Contents lists available at GrowingScience

Journal of Project Management

homepage: www.GrowingScience.com

Multi-objective flexible job-shop scheduling in hospital using discrete particle swarm optimization algorithm with adaptive inertia weight (DPSO-AIW)

Md. Limonur Rahman Lingkon[a*] and Adri Dash[a]

[a]Department of Industrial & Production Engineering, Rajshahi University of Engineering & Technology, Rajshahi-6204, Bangladesh

C H R O N I C L E

Article history:
Received: April 1, 2024
Received in revised format: June 28, 2024
Accepted: July 28, 2024
Available online: July 28, 2024

Keywords:
Discrete Particle Swarm Optimization (DPSO)
Adaptive Inertia Weight (AIW)
Flexible Job Shop Problem (FJSP)
Global Selection based on Operation (GSO)
Machine Assignment (MA)
Pareto Optimality

A B S T R A C T

A multi-objective Flexible Job-shop Scheduling technique for hospitals is proposed using DPSO-AIW i.e. discrete particle swarm optimization with adaptive inertia weight method. The approach encodes the layer of the chromosomes using an operation sequence (OS) and machine assignment (MA) which is a two-layer coding structure. Global selection based on the operation (GSO) of MA and random selection of OS are coupled in the initial population. Rapid non-dominated sorting yields fronts of non-domination, which are necessary for getting the Pareto optimum solution. The diversity of the population is increased during the evolution process by adaptive adjustment of the variation of the weight of inertia, expressed by ω. Then, the Pareto optimal solution found during the process is kept in the Pareto optimal solution set (POS). The discrete particle swarm optimization algorithm is utilized to solve the values of the next generation chromosomes in the discrete domain directly. Lastly, comparisons with certain current techniques and numerical simulation based on two sets of international standard examples are performed, which are already established. The findings from the comparison show that the suggested DPSO-AIW is practical, effective, and more feasible for solving the problem related to the Multi-objective Flexible Job-shop Scheduling Problem.

1. Introduction

An expansion of the Job-shop Scheduling Problem normally expressed by JSP is the Flexible Job-shop Scheduling Problem which is called FJSP. When numerous operations of distinct tasks may be handled on separate machines by FJSP, processing flexibility is improved more by the actual company. This modifies the equipment's uniqueness and makes it possible to choose the processing machine depending on the load circumstances of resources, such as machines. Because of this, theoretical research on FJSP is essential to tackling the enterprise's combination optimization type actual workshop problem. The production schedule optimization problem must be taken into consideration for factors such as customer satisfaction, processing time, and production cost in the real production process. It is challenging to capture the actual state of the scheduling workshop in a single scheduling objective. Zhang looked into the two-archive multi-objective artificial bee colony approach known as TMABC-FS. Two new operators are applied to enhance the search performance of different bee species, and two archives are proposed to provide a set of non-dominated feature subsets with good distribution and convergence (Zhang et al., 2019). This work develops a particle swarm optimization that is discrete with adaptive inertia weight (DPSO-AIW) to solve FJSP based on the body of current research. The FJSP model is established using makespan, bottleneck machine workload, and overall machine workload. The chromosomes of the next generation are solved in the discrete domain, which is directly involved through the evolution process, which employs discrete particle swarm optimization. The algorithm encodes chromosomes using a two-layer structure. The genetic algorithm utilized for crossover operation is used for the location update, and adaptive adjustments are made to the inertia weight ω value to improve population variety. This study establishes the model related to FJSP with the maximum time required for completion of the machines, the workload of the bottleneck machine, and the workload of the overall machine. The hybrid approach combines global selection by operation (GSO) with OS random selection, which is used to construct the initial population. The process by which an

* Corresponding author.
E-mail address: limonurrahman16@gmail.com (Md. L. R. Lingkon)

ISSN 2371-8374 (Online) - ISSN 2371-8366 (Print)
© 2024 by the authors; licensee Growing Science, Canada.
doi: 10.5267/j.jpm.2024.7.007

operation will be chosen by the machines in its optional machine set is referred to as the GSO. All of the scheduling produced in this manner are workable. The machine containing the lowest global workload will be chosen for processing in the optional machine set when the GSO is chosen. In addition to guaranteeing the viability of the created scheduling, it minimizes the machine's burden, expedites the optimization process, and assures the original solution's quality. The machine's effort is minimized, the optimization process is expedited, and the original solution's quality is ensured. The OS random selection keeps the best solution from being lost while also increasing population variety. The quality found for the solution and the time it takes to find the best answer is significantly increased by using this strategy to create the initial population. Discrete particle swarm optimization is used in the evolutionary process for solving the next generation of chromosomes in the discrete domain. The genetic algorithm's mutation and crossover procedures are used in the location update. The OS uses linear order crossover (LOX) and precedence operation crossover (POX) throughout the process required for the crossover, whereas the MA uses a single-point crossover technique, which is in an improved state. This crossover approach guarantees that the produced solution is always feasible and that the search capacity is enhanced quickly. The population's variety is increased via the mutation process of the genetic algorithm. To avoid the algorithm entering an immature convergence state, its local search capability is improved. To increase population variety, choose an adaptive inertia weight modification. The particle's global optimum value of the population and the exponential function of the current value are used to update the inertia weight. Using the exponential function of the current value and the global optimum of the particles in the population, the inertia weight ω is adaptively changed. Each candidate solution's fronts are found using the fast non-dominated sorting approach, and the chosen solution is then put in the Pareto optimum solution set in the order of its fronts. The dataset used for the analysis was taken from a hospital in Bangladesh. The structure of the paper is given as follows: the introduction of formulation of the FJSP; basic particle swarm optimization is covered; the DPSO-AIW algorithm is implemented in detail, the inclusion of the encoding and decoding part; the population is initialized; the PSO location updating method; the adaptive inertia weight; and finally, the optimal solution set (POS) is constructed, along with the DPSO-AIW algorithm flow. Findings of the computer experiments were utilized for the DPSO-AIW method and the comparisons with alternative two different algorithms for the data were then displayed in a study of the worst computational complexity of the applied algorithm in this study. A sensitivity analysis of the parameters was carried out lastly to represent the findings in a better way.

