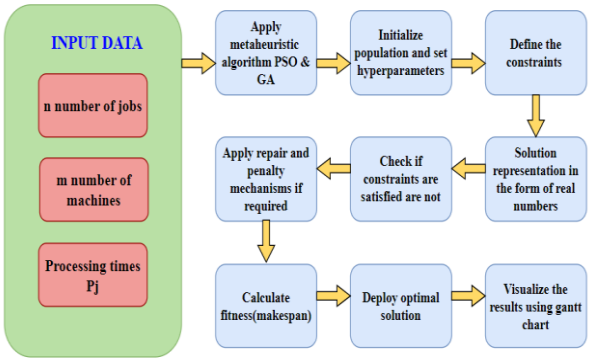


Fig 1: System Architecture

The system architecture illustrates how the scheduling of n number of jobs on m identical parallel machines can be achieved by using meta-heuristics like evolutionary and swarm intelligence techniques. The system receives as input the number of jobs (n), the number of machines (m), and the processing times (Pj) for each job. Then, we employ the GA and PSO algorithms. We first set the hyper parameters, initialize the random population, and provide the constraints for jobs and machines. Then, the heredity chromosomes in evolutionary algorithms and the particles in swarms are represented as real numbers, which aids in determining the sequence and allocation of the jobs that need to be scheduled on machines. If the constraints are not satisfied repair and penalty techniques are used to fine-tune the solution. The final result is calculated in the form of fitness i.e. makespan. Finally, after testing for different scenarios the optimal solution is deployed and the result is visualized by a Gantt chart.

B. Mathematical Formulation of the Problem

A scheduling problem is represented as α | β | γ, where α is the machine environment, β is the characteristics of the job and γ is the performance indicator. The mathematical understanding of the job scheduling problem is that n jobs need to be scheduled on m machines in parallel with aim to minimize the total makespan.

$$Minimize\ Cmax \quad (1)$$

$$\sum_{i \in M} xij = 1,\ \forall j \in N' \quad (2)$$

$$\delta jk + \delta kj = 1,\ \forall j, k \in N', j != k \quad (3)$$

$$Ck \geq Cj + pk,\ \forall j, k \in A, j != k \quad (4)$$

$$H(1 - xij) + H(1 - xik) + Cj \geq Ck + pj,\ \forall i \in M, \forall j, k \in A'', j != k \quad (5)$$

$$H(1 - xij) + H(1 - xik) \geq 1,\ \forall i \in M, \forall j, k \in A''', j != k \quad (6)$$

$$\sum_{i \in Mj} xij = 1, \ \forall j \in N' \quad (7)$$

$$Minimize\ Cmax + penalty(\phi) \quad (8)$$

Where,

i	Machine index
j, k	Job indices
Cj	Completion time of job j
Cmax	Makespan
m	Total number of machines
M	Set of machines, M = {1, 2 . . . m}
n	Total number of jobs
N′	Set of jobs to be scheduled, N′ = {1, 2 . . . n}
Mj	Machine eligibility restrictions for job j
pj	Processing time of job j
A′	Set of precedence constraints with jobs j and k to be assigned on the same machine
A′	Set of precedence constraints with jobs j and k to be assigned on any machine
A″	Set of precedence constraints where job k cannot be assigned after the completion of job j on machine i, but can be assigned before the start of job j on machine i, or jobs j and k can be assigned on different machines
A‴	Set of jobs j and k that need to be assigned on different machines
H	Large positive number, say 106
xij	1, if job j is assigned on machine i 0, otherwise
δjk	1, if job j precedes job k, directly or indirectly 0, otherwise
ϕ	Penalty that will be added to makespan if any of the mentioned constraints fails

II. The Experimental Setup and Algorithms

The experimental setup includes algorithms, different steps and their mathematical representation of the solution.

