OPTIMIZED SCHEDULING IN FLEXIBLE JOB SHOP PROBLEM WITH MAINTENANCE MANAGEMENT USING HYBRID DISCRETE FIREFLY ALGORITHM

1. S.Karthikeyan, Professor, Department of Mechanical Engineering, Indra Ganesan College of Engineering, Manikandam, Trichy drkarthikeyan.shanmugam@gmail.com
2. M.Santhi, Professor, Department of Mechanical Engineering, Indra Ganesan College of Engineering, Manikandam, Trichy
3. SP.KalaiSelvan, Assistant Professor, Department of Mechanical Engineering Sudharsan Engineering College, Sathiyamangalam
4. D.Sriram, Assistant Professor, Department of Mechanical Engineering Sudharsan Engineering College, Sathiyamangalam

Abstract

Most scheduling problems, including the standard Flexible Job Shop Scheduling Problem (FJSP), typically assume that machines remain continuously available throughout the entire planning horizon. While this assumption may be valid in certain scenarios, it often fails to reflect real-world situations where machines may experience downtime due to maintenance, pre-scheduled tasks, or unforeseen disruptions. Therefore, a more practical scheduling model must incorporate machine availability constraints. In this study, we address the Flexible Job Shop Scheduling Problem with non-fixed availability constraints (FJSP-nfa), where the completion time of maintenance tasks is not predetermined and must be determined during the scheduling process. To solve this problem, we propose a Hybrid Discrete Firefly Algorithm (HDFA) designed for multi-objective optimization. The algorithm simultaneously minimizes three objectives: the maximum completion time, the workload of the critical machine, and the total workload across all machines. Since the traditional Firefly Algorithm was originally developed for continuous optimization problems, it cannot be directly applied to discrete problems. To overcome this limitation, we introduce a modified approach that adapts the algorithm for discrete optimization. This involves converting continuous functions such as attractiveness, distance, and movement into appropriate discrete functions for machine assignment and operation sequencing. Additionally, to enhance solution quality, the algorithm integrates a neighbourhood-based local search method. This local search is applied to promising solutions, focusing on the concept of the critical path to identify and explore improvements. The effectiveness of the proposed algorithm is validated using representative FJSP-nfa benchmark problems, demonstrating its capability to provide efficient and high-quality scheduling solutions.

Keywords: Flexible job shop scheduling problem, Hybrid discrete firefly algorithm, multi-objective optimization, Maintenance activity, Local search.

INTRODUCTION

In production scheduling, the classical Job Shop Scheduling Problem (JSP) is classified as NP-hard [1]. The Flexible Job Shop Scheduling Problem (FJSP) extends the classical JSP by allowing an operation to be processed on one machine selected from a set of alternatives, making it more reflective of real-world manufacturing scenarios. Compared to JSP, FJSP is more challenging because it involves both assigning operations to machines and sequencing those operations. This introduces two sub-problems: the routing sub-problem, which assigns operations to machines, and the scheduling sub-problem, which sequences these operations to generate an optimal schedule [2]. Due to its NP-hard nature, no exact method has yet been proposed that can solve FJSP within a reasonable time frame. Consequently, numerous heuristic and metaheuristic approaches, including Tabu Search (TS), Simulated Annealing (SA), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Variable Neighborhood Search (VNS), and Artificial Immune Algorithm (AIA), have been employed to obtain optimal or near-optimal schedules efficiently.

Recently, the multi-objective version of FJSP has gained considerable interest among researchers. Solutions to multi-objective FJSP can be approached in two ways: by combining all objectives into a single weighted objective or using Pareto-based methods.

Machine Availability Constraints

Most scheduling models in the literature assume continuous machine availability throughout the planning horizon. While this assumption may hold in certain cases, real-world scenarios often involve machine downtime due to maintenance, breakdowns, or pre-scheduled tasks [3]. Consequently, a more realistic scheduling model must incorporate machine availability constraints.

Availability constraints can generally be categorized into two types: fixed and non-fixed. Fixed availability constraints specify predetermined unavailability periods starting at fixed time points. In contrast, non-fixed availability constraints allow flexibility in determining the starting time of unavailability periods during the scheduling process. For instance, preventive maintenance (PM) tasks are often assigned within a flexible time window, permitting adjustments to the start time [4].

Schmidt [3] reviewed studies on deterministic scheduling problems with availability constraints published before 1998. Since then, numerous studies have explored production scheduling with maintenance activities across various machine environments and job characteristics. Recent surveys by Ma et al. [5] indicate that most research has addressed single machine problems, parallel machine problems, flow shop problems, and job shop problems, with limited work on FJSP with availability constraints.

Gao et al. [4] proposed a hybrid Genetic Algorithm (GA) combined with a local search method for solving multi-objective FJSP with preventive maintenance tasks. Zribi et al. [6] investigated the MPM job shop scheduling problem under maintenance constraints. Wang and Yu [7] applied a Filtered Beam Search (FBS) algorithm to FJSP with maintenance tasks. Rajkumar et al. [8] introduced a GRASP algorithm for multi-objective FJSP with non-fixed availability constraints. Li and Pan [9] presented a discrete chemical-reaction optimization (DCRO) algorithm for FJSP

with maintenance activities, while Moradi et al. [10] integrated preventive maintenance tasks into a multi-objective FJSP framework.