2. Literature Review

Many academics these days have also turned their focus to multi-objective FJSP solutions. Huang & Yang presented a hybrid genetic algorithm-based multi-objective FJSP problem (X. Huang & Yang, 2019). Dai suggested a better NSGA for the FJSP. The study presented the elite retention method and the adaptive mutation operator. The simulation experiment demonstrates that by splitting the entire population into three halves, the non-dominated sorting approach can find the Pareto optimum solution quickly and accurately (Dai, 2021). Piroozfard et al. (2018) suggested a more effective multi-objective evolutionary method to solve the recently expanded dual-objective issue. Institute of Electrical and Electronics Engineers suggested a hybrid local search (PLS) method based on Pareto that may be used for making the solution of the multi-objective FJSP. Continuous optimization was the initial challenge that the PSO method was designed to answer. Nonetheless, many real-world engineering application challenges are discrete. Thus, by modifying the PSO algorithm's fundamental concept, a discrete scholar was created. Hui (2012) presented a hybrid PSO technique for the three-target FJSP problem that uses Pareto archives set. Huang created a system for automatic scheduling decoding and extended process coding. The study presents the creation of particle swarm optimization which is a multi-objective method for flexible production scheduling, considering the maximum and minimum number of particles, the rate of convergence, and any associated boundary conditions (Huang et al., 2016a). Zhang suggested using a particle swarm optimization technique of hybrid strategy to investigate the Pareto-dominance-based multi-objective FJSP. based on the Variable Neighbourhood Search (VNS) algorithm's and PSO's complementary strengths (Zhang et al., 2017). A study presented multiple hybrid algorithms. Such as PSO and VNS Cooperative algorithm (PVC), VNS in Turn (PVT) method, and PSO (Zhang & Gu, 2023) . To optimize the flow shop scheduling issue of the hybrid strategy, Lai created a no-wait method of grading for the constraint of no wait between two of the sequential operations related to the task (Lai et al., 2021). Huang solved the FJSP problem successfully by combining the variable neighborhood search approach with the multi-objective particle swarm optimization (S. Huang et al., 2016). It was Kacem who initially suggested the localization (AL) method (Kacem et al., 2002). Pezzella set up the order of operations using three scheduling rules. Global selection (GS), local selection (LS), and random selection (RS) are the three steps of Gao's proposed GLR machine selection technique (Pezzella et al., 2008). This work suggests a way for the combination of the random selection of OS with the GSO of MA, based on the literature research mentioned above. Researchers created the multi-offspring genetic algorithm of single-point crossover. They used model formulation for the demonstration of the degree of optimization, but they didn't concentrate on minimizing the manufacture span for a given project (Jin & Wang, 2022). Another perspective on that kind of study effort was the optimization and application of this crossover, whereby they demonstrated that the genetic algorithm of the multi-offspring strategy enhances the single-point crossover, which performs better in addressing nonlinear mixed integer programming problems (Li et al., 2016). Several scheduling issue types were examined at various stages of the study process, taking various factors into account. For example, Delivery time, rate of delivery delay, service quality, and delay time (Xu et al., 2021). Several studies have been conducted on the Flexible Job Shop Scheduling Problem (FJSP) with the consideration of Handling and Setup Time together which was based on the Improved Discrete Particle Swarm approach (Kong & Wang, 2024). The performance efficacy of this approach is confirmed by testing and analysis of fifteen FJSP test examples. Lastly, by creating a FJSP test case including processing, setup, and handling times, the viability and efficacy of the developed method for solving multi-objective FJSPs are confirmed. But the optimization of the makespan was also missing in their conclusion. The spread of the PSO-solving technique was the study's main objective in preparation for its eventual application on systems or embedded systems which can make decisions in real-time based on resource conditions and unanticipated or unplanned occurrences (Nouiri et al., 2018). Two multi-agent-based methods are suggested for this purpose, and they are contrasted with various benchmark examples. One suggests several mathematically precise parameters tuning approaches, and these systems can be very useful in determining more suitable settings (Ding & Gu, 2020b). The findings of the last experiment demonstrate that the enhanced PSO has a great capacity to solve FJSP. The suggested HLO-PSO can be implemented readily and may be integrated into other different production system software or learning system software thanks to the thorough presentation and analysis (Ding & Gu, 2020a). A study's suggested algorithm was put to the test on benchmark situations, and the outcomes were contrasted with those of algorithms that already existed. Compared to the current algorithms, the suggested approach demonstrated better convergence performance and solution correctness (Liu et al., 2021). One aims to introduce a hybrid method which is comparatively new and a mathematical model for the FJSP issue with the activities related to assembly. Each product in a certain challenging environment, is made by the properly assembled system using several distinct elements (Fattahi et al., 2020) . The pieces go through a flexible job shop system processing stage initially, followed by assembly and product production in the second step. An enhanced particle swarm optimization technique of hybrid strategy (IH-PSO) is put out in a study to increase the effectiveness of addressing the FJSP having multiple objectives. (Y. Zhang et al., 2020). Additionally, the goals of the model of mixed-integer programming (MIP) were implemented to minimize both makespan and total carbon emissions for the research (Tan et al., 2021). A paper addressed a variation of the JSP, where a restricted number of trucks will be required to transfer tasks to the operations that are important for machine processing (Fontes et al., 2023). To improve the PSO's capacity for subsequent problem-solving, a portion of the study suggested an elite retention technique and integrated it (Wu et al., 2022). For the variation of the conditions for manufacturing and the tolerance of moderate delay, the time required for the processing of operations and the due time for the orders into the practical production scheduling is never fully estimated as predictable quantities, as explained in a work (Zhu & Zhou, 2021). The focus of the research project was multi-objective optimization for the FJSP issue of energy-consciousness including activities related to assembly (Ren et al., 2021). Additionally, the Hybrid Salp Swarm Algorithm was used to solve the Job Shop Green Scheduling Problem which was Double-Flexible, and completely different from others (Liu et al., 2022). The main finding of the study was that the enhanced gaming PSO effectively minimized the maximum completion duration of FJSP after evaluating benchmarks in a standard manner and was compared with the findings of the other results achieved by using other PSOs. But these were relatively improved than previous (Xu & Wang, 2021). A typical flexible job shop problem was solved by using the combination of a discrete particle swarm optimization algorithm with an adaptive inertia weight method (DPSO-AIW) which also used a Pareto optimal solution (Gu et al., 2020). Finally, a very small body of research was employed to apply the FJSP problem analysis to minimize the makespan. However, the combination of Discrete Particle Swarm Optimisation and Adaptive Inertia Weight (DPSO-AIW) was not used in the investigation. For the reduction of the total completion time for a patient in a hospital i.e. Flexible Job Shop Problem (FJSP), this research combines these two techniques. This is the novelty of this paper.

3. Formulation of FJSP

There are o jobs $K = (K_1, K_2, K_3, \dots, K_o)$ which is required to be presented on n number of machines where, $N = (N_1, N_2, N_3, \dots, N_n)$ Here, the term 'job' expresses the patients and machine expresses the rooms used for testing in hospitals. A job may contain more than one operation, and P_{jk} will represent the *k*th number of operations for the job j. According to the rules of FJSP, each of the operations will be performed on different testing rooms/machines at a time. Times required for the processing of P_{jk} , performed on the machine l is u_{jkl}, which will be larger than 0. The time required for the completion of the operation P_{jk} is D_{jk}. Sequence of the processing is provided here. To minimize design complexity, FJSP can be divided into two distinct parts. One is Total-FJSP which will be represented as (T-FJSP) and another is Partial-FJSP which will be treated as (P-FJSP). These two types are represented in the following table. There are some differences between the two types. For T-FJSP, any machine can be utilized for every operation of all the jobs. Alternatively, for the P-FJSP, operations may be performed on specific machines. This will represent the real subject of the machine set.

Table 1
T-FJSP of (2×4)

Job/Patient	Operations	Machines/Rooms			
		N_1	N_2	N_3	N_4
K_1	P_{11}	2	6	3	2
	P_{12}	5	4	1	3
	P_{13}	9	3	4	1
K_2	P_{21}	7	3	1	5
	P_{22}	3	1	4	6

Table 2
P-FJSP of (2 × 4)

Job/Patient	Operations	Machines/Rooms			
		N_1	N_2	N_3	N_4
K_1	P_{11}	2	6	3	2
	P_{12}	5	4	1	3
	P_{13}	9	3	4	1
K_2	P_{21}	7	3	1	5
	P_{22}	3	1	4	6

Multi-objective FJSP was considered here in this research of the minimization of the maximum completion time required for the machine to complete the job which is represented as G_1, to minimize the workload of the machine which is in bottleneck i.e. G_2, that will consider the balance of the workload towards all the machines for the prevention of the highly loaded work which was assigned in machine (single) and to minimize the summation of workload of that specific machine and it is represented by G_3, this is basically of interest in the assignment of the relatively short processing time required for the improvement of the efficiency. All these three objectives will drive the minimization of the total makespan for the completion of all the operations by a patient in the hospital so far. Mathematical Representation of the objective functions can be represented as follows:

$$G_1 = \min\left(max\left(D_{jk}\right)\right) \left(1 \le j \le o, 1 \le k \le o_j\right) \tag{1}$$

$$G_2 = \min\left(max\left(\sum_{j=1}^{o} \sum_{k=1}^{oj} u_{jkl}\right)\right) \left(1 \le l \le n\right) \tag{2}$$

$$G_3 = \min\left(max\left(\sum_{j=1}^{o} \sum_{k=1}^{oj} \sum_{l=1}^{n} u_{jkl} . y_{jkl}\right)\right) \tag{3}$$

$$y_{jkl} = \begin{cases} 1, if\ operation\ P_{jk} is\ processed\ in\ machine\ l \\ \qquad 0.\ otherwise \end{cases} \tag{4}$$

$$D_{jk} \ge 0 \tag{5}$$

$$D_{jk} - D_{jk-1} \ge u_{jkl} . y_{jkl} (k = 1,2, \dots, o_j; j = 1,2, \dots., o; l = 1,2, \dots, n) \tag{6}$$

$$\sum y_{jkl} = 1 \tag{7}$$

where, Eq. (7) indicates that 1 machine will be chosen from a pool of the machines that are available for the completion of each operation, and Equation no. 6 guarantees, the operations are related to that task which is the same and they will meet restrictions of the precedence.

Assumptions & constraints made for the operations performed by FJSP.

1. At time zero, every machine is available, and every work may be completed at that moment.
2. One operation can be handled by only one machine at a time at a given point in time. The machine can be utilized for other tasks when the procedure is finished.
3. Once processing starts, it cannot be stopped. Moreover, there are consecutive restrictions between the operations connected to complete the same job rather than sequential constraints between the operations of distinct tasks.
4. Both the machine's setup time and the operation's transit time are disregarded.