A. Evolutionary Algorithm (GA)

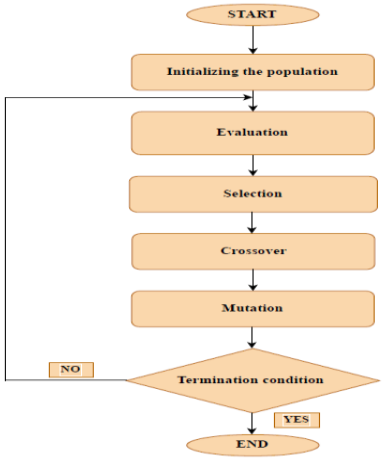


Fig 2: Genetic algorithm

Job scheduling task on identical parallel machines using swarm and evolutionary algorithms requires the solution to be represented in mathematical format so that the algorithms can decide the machine allocation and sequence of the jobs for execution. To increase the population variability, an initial population of N solutions is generated randomly. In this case, randomization is achieved by picking a randomly chosen job from the list of jobs and allocating it to a randomly chosen machine. A real value with an integer component and a fraction part is filled into each gene or particle in the representation of a solution. The integer portion represents a machine that will be selected at random and range from 0 to m - 1. Furthermore, a random number between 0 and 1 is chosen for the fraction component, up to two or more decimals. It is possible to choose a population size N that is sufficient to find the ideal solution more quickly with proper allocation and sequencing of Jobs. Genetics and the process of natural selection serve as the basis for GA. Each individual in the population serves as a potential solution (a job schedule) in the context of job scheduling and is endowed with a set of genes that encode the traits of that solution. Each individual in the population undergoes genetic operations like crossover and mutation to produce new offspring as the population changes over generations. Every generation measures the makespan of the corresponding job schedule to assess each individual's fitness. Individuals with higher fitness often have shorter makespans, and are more likely to be chosen for reproduction. The population selection and candidate solutions are improved over time as the algorithm iterates through multiple generations. Algorithm terminates when a workable solution is identified, and is met or reached a maximum number of iterations. When the GA terminates, it delivers the best solution found, it is the job schedule with the shortest makespan among all individuals in the final population.

B. Swarm Intelligence Algorithm (PSO)

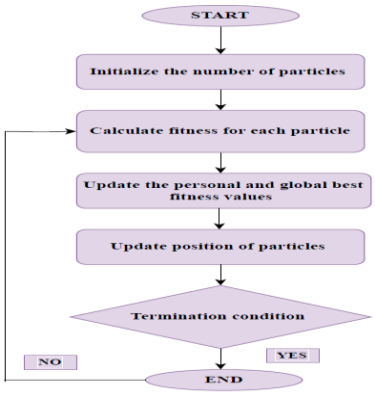
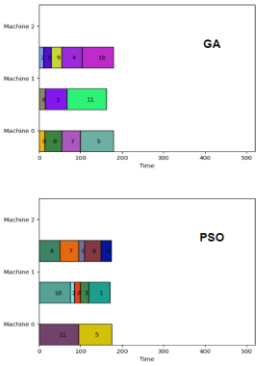


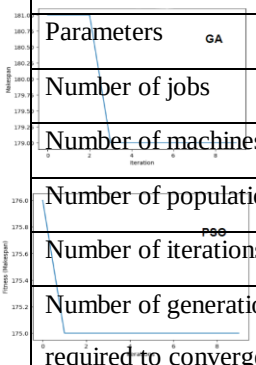
Fig 3: Particle Swarm Optimization algorithm

PSO algorithm mimics the social behavior of the swarms and the particles have memory of the best solution encountered. In the context of job scheduling each particle can be represented as a potential solution (a job schedule) and has a position and velocity in the solution space. The fitness of each particle is calculated by measuring the makespan of the corresponding job schedule [11]. The velocity of each particle is adjusted based on its previous velocity, its personal best solution, and the global best solution. The position of each particle is updated according to its new velocity. The algorithm Iterates through the evaluation steps until termination condition is reached. Once the algorithm terminates, it returns the best solution found, which corresponds to the job schedule with the minimum makespan.