Most of these studies have focused on fixed machine availability constraints, where maintenance task times are predetermined. However, in practical applications, preventive maintenance tasks often impose non-fixed availability constraints, meaning the starting time of unavailability periods must be determined during the scheduling process [4]. Few studies, including those by Graves and Lee [11], Aggoune [12], Kubzin and Strusevich [13], and Gao et al. [4], have addressed non-fixed availability constraints.

The Firefly Algorithm (FA) is a nature-inspired metaheuristic algorithm, introduced by Yang in 2008 [14], based on the flashing behavior of fireflies. Yang [15] applied the FA to multimodal optimization problems, while Lukasik and Zak [16] adapted it for constrained continuous optimization, demonstrating its efficiency. Sayadia et al. [17] used a discrete firefly algorithm for minimizing makespan in permutation flow shop scheduling problems. Jati [18] proposed a discrete FA for solving the Traveling Salesman Problem (TSP), and Khadwilard et al. [19] applied FA to job shop scheduling problems. Additionally, Marichelvam et al. [20] introduced a discrete FA using the SPV rule for multi-objective hybrid flow shop scheduling problems.

While the traditional FA is primarily used for continuous optimization, its direct application to discrete optimization problems is limited. The standard FA relies on real-number operations, making it unsuitable for discrete problem-solving.

This paper proposes a Hybrid Discrete Firefly Algorithm (HDFA) to address the multi-objective Flexible Job Shop Scheduling Problem with non-fixed availability constraints (FJSP-nfa). We consider a deterministic case where maintenance task durations are known in advance and fixed. Additionally, operations are strictly non-preemptive, meaning once an operation begins, it cannot be interrupted except by a maintenance task or another operation.

The primary objectives of this study are to minimize:

The maximum completion time,

The workload of the critical machine, and

The total workload across all machines.

The rest of the paper is organized as follows: Section 2 presents the problem formulation and notation. Section 3 introduces the standard Firefly Algorithm. The proposed algorithm is detailed in Section 4. Section 5 presents the computational results based on well-known FJSP-nfa benchmark problems. Finally, Section 6 concludes with key findings and suggestions for future research.

PROBLEM DESCRIPTION

The Flexible Job Shop Scheduling Problem with Non-Fixed Availability Constraints (FJSP-nfa) belongs to the broad field of deterministic scheduling with machine availability constraints. It can be described as follows:

Each job J_i ($1 \leq i \leq n$) consists of a sequence of n_i operations. Each operation O_{ij} ($i = 1,2,...,n$; $j = 1,2,...,n_i$) of job (J_i) can be processed by one machine m_{ij} in the set of eligible machines M_{ij}. The

processing time of an operation O_{ij} on the machine M_k is p_{ijk} ($M_k \in M_{ij} \subseteq M$). There are L_k (k =1, 2, . . . , m)

A key assumption is that the operations are non-resumable, meaning that if an operation is interrupted by maintenance, it must be reprocessed entirely. The objective is to assign operations to machines and determine the schedule for both job operations and maintenance tasks, adhering to the following constraints:

1. The sequence of operations for each job is fixed.
2. Each machine processes only one operation or preventive maintenance (PM) task at a time.
3. Each maintenance task must be completed within its time window.

Objectives

The following three criteria are to be minimized:

Makespan (C_m) of the jobs, i.e. the completion time of all jobs
Maximal machine workload (W_m), i.e. the maximum working time spent on any machine
Total workload of the machines (W_t) which represents the total working time over all machines.

Assumptions

jobs are to be scheduled on machines.

Each job has ordered operations.

Every machine processes only one activity (operation or maintenance) at a time.

Maintenance tasks have a predefined time window.

Setup time is machine-independent and included in the processing time.

Indices and Parameters

: i, h index of jobs, i, h = 1,2, . . . , n
j, g index of operation sequence, j, g = 1,2,. . . , n_i
k index of machines, k = 1,2, . . . , m
l index of maintenance tasks, l = 1,2, . . . , L_k

Parameters

n Total number of jobs
m Total number of machines
n_i Total number of operations of job i
L_k Total number of preventive maintenance tasks on machine k
O_{ij} The jth operation of job i
PM_{kl}The lth preventive maintenance task on machine k
M_{ij} The set of available machines for the operation O_{ij}
p_{ijk} Processing time of operation O_{ij} on machine k

d_{kl} Duration of the maintenance task PM_{kl}
C_{ij} Completion time of the operation O_{ij}
y_{kl} Completion time of PM_{kl}
$[t_{kl}^{E}, t_{kl}^{L}]$ Time window associated with PM_{kl} where t_{kl}^{E} is the early starting time and t_{kl}^{L} is the late completion time