4. Particle Swarm Optimization

Kennedy (1995) introduced the notion of swarm intelligence, which is implemented based on the particle swarm optimization (PSO) algorithm. Kennedy was motivated by the habits of foraging of birds and other swarm organisms in the part nature (Gu et al., 2020). Numerous academics are drawn to the PSO algorithm due to its straightforward parameters, straightforward implementation, and potent global optimization capability. The fields related to function optimization, processing of images, fuzzy system control, and optimization for scheduling have all made extensive use of it. The continuous optimization problem was the first one that the PSO method was intended to answer. Nonetheless, many real-world engineering application issues are discrete. Thus, it has been a popular topic among academics to use the fundamental notion of the PSO algorithm and the transformation of it into a new version that is discrete to address the discrete issues of large-scale like combinatorial optimization. The analysis of the social behavior of foraging birds is the foundation of the particle swarm optimization method. In these situations, the algorithm first represents a collection of potential issue solutions using a set of particles. Next, every particle in the population recalls and proceeds to seek the solution space by following the currently optimum particle. Assume that in the E-dimensional search space, a collection of N particles z moves at a certain speed. The particle j's state property is configured as follows:

The particle's current position: $y_j = (y_{j1}, y_{j2}, \dots, y_{je})$;
The particle current velocity: $w_j = (w_{j1}, w_{j2}, \dots, w_{je})$;
The particle j's experienced the best position: $qC_j^u = (qC_{j1}^u, qC_{j2}^u, \dots, qC_{jd}^u)$;

hC^u is the site where the largest value is created, and it is the global ideal position perceived by the population. The required formula for jth particle for updating the position of a particle and the velocity of it at *(u+1)*th generation is given in Eqs. (8-9), respectively.

$$w_j^{u+1} = \omega \times w_j^u + d_1 \times s_1(qC_j^u - y_j^u) + d_2 \times s_2(hC^u - y_j^u) \tag{8}$$
$$y_j^{u+1} = y_j^u + w_j^{u+1} \tag{9}$$

Here,

ω = Inertia Weight for the population

w_j^u = Velocity at the current state

d_1 = Constant of acceleration for $1st$ particle

d_2 = Constant of acceleration for $2nd$ particle

r_1, r_2 = Random Number between (0,1)

They are included in the formula to mimic a small amount of unexpected group behavior in nature and to calculate the length of time the particle stays on the initial path from qC_j^u and hC^u. Additionally, it strikes between exploration and exploitation through its balance; qC_j^u refers to the ith particle's unique optimum location; hC^u refers to the optimal position of the current state.

Table 3
The process of PSO (a)

Job/Patient	Operations	Machines/Rooms			
		N_1	N_2	N_3	N_4
K_1	P_{11}	2	6	**4**	2
	P_{12}	5	4	**2**	3
	P_{13}	9	3	**5**	1
K_2	P_{21}	7	3	***1***	5
	P_{22}	3	1	**5**	6

Table 3
The process of PSO (b)

Job/Patient	Operations	Machines/Rooms			
		N_1	N_2	N_3	N_4
K_1	P_{11}	***2***	6	4	2
	P_{12}	**8**	4	2	3
	P_{13}	**10**	3	5	1
K_2	P_{21}	**8**	3	1	5
	P_{22}	**5**	1	5	6

Table 3
The process of PSO (c)

Job/Patient	Operations	Machines/Rooms			
		N_1	N_2	N_3	N_4
K_1	P_{11}	2	6	**6**	**2**
	P_{12}	8	4	***2***	**3**
	P_{13}	10	3	**7**	**1**
K_2	P_{21}	8	3	**3**	**6**
	P_{22}	5	1	**7**	**6**

Table 3
The process of PSO (d)

Job/Patient	Operations	Machines/Rooms			
		N_1	N_2	N_3	N_4
K_1	P_{11}	2	**6**	6	2
	P_{12}	8	**5**	2	3
	P_{13}	10	**4**	7	1
K_2	P_{21}	8	**4**	3	6
	P_{22}	5	***1***	7	6

Table 3
The process of PSO (e)

Job/Patient	Operations	Machines/Rooms			
		N_1	N_2	N_3	N_4
K_1	P_{11}	2	6	6	**3**
	P_{12}	8	5	2	**3**
	P_{13}	10	4	7	***1***
K_2	P_{21}	8	4	3	**7**
	P_{22}	5	1	7	**8**

Eq. (8) has three parts. First part is $\omega \times w_j^u$which expresses the velocity of the previous state which indicates the particle of the current state. Next part $d_1 \times s_1\left(qC_j^u - y_j^u\right)$ expresses the particle's self-cognitive part which is the influence of the particle. This will express the ability of global search and check whether it is strong; Lastly, $d_2 \times s_2(hC^u - y_j^u)$ expresses particles' social learning part which aids in information sharing among particles in different states.

5. Algorithm of DPSO-AIW

5.1 Encoding-Decoding

Every FJSP chromosome is represented by two coding layers. The sorting of the machine's assigned order is known as operations sequencing (OS). The job index is represented by the number in this sequence; the order using which jobs occur in the operations sequence indicates the order in which different operations of the job are processed. The second one is the machine assignment (MA), which entails designating each of the operations to a set of competent machines and computing the machine's start and finish times (Fontes et al., 2023). The process by which the analysis of the operation chooses the machines in its set of optional machines is referred to as the MA selection. The schedules produced in this manner are workable. To make sure that the actively performed schedule is created once the chromosome starts decoding, the decoding employs a plug-in greedy decoding algorithm (Zhang et al., 2020). The order of the operations on these sequences is decoded based on the chromosome's OS coding. The first action on these sequences is set up for processing for the first as a sequential order, and then the second one will be taken and placed on the associated machine's time for processing at the efficient processing time. This allows for the optimal placement of each operation in the sequence, starting as early as feasible.

5.2 Population Initialization

Several research suggested firstly a global selection (GS), secondly local selection (LS), and lastly random selection (RS) technique for the GLR machine selection. This research presents a strategy for the combination of the OS random selection with the GSO of MA, based on the literature review mentioned above. The procedure of implementation is provided in the following two points:

a. OS employs the first step i.e. random selection; then OS portion utilizes the operation related to the JSP mode selection in the coding, and for each of the operations, it will pick and create the OS randomly;
b. Then, MA causes the selection of GSO: since the OS is coded in a random order, every operation performed in the MA was picked by a machine required for the processing. Our objective is to choose the machine from the set of optional machines that has the global least burden at the processing.

The whole method can be expressed as follows,

For each operation, choose the machine from its set of optional processing machines with the least amount of work, and then the workload value was added to the loads of the machine of other operations into the same column. In Table 1, for instance, the randomly produced OS can be represented as *2 1 1 2 1* for the (2×4) T-FJSP. In A similar process, select another machine, let N_3 consisting of minimal workload in a machine set which is optional, having a value of 1. Then increase the remaining value slot of the column N_2 by 1 based on the value of the original one. Perform the remaining operation sequentially to get MA. The processing of them is shown in Table 3 (b)- Table 3(e). Then, selected operations are indicated in italic form and bold, where values that represent the machine workload (workload) are in bold. (2×4). The Gantt chart of T-FJSP is represented in Fig. 1.