Results And Discussion

The trials are parted into sections for detailed analysis. In the initial phase, we kept the number of jobs and machines constant while adjusting the population and iterations. To examine how other parameters could impact outcomes, we increased the number of jobs. In addition to that we adjusted the number of machines in the second simulation while maintaining the population and iteration counts. To obtain an optimal schedule with the fulfillment of precedence constraints and machine eligibility restrictions we have considered some of the important benchmark metrics for both the algorithms and compared their results.





Experiment I		
Parameters	GA	PSO
Number of jobs	12	12
Number of machines	3	3
Number of population	50	50
Number of iterations	10	10
Number of generations required to converge	1	3
CPU time	0.04	0.02
Fitness	179	175

Fig 4: Experiment I GA vs PSO

Experiment II		
Parameters	GA	PSO
Number of jobs	12	12
Number of machines	3	3
Number of population	100	100
Number of iterations	20	20
Number of generations required to converge	3	3
CPU time	0.16	0.08
Fitness	176	174

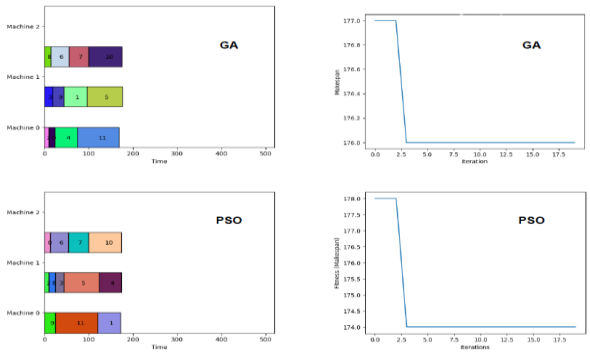


Fig 5: Experiment II GA vs PSO

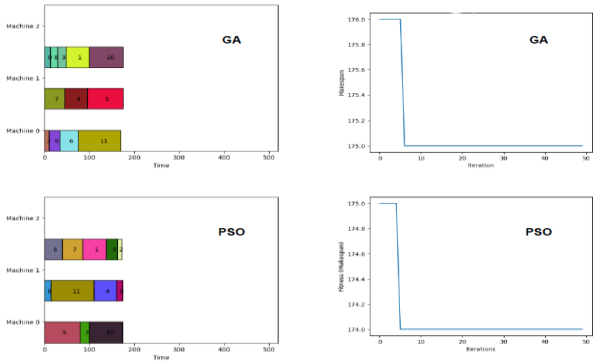


Fig 6: Experiment III GA vs PSO

Experiment III		
Parameters	GA	PSO
Number of jobs	12	12
Number of machines	3	3
Number of population	150	150
Number of iterations	50	50
Number of generations required to converge	6	5
CPU time	0.61	0.28
Fitness	175	174

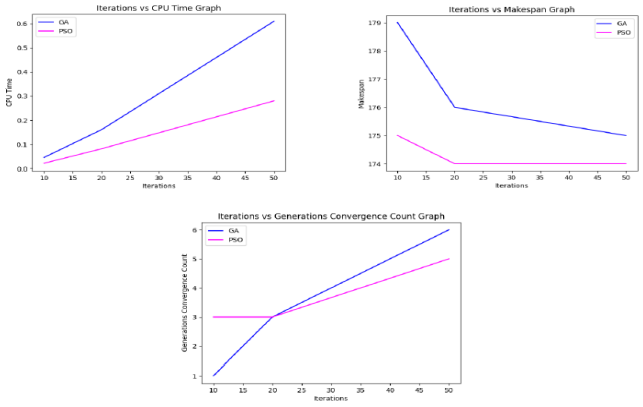


Fig 7: CPU time, Convergent generations count and Fitness vs Iterations

The constraints considered in the experiments depict the real time scenarios in job scheduling, Equation (1) discusses minimizing the makespan, while equation (2) ensures sure that a job j is only ever allocated to machine i once. The constraints (3) and (4) govern the order of the jobs they make sure that jobs j and k are assigned to the same machine and follow the precedence constraint j k, but not necessarily that job j comes just after job k. Either