Decision variable

$$x_{ijk} = \begin{cases} 1, if\ machine\ k\ is\ selected\ for\ the\ operation\ \ O_{ij} \\ 0, otherwise \end{cases}$$

Under these assumptions and notations, the proposed mathematical model for the problem is defined as follows.

$$minf_1 = \max_{1 \le i \le n} \left(C_{in_i} \right) \qquad (1)$$

$$minf_2 = \max_{1 \le k \le m} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n_i} x_{ijk} \, . \, p_{ijk} + \sum_{l=1}^{L_k} d_{kl} \right\} \qquad (2)$$

$$minf_3 = \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n_i} x_{ijk} . p_{ijk} + \sum_{k=1}^{m} \sum_{l=1}^{L_k} d_{kl} \qquad (3)$$

Where

f_1 = maximum completion time of all jobs
f_2 = maximum workload of all machines
f_3 = the total workload of all machines

The weighted sum of the above three objective values are taken as the combined objective function:

Minimise $F(c) = W_1 \times f_1 + W_2 \times f_2 + W_3 \times f_3$ (4)

Subject to:

$W_1 + W_2 + W_3 = 1, \quad 0 \le W_1, W_2, W_3 \le 1$ (5)

Objective Functions

The problem is formulated as follows:

The weighted sum of these objective values forms the combined objective function:

Where:

Firefly Algorithm

The Firefly Algorithm (FA) is a population-based metaheuristic technique for solving continuous optimization problems, particularly NP-hard problems. Inspired by the flashing behavior of fireflies, the algorithm models the attractiveness of fireflies as a function of their brightness, which is determined by the objective function value.

Idealized Rules of FA

Unisex Behavior: All fireflies are attracted to others regardless of gender.

Attraction and Movement: A less bright firefly moves toward a brighter one. Attractiveness is proportional to brightness and inversely proportional to distance.

Brightness Evaluation: For a minimization problem, brightness is inversely proportional to the objective function value.

These rules are integrated into the optimization process, leading to efficient and near-optimal solutions for complex scheduling problems.

Firefly Algorithm
Objective function f(X), $X=(X_1, \ldots, X_d)^T$
Generate initial population of fireflies X_i (i=1,2,...,n)
Light intensity I_i at X_i is determined by $f(X_i)$
Define light absorption coefficient γ
While (*t < Max Generation)*
for *i =1 : n all n fireflies*
for *j = 1 : I all n fireflies*
if *($I_j > I_i$), Move firefly i towards j in d-dimension;* **end if**
Attractiveness varies with distance r via exp [-*γr*]
Evaluate new solutions and update light intensity
end for *j*
end for *i*
Rank the fireflies and find the current best
end while
Post process results and visualization

Fig. 1: Pseudocode of the firefly algorithm

Based on Yang [15], FA is very efficient in finding the global optimal value with high success rates. Simulations and results indicate that FA is superior to both PSO and GA in terms of both efficiency and success rate. These facts give inspiration to investigate how to find optimal solution using FA in solving FJSP. The challenges are how to compute discrete distance between two fireflies and how they move in coordination. The following issues are important in this algorithm.

Distance

The distance between any two fireflies *i* and *j*, at positions x_i and x_j, respectively can be defined as a Cartesian distance:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \qquad (6)$$

where $x_{i,k}$ is the k^{th} component of the spatial coordinate x_i of the i^{th} firefly and d is the number of dimensions.

Attractiveness

The attractiveness of a firefly is determined by its light intensity. Each firefly has its distinctive attractiveness β which implies how strong it attracts other members of the swarm. The form of attractiveness function of a firefly is the following monotonically decreasing function [16]

$$\beta(r) = \beta_0 e^{-\gamma r^m}, (m \geq 1) \qquad (7)$$

where r is the distance between two fireflies, β_0 is the attractiveness at $r = 0$ and γ is a fixed light absorption coefficient.

Movement

The movement of a firefly i which is attracted by a more attractive (i.e., brighter) firefly j is given by the following equation [16]

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(rand - 1/2) \qquad (8)$$

where the first term is the current position of a firefly, the second term is due to attraction and the third term is a randomization with α being the randomization parameter, while *rand* is a random number generator uniformly distributed in the space [0, 1].

Based on the effectiveness of the firefly algorithm in optimizing continuous problems, it is predicted that this algorithm will be impressive in solving discrete optimization problems.

HYBRID DISCRETE FIREFLY ALGORITHM

The firefly algorithm has been originally developed for solving continuous optimization problems. The firefly algorithm cannot be applied directly to solve the discrete optimization problems. In this study, we propose a possible way that can be modified to solve the class of discrete problems, where the solutions are based on discrete job permutations. In discrete firefly algorithm, the target individual is represented by two vectors: one is for the machine assignment and the other is for permutation of jobs. Hamming distance is used to measure the distance between two permutations. Movement is implemented by breaking the attraction step into two sub steps as β-step and α-step.