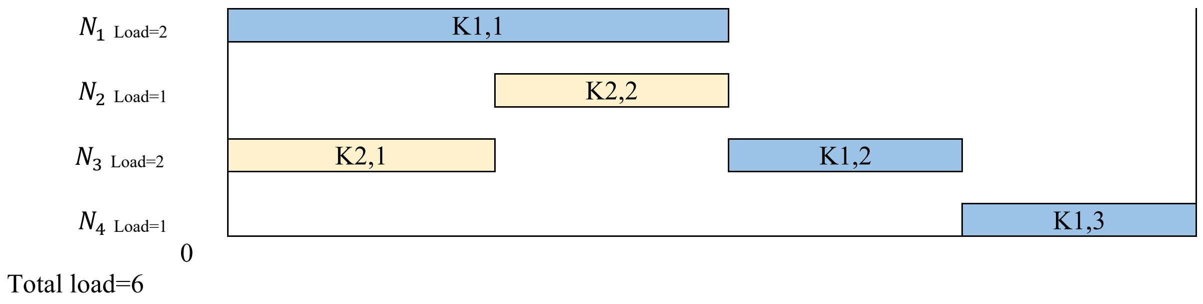


Fig. 1. Gantt Chart Representations for the T-FJSP of (2×4)

5.3 PSO Location Update Rule

An influence of the current velocity of the particle, the cognitive component, and the social component comprise the three key components of equations used to update the amount of velocity and the coordinate of position evolution in the basic PSO algorithm. Together, these 3 components determine the location of the upcoming generation of the particles. However, the basic PSO technique requires discrete issues and is not appropriate for continuous problems. To address this, one must update the formulas, simulate the original PSO algorithm's optimization process, and denote the placement of discrete particles in clusters based on the formulation of Eq. (10) shown below,

$$y_j^{u+1} = d_2 \otimes q\{d_1 \otimes r\left(\omega \otimes n\left(y_j^u\right), qC_j^u\right), hC^u \tag{10}$$

Here,

ω = Inertia Weight for the population

d_1 = Cognitive coefficient

d_2 = Social coefficient

$\otimes$ represents operations for optimization

$$N_j^u = \omega \otimes n\left(y_j^u\right) = \begin{cases} n\left(y_j^u\right) q_n < \omega \\ y_j^l, otherwise \end{cases} \tag{11}$$

The first part of Eq. (11) expresses the affected part for the utilization of the current state. $n\left(y_j^u\right)$ expresses the current particle velocity. A random number q_n expresses the mutation probability which ranges is 0 to 0.1. This will be applicable only when this random number is less than ω and will execute the operation $n\left(y_j^u\right)$. If this operation can't be executed using the base condition, then the original particle will remain unchanged. Here, $n(y_j^u)$ expresses the chromosome's operation for mutations. The operating system chooses a chromosome based on the likelihood of mutations and chooses an operation at random. Given the sequence order constraint on the job, the first step is to identify the positions of the precursor and subsequent operations. Next, randomly select a position between these two positions to insert the operation. This will used to ensure the schedule found from the result, is a workable solution. Based on the likelihood of the mutation, MA chooses the parent chromosome for the process of mutation and then chooses a processing procedure. Every operation has a set of machines for optional processing since every operation can be processed on several machines. Choose the machine in random order from the set of processing machines required to finish the mutation.

$$R_j^u = d_1 \otimes r(N_j^u, qC_j^u) = \begin{cases} r(N_j^u, qC_j^u) & q_{d1} < d_1 \\ N_j^u, otherwise \end{cases} \tag{12}$$

The learning part of the chromosome is expressed in equation 12 and $r(N_j^u, qC_j^u)$ expresses the chromosome adjustment based on the position qC_j^u which is optimal. The probability of the crossover q_{d1} is also a random number that range is (0.5 to 1.0) and it will be also less than $d_1$at the time of performing mutation of the crossover. Otherwise, the original particle will be unchanged as in the previous equation. $r(N_j^u, qC_j^u)$ is established using the operation of the crossover performed in the Genetic Algorithm (GA). For these 2 sets of codes of the FJSP, OS will adopt the precedence of the crossover of the operations (POX) (Nouiri et al., 2018) and MA will adopt the single crossover (ISX) (Ding & Gu, 2020).

$$Q_j^u = d_2 \otimes q(R_j^u, hC^u) = \begin{cases} q(R_j^u, hC^u) & Q_{d2} < d_2 \\ R_j^u, otherwise \end{cases} \tag{13}$$

Eq. (13) embodies the adjustment of the particle based on the position of a globally optimal particle and expresses the coordination between particles $q(R_j^u, hC^u)$ represents the operation of crossover between $R_j^u$and hC^u. Establish a crossover operation of the probability Q_{d2} must be less than that of d_2. The range of it is (0.5 to 1.0), otherwise, the original particles will remain unchanged. Operation processes of the crossover: OS will use the LOX type of crossover, and MA will use the improved Single-point crossover (ISX). Parent chromosomes after encoding will be presented as $Q_1, Q_2, Q_3, \ldots., Q_o$ and then the offspring-obtained chromosome will be presented as $D_1, D_2, D_3, \ldots., D_o$ (Where, o is used to express the size of the population). The steps related to POX are provided as follows,

Step 1: Q_1 & Q_2 taken in a sequence from the parent chromosome. Copy the operations including the job K_1 in Q_1 to D_1 which expresses the real order.
Step 2: Make the operations copy including the job represented as K_2 in the Q_1 to D_2 which also represents the original order, make copies of all operations related to the job K_1 in Q_2 to D_2 in the particular real order or the original one.
Step 3: Repeat steps 1 to 2 in chromosomes which are present until the offspring chromosome o and $D_1, D_2, D_3, \ldots., D_o$ are achieved.

Alternatively, the steps of LOX are provided as follows,

Step 1: Generation of 2 positions for the inspection in a random order in 2 parents Q_1 & Q_2; make sure the fragments are exchanged in two intersectional positions.
Step 2: Delete the gen of the original parents that was interchanged before.
Step 3: The remaining genes will be copied from the original parents from the first position. The whole process is represented in the flow chart of Fig. 2.

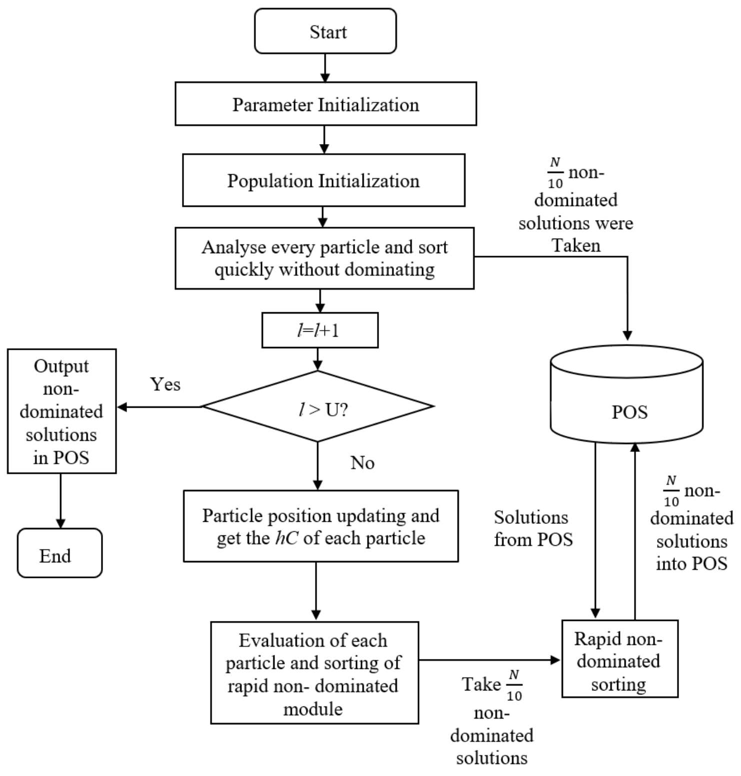
The method of ISX can be represented as Dividing all chromosomes connected to the crossover in (o/2) number of groups. Then, for each group's two parent chromosomes, perform single-point crossover by randomly selecting just a single crossover point, then after exchanging the machines that were assigned by the selected operations included with 2 parents before reaching the crossover point.

5.4 Adaptive Inertia Weight Calculations

The PSO algorithm's search procedure is intricate and nonlinear. An essential algorithmic parameter that balances the algorithm's capacity for both local and global search is the inertia weight. Large inertia weights are advantageous for global search, but lesser weight values can quicken algorithmic convergence and keep the algorithm from reaching a local optimum. To increase population variety, this article adopts the adaptively adjusted inertia weight approach. The current value's exponential function and the global optimum value of the particles in the population are used to update the inertia weight. The inertia weight is correlated with the particle's current value at each of the iterations; that is called inertia weight rises and vice versa when there is a significant interchange between the particle's present value and the ideal value of the globally organized.

$$\omega(t+1) = \omega_{start} + (\omega_{end} + \omega_{start}) \times (\omega_{end} + \omega_{start})^{\frac{e^{n_j(t)}+1}{e^{n_j(t)}-1}} \tag{14}$$

$$n_j(t) = \frac{gC^u - y_j^u}{gC^u + y_j^u} \tag{15}$$



Start
Parameter Initialization
Population Initialization
Analyse every particle and sort quickly without dominating
$\frac{N}{10}$ non-dominated solutions were Taken
l=l+1
l > U?
Yes
No
Output non-dominated solutions in POS
End
POS
Particle position updating and get the hC of each particle
Solutions from POS
$\frac{N}{10}$ non-dominated solutions into POS
Evaluation of each particle and sorting of rapid non- dominated module
Take $\frac{N}{10}$ non-dominated solutions
Rapid non-dominated sorting

Fig. 2. Flow chart representations for the DPSO-AIW Algorithm

In Eq. (14) and Eq. (15),

ω_{end} = The weight of inertia when the iteration is in the peak.
ω_{start} = The weight of the inertia at the current state.
u = Represents the iterations number of the current state.