or function is supported by the constants (5) and (6). Equation (7) depicts the machine eligibility constraint. Equation (8) describes the calculation of the makespan, if any of the conditions are not met, the algorithm first tries to repair it; otherwise, it adds the penalty to the total makespan. In comparison to GA, PSO tends to be more sensitive to its hyperparameters. PSO necessitates fine-tuning parameters like the inertia weight (w), cognitive (c1), and social (c2) coefficients. The population size, crossover, and mutation rates, as well as the selection process, are some of the hyperparameters unique to GA that need to be adjusted. Following a series of tests and references to earlier literature, the PSO parameters are set to 0.5 (w), 2 (c1), and 2 (c2). The GA's mutation rate, crossover rate, and selection rate are each set to 0.1, 0.5, and 3, respectively. The effectiveness of the algorithm's solution and its decision-making capacity are significantly influenced by the tuning of these parameters.

Experiment IV		
Parameters	GA	PSO
Number of jobs	15	15
Number of machines	3	3
Number of population	100	100
Number of iterations	50	50
Number of generations required to converge	8	4
CPU time	0.42	0.19
Fitness	205	204

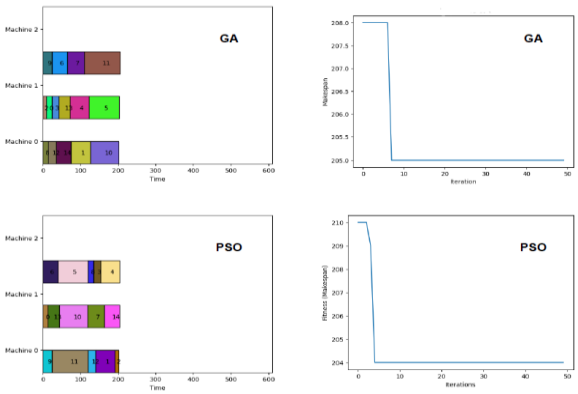


Fig 8: Experiment IV GA vs PSO

Experiment V		
Parameters	GA	PSO
Number of jobs	15	15
Number of machines	4	4
Number of population	100	100
Number of iterations	50	50
Number of generations required to converge	2	5
CPU time	0.41	0.20
Fitness	160	155

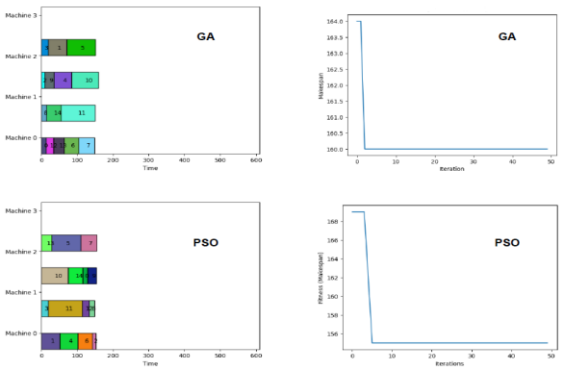


Fig 9: Experiment V GA vs PSO

Experiment VI		
Parameters	GA	PSO
Number of jobs	15	15
Number of machines	5	5
Number of population	100	100
Number of iterations	50	50
Number of generations required to converge	9	4
CPU time	0.46	0.17
Fitness	135	129