Solution Representation

In the proposed algorithm, each solution contains two components, i.e., the machine assignment component and the operation scheduling component. Each element in the machine assignment component gives the selected machine for the corresponding operation, while each element in the scheduling component denotes the job number being operated. Consider an example problem with three jobs and two machines, where each machine has one maintenance task as shown in Table 1. Table 2 gives the operation processing time for the problem. One solution for the example problem is {1, 2, 2, 1, 1, 2, 1, 1 | 1, 1, 2, 2, 1, 2, 3, 3}. The first part is the machine assignment component while the second part is the scheduling component. The corresponding machine assignment and scheduling component of the solution are given in Figs. 2 and 3, respectively.

Table 1: Preventive Maintenance Tasks

PM tasks		Time window		Duration (d_{kl})
		t_{kl}^{E}	t_{kl}^{L}	
M_1	PM_{11}	0	7	3
M_2	PM_{21}	1	8	3

Table 2: Example of FJSP with 3 jobs and 2 machines

Job	Position	Operation	M_1	M_2
J_1	1	O_{11}	2	3
	2	O_{12}	5	3
	3	O_{13}	1	1
J_2	4	O_{21}	2	2
	5	O_{22}	2	2
	6	O_{23}	3	2
J_3	7	O_{31}	5	2
	8	O_{32}	1	2

Position	1	2	3	4	5	6	7
Operation	O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O
Machine	1	2	2	1	1	2	1

Fig. 2: Machine Assignment Component

Sequence	1	1	2	2	1	2	3
Operation	O_{11}	O_{12}	O_{21}	O_{22}	O_{13}	O_{23}	O_3
Position	1	2	4	5	3	6	7

Fig. 3: Operation Scheduling Component

It is notable that the solution encoding given above contains no scheduling information for the PM tasks. In this study, the maintenance tasks are scheduled using the heuristic below:

Step 1: The maintenance tasks on each machine are firstly scheduled at the end of their time window, i.e., set $y_{kl} \leftarrow t_{kl}^{L}$.

Step 2: When schedule an operation O_{ij} on machine M_k, denote the possible start time and end time of O_{ij}, without considering the PM tasks, S_{ij} and C_{ij}, respectively.

Step 3: If each maintenance task PM_{kl}, does not overlap with the operation O_{ij}, then schedule O_{ij} at duration [S_{ij} , C_{ij}]. Otherwise, perform Step 4.

Step 4: If [S_{ij} , C_{ij}] is overlapped with a maintenance task PM_{kl}, shift PM_{kl} to left as possible as compact, then schedule O_{ij} after PM_{kl}.

The decoding of the machine assignment, operation scheduling and maintenance tasks for the example solution mentioned above can be represented as a following schedule with starting and completion time of each operation on its assigned machine.

S_1 = {(O_{11}, M_1: 0-2), (O_{12}, M_2: 2-5), (O_{21}, M_1: 5-7), (O_{22}, M_1: 7-9), (O_{13}, M_2: 8-9), (O_{23}, M_2: 9-11), (O_{31}, M_1: 9-14), (O_{32}, M_1: 14-15), (PM_{11}: 2-5), (PM_{21}: 5-8)}

Fig. 4 gives the Gantt chart of the above mentioned schedule.

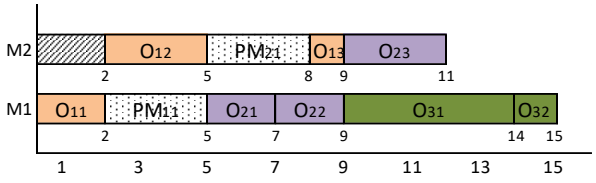


Fig. 4: Gantt chart

Population Initialization

The quality of the initial population has a greater effect on the performance of an algorithm. A good initial population locates promising areas in the search space and provides enough diversity to avoid premature convergence.

Machine Assignment Component Initial Rules

Initiate the machine assignment component of the population using the following two rules: the operation minimum processing time rule [21] denoted by R_{ma1}, the global machine workload balance rule [22] denoted by R_{ma2}.

Scheduling Component Initial Rules

The scheduling component considers how to sequence the operations at each machine, i.e., to determine the start time of each operation. Following are initial approaches for scheduling component: the most work remaining (MWR) rule [23] denoted by R_{sc1}, the most number of operations remaining (MOR) rule [21] denoted by R_{sc2}.

To consider both the problem features and solution quality, in the first part of the population, machine assignment components and scheduling components are generated according to percentage of population size given for rules R_{ma1}, R_{ma2}, R_{sc1}, and R_{sc2} respectively. All other solutions in the initial population are generated randomly to enhance the diversity of the population.