5.5 Flow of DPSO-AIW

DPSO-AIW algorithm to resolve multi-objective FJSP issue is presented as follows,

Step 1: Set the parameters which will be used initially. The number of iterations U, the size of the population is O, and the weight of inertia ω, the coefficient of cognitive is d_1 and social coefficient d_2, with the cycle variable u. Let, the variable of cycle u = 1;
Step 2: Set the initial values for the population P. The GSO technique is used to create the MA, whereas the OS is generated at random. Assess every particle, quickly sort the population using non-dominated sorting, identify (N/10) non-dominated solutions from the present population, and then store the items into the Pareto optimal solution set (POS), which is (N/10);
Step 3: After determining each particle's value, quickly sort the population to find N/10 non-dominated solutions of the population of the current state. These should then be compared to N/10, which means that N/10 should be chosen from the N/5 solutions for updating the POS.

According to method the of quick sorting, the front G_j level of non-domination level. Let, the solution number in each front be o_j, and if $(\frac{N}{10}$-$\sum o_{j-1} \leq o_j)$, $(\frac{N}{10}$-$\sum o_{j-1})$ randomly selected as an individual in the front G_j to the POS store.

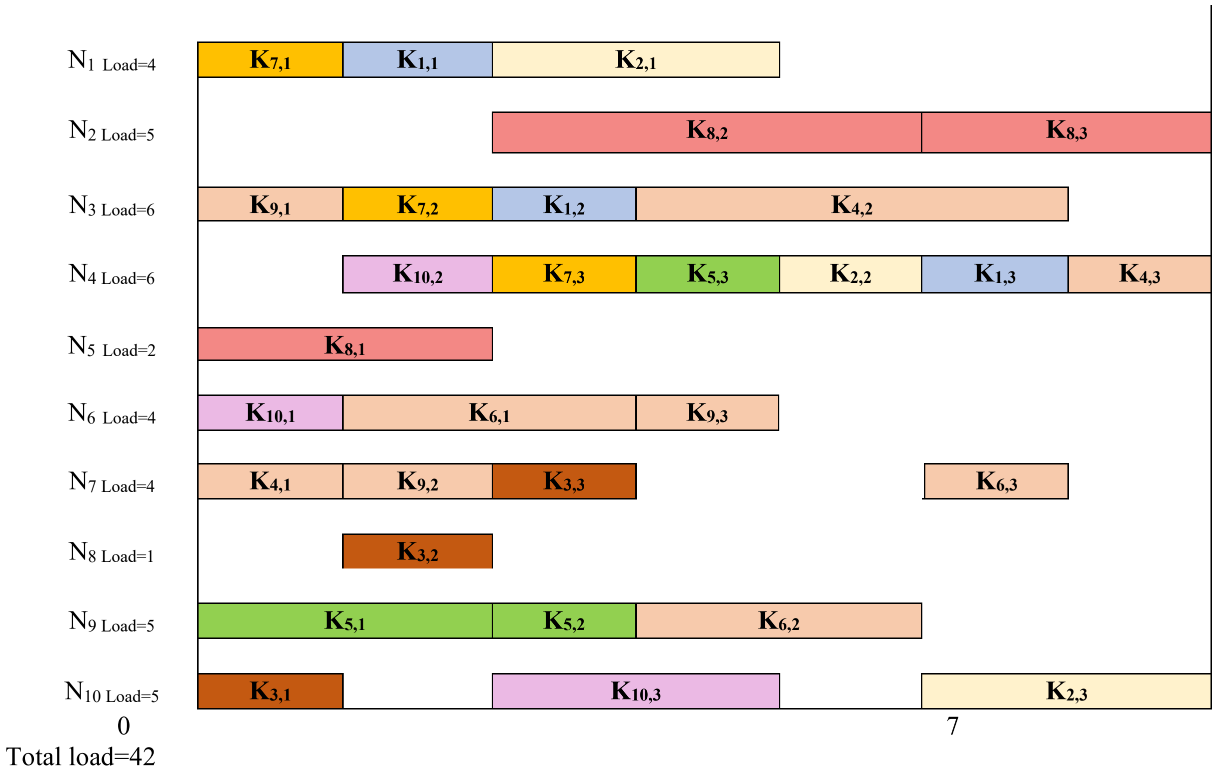


Fig. 3. Gantt charts produced by DPSO-AIW for 10 × 10 instances.

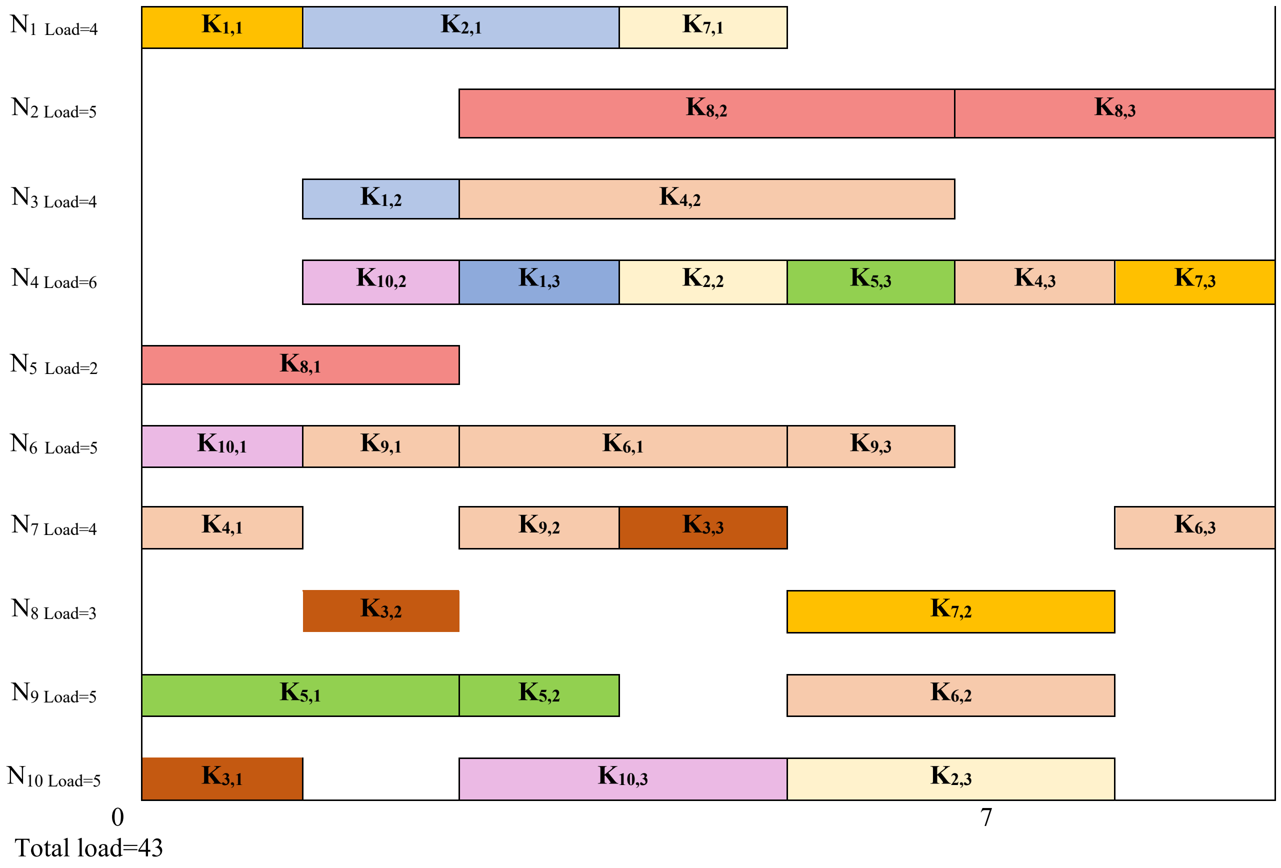


Fig. 4. Plotting diagrams produced by two distinct Pareto optimum solutions for 10 × 10 instances.