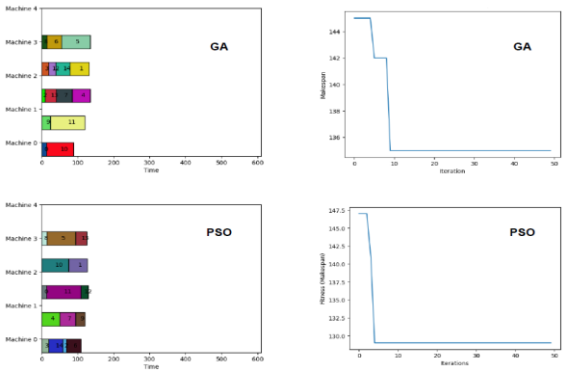


Fig 10: Experiment VI GA vs PSO

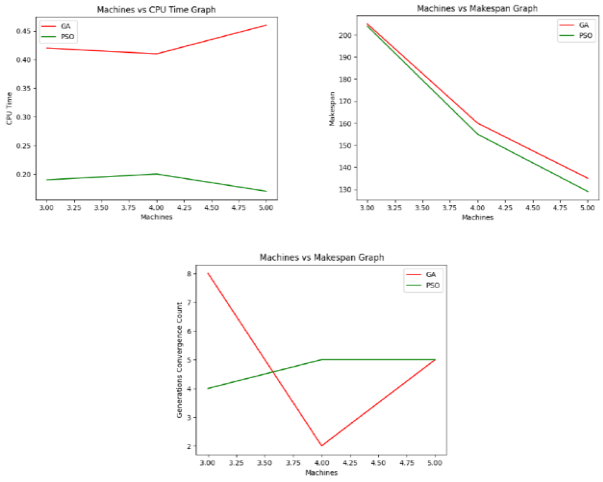


Fig 11: CPU time, Convergent generations count and Fitness vs Machines

The first phase of the tests involves changing the population to 50, 100, and 150 while maintaining the same number of jobs and machines. Adjustments are made to the number of iterations to 10, 20, and 50. The following testing phase has increased the number of jobs to 15, altered the number of machines to 3, 4, and 5, and left the population (100) and iterations (50) unchanged. Parameter modifications have been made to better assess the algorithms' effectiveness in exploring and using search space, as well as the impact of these parameters on the quality of the solution provided. This strategy helps to obtain the optimal makespan in less time possible. While analyzing the results, minimum makespan in less amount of time was one of the primary benchmark instances. The importance of minimal makespan in the job scheduling can be attributed to a number of factors. It serves as a gauge of the schedule's effectiveness first. A shorter makespan schedule is more effective than a larger makespan schedule. Second, a minimal makespan can be employed to increase resource utilization. Resources can be used more effectively by planning jobs in a way that minimizes makespan. Third, deadlines can be met by using minimal makespan. A scheduler might need to discover a technique to reduce makespan if a deadline is drawing near so that they can meet it. Overall, the minimum makespan should be taken into account while planning jobs. Schedulers can increase the effectiveness of their schedules, make better use of their resources, and fulfill deadlines by minimizing makespan. Swarm based algorithm particle swarm optimization outperformed the evolutionary genetic algorithm in terms of CPU time, fitness (makespan), and the number of generations needed to converge after a detailed analysis of a set of trials carried out in two phases. The fact that swarm-based algorithms convey the best solution to the subsequent generation is one of the main factors contributing to their superior performance in optimization issues. However, in evolutionary algorithms, the memory is constrained to the current population. Figures 10 and 11 and the computational tables in this article unambiguously demonstrate how the algorithms PSO and GA performed when they were subjected to various constraints and benchmark conditions. PSO's faster convergence

makes it more effective than GA for solving parallel problems. This is significant because it might be challenging to handle job scheduling problems because they are often NP-hard. PSO can solve these issues more quickly than GA, which can result in a reduction in time and resources. PSO is also more resistant to noise and outliers than GA. The reason for this is that while GA has a local search approach, PSO employs a global search strategy. Even if the problem data is noisy or contains outliers, a global search technique is more likely to uncover credible solutions.