Table 3: Illustration of Firefly Solution Updation

Solution vector	**Machine Assignment**	**Operation Scheduling**
Current firefly position (*P*)	1 2 2 1 1 2 1 1	1 2 4 5 3 6 7 8
Combined objective function	*F(c)* = 16.8	
Best firefly position (P_{best})	1 2 1 1 1 2 2 1	17 4 2 5 6 3 8

Combined objective function	*F(c)* = 13.0	
Difference between the elements *(d)*	{(3,1), (7,2),}	{(2,7), (4,7), (5,7)}
Hamming distance *(r)*	2	3
Attractiveness β – step: $\boldsymbol{\beta(r) = \frac{\beta_0}{(1+\gamma.r^2)}}$	0.71	0.53
***rand ( )* between (0,1)**	{ 0.64, 0.66}	{ 0.3,0.92, 0.06 }
Movement β – step	(3,1), (7,2)	(2,7), (5,7)
Firefly position after β – step	1 2 1 1 1 2 2 1	1 7 4 5 2 6 3 8
Attractiveness α – step: α(rand$_{int}$)	1 2 1 1 1 2 2 1	4 1 7 5 2 6 3 8
Combined objective function	*F(c)* = 13.0	

Firefly Evaluation

Each firefly is represented by machine assignment vector and operation scheduling vector. By using the permutation of these vectors in the population each firefly is evaluated to determine the objective function. The objective function value of each firefly is associated with the light intensity of the corresponding firefly. In this work, the evaluation of the goodness of schedule is measured by the combined objective function which can be calculated using equation (4). For example, the combined objective function value for the schedule mentioned from the example problem is F*(c)* = 16.8.

Solution Updation

In the firefly algorithm, the movement of fireflies is guided by light intensity, with comparisons made between pairs of fireflies. The attractiveness of a firefly is determined by its brightness, which corresponds to the encoded objective function. Consequently, a dimmer firefly will move toward a brighter one. If no other firefly is brighter, it moves randomly.

In this study, the standard firefly algorithm is adapted by discretizing its distance, attraction, and movement functions. Table 3 presents how the solution updates between two fireflies using these modified functions. The distance from the current firefly position to the best firefly position is calculated using the Hamming distance for the machine assignment vector and the number of swaps for the operation scheduling vector.

The attraction process consists of two steps: the β-step and the α-step. During the β-step, the β-probability is determined using the Hamming distance, followed by an insertion or pairwise exchange on the current firefly's elements if a randomly generated number is less than the β-probability. The α-step introduces random movement, shifting the permutation to a neighboring one using a random integer (randint).

In this example, the combined objective function value improves from 16.8 to 13.0, demonstrating the movement from the current position to the best firefly position with an objective function value of 13.0. This process repeats until the termination criterion, defined by the total number of generations, is satisfied.

Local search

A local search mechanism is employed to explore the neighborhood of a generated solution in search of better alternatives. Various neighborhood structures are proposed to enhance both the exploitation and exploration capabilities of the algorithm. These structures improve convergence towards near-optimal solutions while maintaining the diversity of the solution population. By making local moves from one solution to its neighboring solution, the search process effectively explores promising regions within the solution space. The following neighborhood structures are designed for the machine assignment and scheduling components:

Neighborhood Structure for Machine Assignment

Random Neighborhood in Machine Assignment (Lma1):
This neighborhood is generated using the following steps:

Select a position within the machine assignment component.

Randomly choose another machine from the available set for the corresponding operation at the selected position.

Critical Operations Neighborhood in Machine Assignment (Lma2):
This structure identifies the machine with the most critical operations. Some of these critical operations are then reassigned to another machine to potentially enhance the solution's quality.

Neighborhood Structure for Operation Scheduling

Critical Operation Swap Neighborhood (Lswap):
This neighborhood involves swapping operations along the critical path, following these rules:

If the first or last block has more than two operations, only the last two or the first two operations are swapped.

If the block contains only two operations, they are directly swapped.

No swaps occur if a block contains only one operation.

Critical Operation Insert Neighborhood (Linsert):
This neighborhood is formed through the following steps:

Randomly select a critical block with at least two critical operations.

Within each critical block, the first or last operation is inserted into the internal operations of the block.

Internal operations within the block are shifted to the beginning or the end of the block.

During the local search, the current schedule is replaced by a better neighboring solution, if one is found. A solution is considered improved if it meets one of the following criteria:

It has a lower fitness value compared to the initial solution.

It has the same fitness value as the initial solution but features fewer critical paths, provided there are multiple critical paths in the initial solution.

The primary benefit of hybridizing the Firefly Algorithm (FA) with local search is the accelerated convergence towards local optima. However, the downside is the increased computational time per generation. When the computational time is limited, the number of generations is reduced, which restricts the global search capability of genetic algorithms. Therefore, an essential challenge in the hybrid discrete firefly algorithm is effectively balancing the available computation time between the firefly solution search and the local search.

Framework of HDFA

The proposed HDFA could keep the balance of both the global exploration and local exploitation, and it also stresses the diversity of the population during the searching process. Thus, it is expected to achieve good performance in solving the multi-objective flexible job shop scheduling problem. The detailed steps of the proposed HDFA are as follows:

Step1: Set the parameter values;
//set the population size (P_{size});
//set the rate of machine assignment rules and scheduling component rules (R_{ma1}, R_{ma2}, R_{sc1}, R_{sc2});
//set the maximum number of generations (max_{gen});
//set the max local search iterations (ite_{max})
//set attractiveness of fireflies (β_0); light absorption coefficient (γ); randomization parameter (α)
Step 2: Initialize the firefly population stochastically according to the encoding scheme;
Step 3: Evaluate each firefly's objective value in the population and find the P_{best} the best firefly solution with the lowest fitness value.
Step 4: If the termination condition is not satisfied go to Step 5; otherwise go to Step 8
Step 5: Update the firefly solution using β-step and α-step;
Step 6: Start Local search procedure and replace the current solution with the better solution obtained by Local search;
Step 7: Evaluate the fireflies once again and update the best firefly solution using the whole population's experience; then go to Step 4
Step 8: Output computational results.