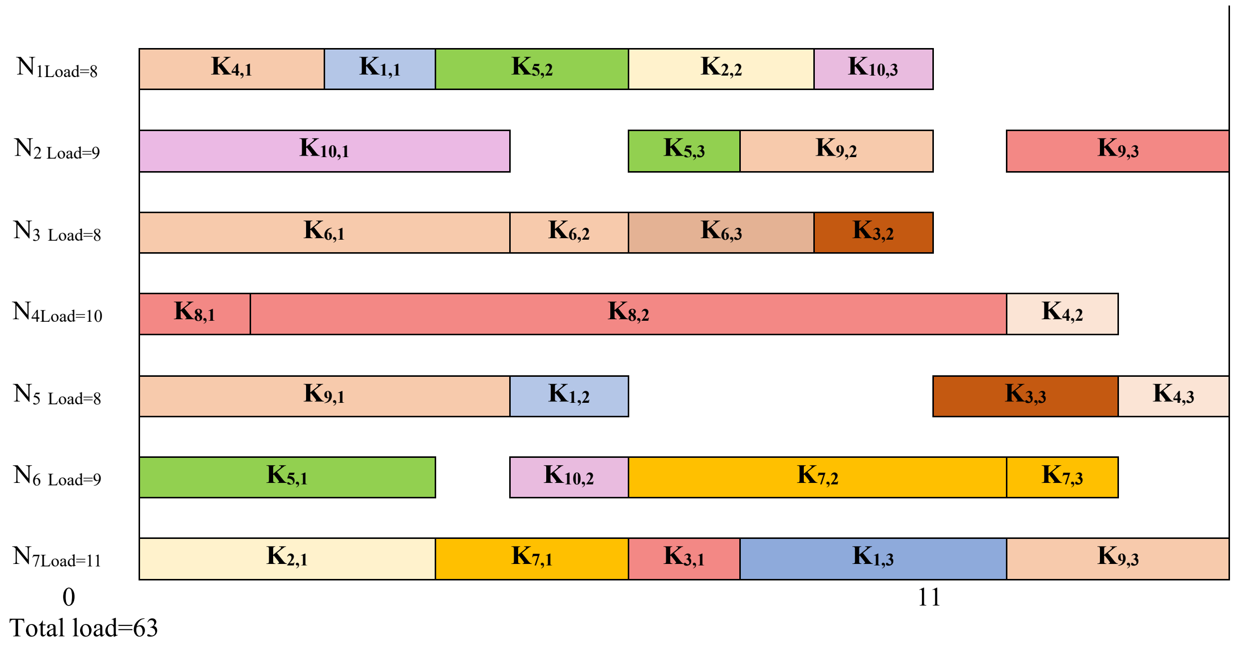


Fig. 5. Gantt charts produced by the DPSO-AIW for 10 × 7 instance.

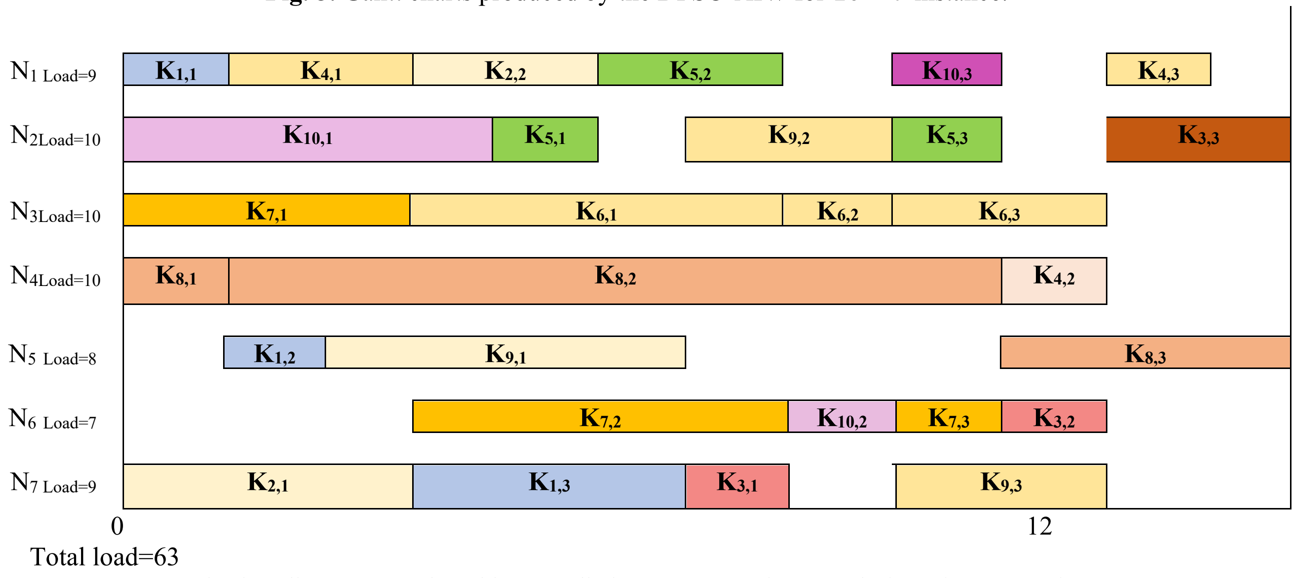


Fig. 6. Plotting diagrams produced by two distinct Pareto optimum solutions for 10 × 7 instances.

Step 4: Consider, $u = u+1$for the further iterations.
Step 5: Check whether, $(u> U)$ is satisfied or not. If satisfied go to the next step, otherwise, go to the previous one.
Step 6: Update the position of the particle, if it satisfies the conditions. Go to the step 3.
Step 7: POS's non-dominated solutions will be set as the output which is shown in figure 2.

Table 4
Optimal Solution Representations

Figure No.	Optimal Solution
3	(7,6,42)
4	(7,6,43)
5	(11,11,63)
6	(12,10,63)

6. Complexity Analysis

Let, there N machines with the size of the population Q, the length of the OS and MA is expressed by M, the Iterations number performed is U and the number of the objective is indicated by C. There are several parts of the complexity analysis for the DPSO-AIW algorithm. Firstly, population initialization, operations of crossover and mutations, Fastly performed non-dominated sorting with decoding. At the time of initialization of the populations, OS will be generated randomly, and iteration of computational complexity with the worst case will be found i.e. $P(M \times Q)$. At the time of performing GSO, MA will be generated where the computational complexity will be represented as $P(2 \times N \times M \times Q)$. Then, for the operation of mutation, OS will be mutating in a random order, and the computational complexity for that case will be $P(M \times Q)$. There are number of optional machines that will affect MA, the computational complexity for that case will be $P(N \times Q)$. Two times the operations for crossover will be performed. For performing the first crossover, POX type crossover will be used by OS. For that case, computational complexity will be $P(4 \times M \times Q)$. Then ISX crossover will be used by MA, their computational complexity will be $P(2 \times M \times Q)$. For the second operation of the crossover, LOX crossover will be taken by OS, their worst computational complexity will be $P(4 \times M \times Q)$. Then, ISX crossover will be used by MA, their computational complexity will be $P(2 \times M \times Q)$. For the decoding processes, there are two steps performed by OS. Firstly, selection of the corresponding machine in MA will be performed. Then, the processing time of scheduling will be found, decoding will be done to generate a solution which will be feasible. Their worst computational complexity will be expressed by $P(N \times M^2)$. The non-dominating sorting's computational complexity is $P(N \times C^2)$. The overall complexity can be found from calculation of individual complexity. The representation of that is provided below-

$$P(M \times Q) + P(2 \times N \times M \times Q) + P(M \times Q) + P(N \times Q) + [P(4 \times M \times Q) + P(2 \times M \times Q)] \times 2 + P(N \times M^2) + P(N \times C^2) \approx P\{[14 + 2 \times N) \times M + N] \times Q + P[N \times (M^2 + C^2)$$

This equation is basically related to the size of the population, machine number, objective number, chromosome length and number of iterations which is described above.