Conclusion And Future Scope

According to the outcomes of the research, Particle Swarm Optimization (PSO) outperformed Genetic Algorithm (GA) in the comparative study on task scheduling for identical parallel machines using evolutionary and swarm intelligence techniques. PSO showed greater performance in terms of job schedule optimization, CPU time, parameter sensitivity, leading to increased effectiveness and reduced makespan. Better solutions were produced as a result of the effective exploration and exploitation of the search space by PSO's swarm intelligence technique. These results demonstrate the potential of PSO as a robust and efficient technique for job scheduling on identical parallel machines, highlighting its superiority over GA in this specific application. The intent of this research's future work is to explore the hybridization of evolutionary and swarm intelligence based algorithms to create more reliable and effective employment scheduling techniques and deploy for real time data. Additionally, testing these algorithms' scalability and practical applicability on larger problem instances, scheduling with uncertain parameters, and realistic circumstances will aid in their validation.

References

[1] Lalwani, S., Sharma, H., Satapathy, S.C. et al. A Survey on Parallel Particle Swarm Optimization Algorithms. Arab J Sci Eng 44, 2899–2923 (2019).

[2] Michael L. Pinedo, Scheduling Theory, algorithms and systems Natural computing series springer (2nd edition, 2015)

[3] Anthony Brabazon, Michael O'Neill & Seán McGarraghy, Natural Computing Algorithms Natural computing series springer (5th edition ,2015)

[4] Wei-Chang Yeh, Mei-Chi Chuang, Wen-Chiung Lee,Uniform parallel machine scheduling with resource consumption constraint, Applied Mathematical Modelling,Volume 39,Issue 8,2015.

[5] Alam, Mahamad. (2016). Particle Swarm Optimization: Algorithm and its Codes in MATLAB.

[6] Abdi, Solmaz & Motamedi, Seyed & Sharifian, Saeed. (2014). Task scheduling using modified PSO algorithm in cloud computing environment. Int Conf Mach Learn Electr Mech Eng. 37-41.

[7] Al-maamari, Ali & Omara, Fatma. (2015). Task Scheduling Using PSO Algorithm in Cloud Computing Environments. International Journal of Grid and Distributed Computing. 8. 245-256. 10.14257/ijgdc.2015.8.5.24.

[8] Ibrahim Alharkan, Mustafa Saleh, Mageed A. Ghaleb, Husam Kaid, Abdulsalam Farhan, A. Almarfadi, Tabu search and particle swarm optimization algorithms for two identical parallel machines scheduling problem with a single server, Journal of King Saud University - Engineering Sciences, Volume 32, Issue 5,2020.

[9] Z. Xilin, T. Yuejin and Y. Zhiwei, "Resource allocation optimization of equipment development task based on MOPSO algorithm," in Journal of Systems Engineering and Electronics, vol. 30, no. 6, pp. 1132-1143, Dec. 2019, doi: 10.21629/JSEE.2019.06.09.

[10] R. Wu, Y. Li, S. Guo and X. Li, "An Efficient Meta-Heuristic for Multi-Objective Flexible Job Shop Inverse Scheduling Problem," in IEEE Access, vol. 6, pp. 59515-59527, 2018, doi: 10.1109/ACCESS.2018.2875176.

[11] Kamble, S. & Mane, Sandeep & Umbarkar, AJ. (2015). Hybrid Multi-Objective Particle Swarm Optimization for Flexible Job Shop Scheduling Problem. International Journal of Intelligent Systems and Applications. 7. 54-61. 10.5815/ijisa.2015.04.08.

[12] Shahnazari-Shahrezaei, Parisa & Zabihi, Sina & Kia, Reza. (2017). Solving a Multi-Objective Mathematical Model for a Multi-Skilled Project Scheduling Problem by Particle Swarm Optimization and Differential Evolution Algorithms. Industrial Engineering & Management Systems. 16. 288-306. 10.7232/iems.2017.16.3.288.

[13] Seema A. Alsaidy, Amenah D. Abbood, Mouayad A. Sahib, Heuristic initialization of PSO task scheduling algorithm in cloud computing, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 6, Part A, 2022, https://doi.org/10.1016/j.jksuci.2020.11.002.