Table 4: PM tasks of 4 × 5-m

		PM_{11}	PM_{21}	PM_{31}	PM_{41}	PM_{51}
Time window	t_{kl}^E	0	0	0	1	4
	t_{kl}^L	4	6	6	5	10
Duration		1	2	2	1	2

Table 5: PM tasks of 8 × 8-m

		PM_{11}	PM_{21}	PM_{31}	PM_{41}	PM_{51}	PM_{61}	PM_{71}	PM_{81}
Time window	t_{kl}^E	1	3	5	6	0	3	1	3
	t_{kl}^L	10	9	15	17	10	16	14	13

Table 6: PM tasks of 10 × 10-m

		$PM_{1,1}$	$PM_{2,1}$	$PM_{3,1}$	$PM_{4,1}$	$PM_{5,1}$	$PM_{6,1}$	$PM_{7,1}$	$PM_{8,1}$	$PM_{9,1}$	$PM_{10,1}$
Time window	t_{kl}^E	0	1	0	0	2	0	0	0	0	0
	t_{kl}^L	4	7	6	5	7	6	5	7	5	6
Duration		2	1	1	2	1	2	2	3	2	3

COMPUTATIONAL RESULTS

This section describes the computational experiments used to evaluate the performance of the proposed algorithm. In order to conduct the experiment, we implement the algorithm in C++ on an Intel Core 2 Duo 2.0 GHz PC with 4 GB RAM memory.

Setting Parameters

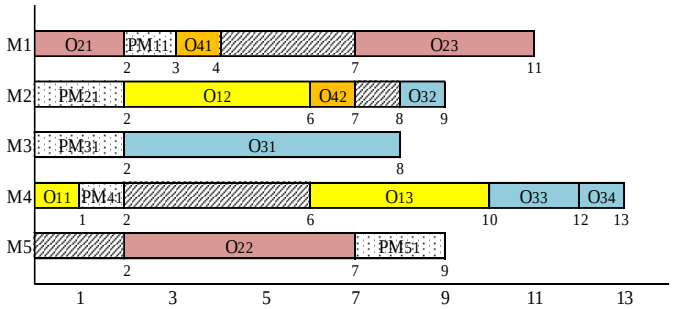
The parameters of the hybrid algorithm are as follows. Set the population size P_{size} = 200, rate of machine assignment rule R_{ma1} and scheduling component rule R_{sc1} = 20%, rate of machine assignment rule R_{ma2} and scheduling component rule R_{sc2} = 30%, the maximum number of generations max_{gen} = 100, maximum local search iteration ite_{max} = 50, attractiveness of fireflies β_0 = 1.0, light absorption coefficient γ = 0.1 and randomization parameter α = 1.0.

Comparison of the FJSP instances with PM Tasks

One representative instance with PM tasks (denoted by problem *n x m*-m) is taken from Rajkumar et al. [8] and the remaining two instances from Gao et al. [4]. Each instance sets one or two PM activities on each machine in the planning horizon. The non-fixed availability constraint is set as the same as in Gao et al. [4], and the time window of starting time and duration of maintenance tasks are shown in Tables 4-6 for 4 × 5-m, 8 × 8-m, and 10 × 10-m instance, respectively.

Table 7: Comparison of Results on FJSP instances with PM Tasks

Algorithm	4 × 5-m			8 × 8-m			10 × 10-m		
	C_m	W_{max}	W_t	C_m	W_{max}	W_t	C_m	W_{max}	W_t
hGA	-	-	-	17	15	105	8	7	61
GRASP	16	9	40	18	16	103	9	7	60
FBS-based algorithm	-	-	-	18	16	103	9	8	60
DCRO	-	-	-	17	15	105	8	7	61
	-	-	-	18	16	103	9	8	60
HDFA	13	9	40	17	15	105	8	7	61
	-	-	-	18	16	103	9	8	60



(C_m = 13, W_{max} = 9, W_t = 40)

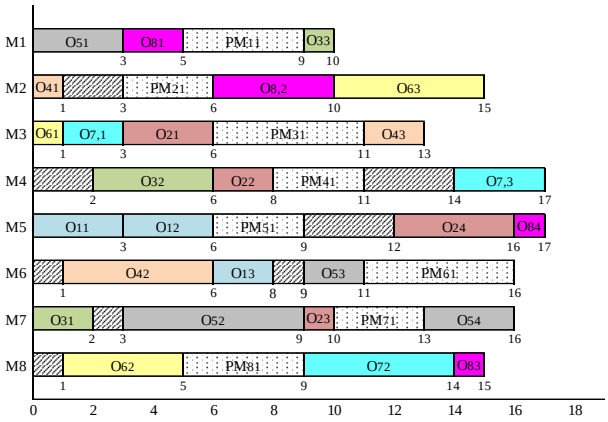


Fig. 6: Optimal Solution of the 8 × 8 instance (C_m = 17, W_{max} = 15, W_t = 105)