7. Comparative Analysis

Our proposed algorithm, DPSO-AIW will be compared with some of the existing algorithms found in different papers of the existing literature review. For this purpose, Numerical simulations were performed. The algorithm was written in the programming language, Python and run on HP EliteBook Intel(R) Core (TM) i7-8650U CPU 2.11 GHz with 16GB RAM. Usually, the scale of Multi-objective FJSP is relatively difficult for the solution. For different instances, the number of iterations will be $(10 \times n \times o)$. Where, the size of the population is $n \times o$, A value of 1.0 will be considered for ω_{start}, and 0.3 will be considered for ω_{end}. The value of d_1 and d_2 is considered as 0.80. The first data set instances will be 4×5, 10×7, 8×8, 10×10 and 15×10 instances. Only 8×8 instances will be considered for P-FJSP, and all the remaining will be considered for T-FJSP. The Gantt charts shown in the figure are the representations of 10×10 and 10×7 instances. At the time of performing the Pareto optimal solution, DPSO-AIW will be compared with other two established algorithms found from the existing literature review, such as, MOPSO+LS (Moslehi & Mahnam, 2011), and P-EDA (Wang et al., 2013). The comparative result is shown in the table 5. It may be observed from Table 5, for 4×5 instances, the optimal solution is found from the DPSO-AIW which is (11 6 30) and it is comparatively better than the non-dominated solution for the other two methods. Then for the 10×7 instances, the optimal solution (11 10 55) is comparatively better than others. For 8×8, (12 10 75) is optimal which is found for P-EDA, and it is better than the other two methods. From the tables representing the result, DPSO-AIW gives the same (6 5 32) as the other two methods. Lastly for 15×10, (10 10 90) is optimal and it is found from DPSO-AIW, better than the other two methods so far. So, it is clearly shown that DPSO-AIW provides better results for different instances compared to the other two methods. So, this one is more effective and feasible. All the optimal solutions are bold and indicated in Table 5.

Table 5
Comparative Result for different instances.

Size of the Instances $(o \times n)$	Objective Function	Result of MOPSO+LS	Result of P-EDA	Result of DPSO-AIW
(4×5)	Makespan (G_1)	16 16	12 12 12 13	**11** 11 12
	Total Workload(G_2)	7 8	11 10 9 9	**6** 6 7
	Maximum Workload(G_3)	33 34	33 34 34 33	**30** 30 31
(10×7)	Makespan (G_1)	16 16 17	12 12 14	**11** 11 12
	Total Workload(G_2)	11 12 12	10 11 14	**10** 10 12
	Maximum Workload(G_3)	55 56 56	55 56 56	**55** 55 55
(8×8)	Makespan (G_1)	15 15 15 16 17	**12** 14 15 16	13 14 13
	Total Workload(G_2)	13 14 14 12 12	**10** 12 12 13	10 11 11
	Maximum Workload(G_3)	77 75 75 76 77	**75** 76 76 77	75 75 76
(10×10)	Makespan (G_1)	6 8 7 7	6 7 7 8	**6** 6 7
	Total Workload(G_2)	5 6 6 7	5 5 6 6	**5** 5 5
	Maximum Workload(G_3)	32 34 34 32	32 33 34 31	**32** 32 32
(15×10)	Makespan (G_1)	12	10 12	**10** 10 11
	Total Workload(G_2)	10	12 12	**10** 11 10
	Maximum Workload(G_3)	92	91 92	**90** 91 92

Table 6
Result for MK01 instances

No. of	MOGA			P-EDA			DPSO-AIW		
Iteration	(G_1)	(G_2)	(G_3)	(G_1)	(G_2)	(G_3)	(G_1)	(G_2)	(G_3)
1	41	37	170	41	37	170	**41**	**37**	**169**
2	43	40	159	40	38	166	41	37	170
3	44	41	157	42	38	164	41	39	170
4	45	41	155	42	39	161	42	38	170

Table 7
Result for MK02 instances

No. of	MOGA			P-EDA			DPSO-AIW		
Iteration	(G_1)	(G_2)	(G_3)	(G_1)	(G_2)	(G_3)	(G_1)	(G_2)	(G_3)
1	27	27	152	27	27	152	27	27	153
2	28	28	147	28	28	146	**28**	**28**	**146**
3	30	28	146	29	29	145	28	30	151
4	30	30	144	30	30	144	29	31	146
5	32	32	142	31	31	143	30	30	149
6	34	34	141	32	27	151	31	30	150

Table 8
Result for MK03 instances.

No. of	MOGA			P-EDA			DPSO-AIW		
Iteration	(G_1)	(G_2)	(G_3)	(G_1)	(G_2)	(G_3)	(G_1)	(G_2)	(G_3)
1	205	134	885	205	205	852	**205**	**169**	**851**
2	205	136	883	211	211	850	205	171	857
3	205	145	872	214	214	845	207	169	848
4	205	199	856	222	222	843	209	169	851
5	214	200	851	223	223	840	211	169	863
6	215	211	850	232	232	836	211	171	864
7	222	200	851	241	241	834	211	173	864
8	223	200	848	249	250	832	211	177	871

BRdata instances were used for the testing in the second level. MK01, MK02 and MK03 instances were taken, and a comparison was performed for the Pareto optimal solution. The data for MOGA was taken from the literature review from the paper by Wang (2010) and data for P-EDA was taken from the previous tables. Here, the Pareto optimal solution was found also from the DPSO-AIW and it is shown in bold in the tables. For example, the non-dominated solution of (41 37 169) for MK01 instance obtained by this algorithm, is better than that of the other two. Similarly, for MK02 instance, (28 28 146) is optimal and it is for the same algorithm shown in Table 7. Then for MK03 instance, (205 169 851) is non-dominated solution found using DPSO-AIW. So for all these three instances, DPSO-AIW provides better results compared to others.

It is evident that when it comes to addressing BRdata cases, DPSO-AIW is better than MOGA and P-EDA. In three cases, DPSO-AIW can produce Pareto optimum solutions that are superior to those produced by the P-EDA and MOGA algorithms. The comparison between Kacem examples and BRdata instances shown above leads to the conclusion that the DPSO-AIW method is a useful tool for resolving multi-objective FJSP problems.

8. Sensitivity Analysis

For the determination of the impact of parameters which are important for the analysis of the algorithm, a sensitivity analysis was performed. A (4×5) instances were taken into consideration to perform this analysis. It is performed in DPSO-AIW environment with the combination of different parameters. Here, are four factors P, ω_{start} ,ω_{end} and (d_1, d_2) were considered shown in table 9. Then this experiment was presented as an orthogonal array representation in table 10. Each factor level has an optional value which is presented in table no. 9. Each of the experiments was performed independently 10 times. Standard deviation and makespan were determined and represented in the following tables. The mean is represented in Table 11. Here, the mean and standard deviation reflect the significance or impact of each parameter on others. Here, we can observe that, ω_{start} was considered the most influential factor for the calculation of the DPSO-AIW algorithm.

Table 9

Parameters Factor Level

Parameters	Factor Level		
	1	2	3
P	$\frac{o \times n}{2}$	$o \times n$	$o \times n \times 2$
ω_{start}	0.9	1	0.95
ω_{end}	0.2	0.5	0.6
(d_1, d_2)	0.5	0.8	0.9

Table 10
Representation of Orthogonal Array.

Serial	Level of Factor				
	P	ω_{start}	ω_{end}	(d_1, d_2)	Makespan
1	1	1	1	1	10.5
2	1	2	2	2	11.1
3	1	3	3	3	11.3
4	2	1	2	3	10.6
5	2	2	3	1	11.7
6	2	3	1	2	11.3
7	3	1	3	2	10.4
8	3	2	1	3	10.5
9	3	3	2	1	11.3

Table 11
Mean and Standard Deviation of the parameters.

Parameters	Level of Factor			
	1	2	3	δ
P	11.02	11.2	10.72	0.1812
ω_{start}	10.5	10.7	11.3	0.2276
ω_{end}	10.76	11.07	10.8	0.0726
(d_1, d_2)	11.16	10.87	10.76	0.1045

9. Conclusion And Future Scope

To minimize the makespan, DPSO-AIW was used in this study to solve a Multi-Objective FJSP problem for a specific hospital dataset. This research demonstrated that, in comparison to other well-established algorithms, the method in question yields superior results or the best solution. This claim was supported by sensitivity analysis and relative comparison. This algorithm finds the best outcome while using the least amount of time and offering the most flexibility. Three goals were considered in this case, and each was accomplished with success. Additionally, it demonstrates how to take a difficult real-world situation and simplify and effectively construct a simulation model. This is an illustration of how to use the latest developments in particle swarm optimization to address various aspects of the problem to find a better solution. After all, the method's implementation reduced the makespan, which was the research's result. This research can be expanded in the following ways:

1. These methods can be utilized to solve more complex real-life problems consisting of more objective functions and constraints.
2. The combination of other algorithms with particle swarm optimization can be made to find comparatively more precise results which will be faster than this one.