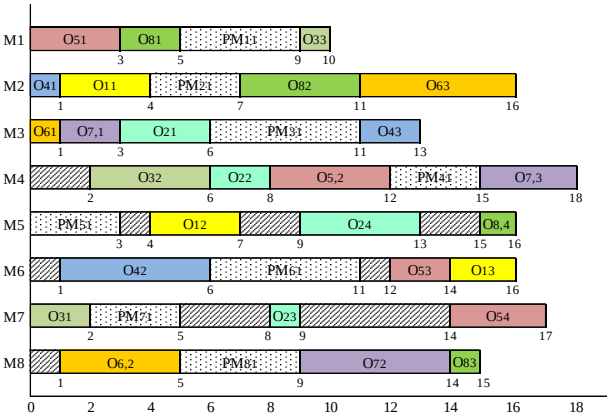


Fig. 6: Optimal Solution of the 8 × 8 instance (C_m = 18, W_{max} = 16, W_t = 103)

Table 7 shows the comparison of the results on the three FJSP instances with PM tasks with other four algorithms, i.e., the hGA in [4], GRASP in [8], the FBS based algorithm in [7] and the DCRO algorithm in [9]. The first column in Table 7 gives the comparison algorithms. The next three columns show the comparison results on the 4 × 5-m instance. The following three columns express the comparison results on the 8 × 8-m instance, while the last three columns display the comparison results on the 10 × 10-m instance. It can be seen from Table 7 that the make span value of the small scale instance 4 × 5-m is better than the GRASP algorithm. Then for the 8 × 8-m instance and the 10 × 10-m instance, the HDFA algorithm can obtain all the two non-dominated solutions, while the other algorithms except DCRO obtain only one optimal solution, respectively.

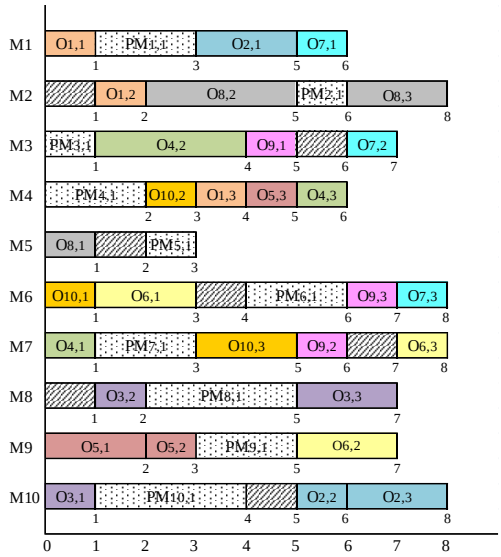


Fig. 7: Optimal Solution of the 10 × 10 instance (C_m = 8, W_{max} = 7, W_t = 61)

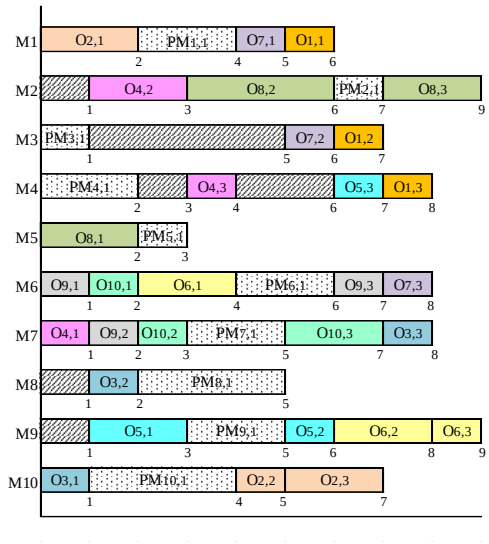


Fig. 8: Optimal Solution of the 10 × 10 instance (C_m = 9, W_{max} = 8, W_t = 60)

The optimal solution for the 4 × 5-m instance obtained by the HDFA algorithm are shown in Fig. 5. Fig. 6 and Fig. 7 display the obtained optimal solution for the 8 × 8-m by the proposed algorithm, while Fig. 7 and Fig.8 gives the scheduling result for the 10 × 10-m instance. In the figures the block marked with "PM" is the maintenance task to the corresponding machine. The hatched block represents the machine's idle periods.

CONCLUSIONS

This paper presents an effective Hybrid Discrete Firefly Algorithm (HDFA) designed to address the multi-objective flexible job shop scheduling problem with non-fixed availability constraints caused by maintenance activities. The objective is to minimize the makespan, maximal workload, and total workload of machines. Instead of using the standard firefly algorithm, a discrete version of

the continuous functions for distance, attractiveness, and movement is introduced to update the firefly positions. A specialized decoding method is applied to account for maintenance activities.

Additionally, two neighborhood structures related to machine assignment and operation sequencing are integrated into the algorithm to guide the local search toward more promising regions of the search space. The effectiveness of the proposed algorithm is validated by comparing its results with those reported in the literature for three representative instances. Future research will focus on extending the application of HDFA to solve other combinatorial optimization problems.

REFERENCES

I. M. R. Garey, D.S. Johnson and R. Sethi "The complexity of flowshop and jobshop scheduling". *Mathematics of Operations Research*, Vol.1, 1976, pp.117-129.

II. W. J. Xia and Z.M. Wu "An effective hybrid optimization approach for multi-objective flexible job shop scheduling problems". *Computers and Industrial Engineering*, Vol.48 No.2, 2005, pp.409-425.

III. G. Schmidt "Scheduling with limited machine availability". *European Journal of Operational Research*, Vol.121, No.1, 2000, pp.1-15.

IV. J. Gao, M. Gen, and L. Sun "Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm". *Journal of Intelligent Manufacturing*, Vol.17, No.4, 2006, pp.493-507.

V. Y. Ma, C.B. Chu and C.R. Zuo "A survey of scheduling with deterministic machine availability constraints". *Computers and Industrial Engineering*, Vol.58 No.2, 2010, pp.199-211.

VI. N. Zribi, A.E. Kamel, P.Borne "Minimising the makespan for the MPM job-shop with availability constraints". *International Journal of Production Economics*, Vol. 112, No.1, 2008, pp. 151-160.

VII. S.J. Wang and J.B. Yu "An effective heuristic for flexible job-shop scheduling problem with maintenance activities". *Computers and Industrial Engineering*, Vol.59 No.3, 2010, pp.436-447.

VIII. M. Rajkumar, P. Asokan and V. Vamsikrishna "A GRASP algorithm for flexible job-shop scheduling with maintenance constraints". *International Journal of Production Research*, Vol.48, No.22, 2010, pp.6821-6836.

IX. J.Q. Li, and Q. K. Pan "Chemical-reaction optimization for flexible job-shop scheduling problems with maintenance activity". *Applied Soft Computing*, Vol.12, No.9, 2012, pp.2896-2912.

X. E. Moradi, S.M.T. Fatemi Ghomi, and M. Zandieh "Bi-objective optimization research on integrated fixed time interval preventive maintenance and production for scheduling flexible job-shop problem". *Expert systems with applications*, Vol.38, No.6, 2011, pp.7169-7178.

XI. G.H. Graves, and C.Y. Lee "Scheduling maintenance and semiresumable jobs on a single machine". *Naval Research Logistics*, Vol.46, No.7, 1999, pp.845-863.

XII. R. Aggoune "Minimizing the makespan for the flow shop scheduling problem with availability constraints". *European Journal of Operational Research*, Vol.153, No.3, 2004, pp.534-543.

XIII. M.A. Kubzin and V.A. Strusevich "Planning machine maintenance in two-machine shop scheduling". *Operations Research*, Vol.54, No.4, 2006, pp.789-800.

XIV. X. S. Yang, " Nature-inspired Metaheuristic Algorithm", Luniver Press, 2008.

XV. X. S. Yang "Firefly algorithms for multimodal optimization", *Stochastic algorithms: Foundations and Applications*, Vol.5792, 2009, pp.169-178.

XVI. S. Lukasik and S. Zak, "Firefly algorithm for continuous constrained optimization tasks", *Lecture notes in Computer Science*, Vol.5796, 2009, pp.97-106.

XVII. M. Sayadi, R. Ramezanian and N. Ghaffari-Nasab "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems". *International Journal of Industrial Engineering Computations*, Vol.1, No.1, 2010, pp.1-10.

XVIII. G. K. Jati, "Evolutionary discrete firefly algorithm for travelling salesman problem", In *Adaptive and Intelligent Systems*, Springer Berlin Heidelberg, 2011, pp.393-403.

XIX. A. Khadwilard, S. Chansombat, T. Thepphakorn, W. Chainate and P. Pongcharoen "Application of firefly algorithm and its parameter setting for job shop scheduling". *The Journal of Industrial Technology*, Vol.8, No.1, 2012, pp.49-58.

XX. M.K. Marichelvam, T. Prabaharan and X.S. Yang "A Discrete firefly Algorithm for the Multi-objective Hybrid Flowshop Scheduling Problems". *IEEE Transactions on Evolutionary Computation*, 2012, No.99

XXI. F. Pezzella, G. Morganti and G. Ciaschetti "A genetic algorithm for the flexible job-shop scheduling problem", *Computers & Operations Research*, Vol.35, No.10, 2008, pp.3202-3212.

XXII. J. Li, Q. Pan and S. Xie “An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems”. *Applied Mathematics and Computation*, Vol.218, No.18, 2012, pp.9353-9371. P. Brandimarte, “Routing and scheduling in a flexible job shop by tabu search”. Annals of Operations Research, Vol.41, 1993, pp.157-183.