Acknowledgment

The authors would like to acknowledge the cooperation of the management of some renowned private hospitals to give us insights into our study.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Dai, Y. (2021). Improved NSGA-II Algorithm for multi-objective flexible job shop scheduling problem. *Journal of Physics: Conference Series*, *1952*(4). https://doi.org/10.1088/1742-6596/1952/4/042065

Ding, H., & Gu, X. (2020a). Hybrid of human learning optimization algorithm and particle swarm optimization algorithm with scheduling strategies for the flexible job-shop scheduling problem. *Neurocomputing*, *414*, 313–332. https://doi.org/10.1016/j.neucom.2020.07.004

Ding, H., & Gu, X. (2020b). Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem. *Computers and Operations Research*, *121*. https://doi.org/10.1016/j.cor.2020.104951

Fattahi, P., Bagheri Rad, N., Daneshamooz, F., & Ahmadi, S. (2020). A new hybrid particle swarm optimization and parallel variable neighborhood search algorithm for flexible job shop scheduling with assembly process. *Assembly Automation*, *40*(3), 419–432. https://doi.org/10.1108/AA-11-2018-0178

Fontes, D. B. M. M., Homayouni, S. M., & Gonçalves, J. F. (2023). A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources. *European Journal of Operational Research*, *306*(3), 1140–1157. https://doi.org/10.1016/j.ejor.2022.09.006

Gu, X. L., Huang, M., & Liang, X. (2020). A Discrete Particle Swarm Optimization Algorithm with Adaptive Inertia Weight for Solving Multiobjective Flexible Job-shop Scheduling Problem. *IEEE Access*, *8*, 33125–33136. https://doi.org/10.1109/ACCESS.2020.2974014

Huang, S., Tian, N., Wang, Y., & Ji, Z. (2016a). Multi-objective flexible job-shop scheduling problem using modified discrete particle swarm optimization. *SpringerPlus*, *5*(1). https://doi.org/10.1186/s40064-016-3054-z

Huang, S., Tian, N., Wang, Y., & Ji, Z. (2016b). Multi-objective flexible job-shop scheduling problem using modified discrete particle swarm optimization. *SpringerPlus*, *5*(1). https://doi.org/10.1186/s40064-016-3054-z

Huang, X., & Yang, L. (2019). A hybrid genetic algorithm for multi-objective flexible job shop scheduling problem considering transportation time. *International Journal of Intelligent Computing and Cybernetics*, *12*(2), 154–174. https://doi.org/10.1108/IJICC-10-2018-0136

Hui, H. (2012). Approach for multi-objective flexible job shop scheduling. *Advanced Materials Research*, *542–543*, 407–410. https://doi.org/10.4028/www.scientific.net/AMR.542-543.407

Institute of Electrical and Electronics Engineers. (n.d.). *Evolutionary Computation (CEC), 2010 IEEE Congress on : date, 18-23 July 2010.*

Jin, X., & Wang, F. (2022). A Multioffspring Genetic Algorithm Based on Sorting Grouping Selection and Combination Pairing Crossover. *Mathematical Problems in Engineering*, *2022*, 1–20. https://doi.org/10.1155/2022/4203082

Kacem, I., Hammadi, S., & Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. In *Mathematics and Computers in Simulation* (Vol. 60).

Kong, J., & Wang, Z. (2024). Research on Flexible Job Shop Scheduling Problem with Handling and Setup Time Based on Improved Discrete Particle Swarm Algorithm. *Applied Sciences*, *14*(6), 2586. https://doi.org/10.3390/app14062586

Lai, R., Gao, B., & Lin, W. (2021). Solving No-Wait Flow Shop Scheduling Problem Based on Discrete Wolf Pack Algorithm. *Scientific Programming*, *2021*. https://doi.org/10.1155/2021/4731012

Li, M., Qianting, L., Meiqiong, M., & Sicong, L. (2016). Optimization and Application of Single-point Crossover and Multi-offspring Genetic Algorithm. *International Journal of Hybrid Information Technology*, *9*(1), 1–8. https://doi.org/10.14257/ijhit.2016.9.1.01

Liu, C., Yao, Y., & Zhu, H. (2022). Hybrid salp swarm algorithm for solving the green scheduling problem in a double-flexible job shop. *Applied Sciences (Switzerland)*, *12*(1). https://doi.org/10.3390/app12010205

Liu, Z., Wang, J., Zhang, C., Chu, H., Ding, G., & Zhang, L. (2021). A hybrid genetic-particle swarm algorithm based on multilevel neighbourhood structure for flexible job shop scheduling problem. *Computers and Operations Research*, *135*. https://doi.org/10.1016/j.cor.2021.105431

Moslehi, G., & Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, *129*(1), 14–22. https://doi.org/10.1016/j.ijpe.2010.08.004

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*, *29*(3), 603–615. https://doi.org/10.1007/s10845-015-1039-3

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers and Operations Research*, *35*(10), 3202–3212. https://doi.org/10.1016/j.cor.2007.02.014

Piroozfard, H., Wong, K. Y., & Wong, W. P. (2018). Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm. *Resources, Conservation and Recycling*, *128*, 267–283. https://doi.org/10.1016/j.resconrec.2016.12.001

Ren, W., Wen, J., Yan, Y., Hu, Y., Guan, Y., & Li, J. (2021). Multi-objective optimisation for energy-aware flexible job-shop scheduling problem with assembly operations. *International Journal of Production Research*, *59*(23), 7216–7231. https://doi.org/10.1080/00207543.2020.1836421

Tan, W., Yuan, X., Huang, G., & Liu, Z. (2021). Low-carbon joint scheduling in flexible open-shop environment with constrained automatic guided vehicle by multi-objective particle swarm optimization. *Applied Soft Computing*, *111*. https://doi.org/10.1016/j.asoc.2021.107695

Wang, L., Wang, S., & Liu, M. (2013). A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem. *International Journal of Production Research*, *51*(12), 3574–3592. https://doi.org/10.1080/00207543.2012.752588

Wang, X., Gao, L., Zhang, C., & Shao, X. (2010). A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, *51*(5–8), 757–767. https://doi.org/10.1007/s00170-010-2642-2

Wu, M., Yang, D., & Liu, T. (2022). An Improved Particle Swarm Algorithm with the Elite Retain Strategy for Solving Flexible Jobshop Scheduling Problem. *Journal of Physics: Conference Series*, *2173*(1). https://doi.org/10.1088/1742-6596/2173/1/012082

Xu, M., Lu, J., Zhu, F., Yu, F., Han, T., & Xu, M. (2021). Research and application for hydraulic cylinder workshop scheduling considering on time delivery rate. *Chinese Control Conference, CCC*, *2021-July*, 1905–1910. https://doi.org/10.23919/CCC52363.2021.9550547

Xu, X., & Wang, L. (2021). An Improved Gaming Particle Swarm Algorithm Based the Rules of Flexible Job Shop Scheduling. *ICSAI 2021 - 7th International Conference on Systems and Informatics*. https://doi.org/10.1109/ICSAI53574.2021.9664124

Zhang, J., Jie, J., Wang, W., & Xu, X. (2017). A hybrid particle swarm optimisation for multi-objective flexible job-shop scheduling problem with dual-resources constrained. In *Int. J. Computing Science and Mathematics* (Vol. 8, Issue 6).

Zhang, S., & Gu, X. (2023). A discrete whale optimization algorithm for the no-wait flow shop scheduling problem. *Measurement and Control (United Kingdom)*, *56*(9–10), 1764–1779. https://doi.org/10.1177/00202940231180622

Zhang, Y., Cheng, S., Shi, Y., Gong, D. W., & Zhao, X. (2019). Cost-sensitive feature selection using two-archive multi-objective artificial bee colony algorithm. *Expert Systems with Applications*, *137*, 46-58. https://doi.org/10.1016/j.eswa.2019.06.044

Zhang, Y., Zhu, H., & Tang, D. (2020). An improved hybrid particle swarm optimization for multi-objective flexible job-shop scheduling problem. *Kybernetes*, *49*(12), 2873–2892. https://doi.org/10.1108/K-06-2019-0430

Zhu, Z., & Zhou, X. (2021). A multi-objective multi-micro-swarm leadership hierarchy-based optimizer for uncertain flexible job shop scheduling problem with job precedence constraints. *Expert Systems with Applications*, *182*. https://doi.org/10.1016/j.eswa.2021.115214