Distributed Flexible Jobshop Scheduling for Enterprise-Wide Model Lifecycle in Commercial Banks

1st Keyao Wang
BOC Postdoctoral Research Center
Bank of China
Beijing, China
wky9685@bank-of-china.com

2nd Dehui Wang
School of Mathematics and Statistics
Liaoning University
Shenyang, China
wangdehui@lnu.edu.cn

***Abstract*—The widely use of models in commercial banks presents significant resource scheduling challenges across model lifecycle, from development and validation to monitoring and exit. These models demand intensive computational and human resources, making consolidated resource allocation critical for effectively control the model risk associated with all models within an institution. These resources are typically distributed across multiple departments and locations for the purpose of disaster recovery redundancies. The expert skills are heterogeneous to maintain different types of models, and the lifecycle workflow varies for models at different risk levels. This paper addresses the optimization of the enterprise-wide model lifecycle by formulating it as a Bilayered Distributed Flexible Jobshop Scheduling Problem (BiDFJSP), capturing the distributed and heterogeneous machine environment, two-stage processing workflows and model maintainance time window constraints, aiming at minimizing the makespan of the first-stage implementation under mandatory second-stage maintenance time windows. Prevailing dispatching rules, constructive heuristics and metaheuristic algorithms are adopted to solve the proposed BiDFJSP. Experimental results demonstrate that the metaheuristics achieve makespan reduction over dispatching rules, offering a scalable solution for model risk management through optimized resource allocation.**

***Index Terms*—model lifecycle, distributed scheduling, flexible jobshop, metaheuristic algorithm, model risk management.**

I. Introduction

The widespread integration of artificial intelligence and machine learning (AI/ML) models within the operational and decision-making frameworks of commercial banks marks a significant technological shift [1]. As the volume and complexity of AI/ML models scale across a bank, the lifecycle workflow scheduling has transformed into a critical operational topic [2]. Since the raise of supervision and regulation letter SR 11-7 [3], both regulators [4] and the industry [5] have prompted to extend Model Risk Management (MRM) principles to address the multifaceted challenges introduced by AI/ML models, e.g., more complicated validation, more frequent monitoring, and more intensive resource requirement. Recent literature has mainly focused on MRM governance frameworks [6], validation techniques [7] and quantified model risk assessment [8], but neglected the treatment of resource scheduling and operational workflow optimization as a mechanism for enterprise-wide model risk control. Poorly coordinated workflows can lead to model backlogs and monitoring lapses, thereby amplifying model risk [9]. This work considers enterprise-wide workflows throughout the entire model lifecycle from model development to exit, as is shown in Fig. 1.

In practice, the resource for model lifecycle management is inherently distributed and heterogeneous. For reasons of business continuity and disaster recovery, computational resources are often dispersed across multiple data centers or departments [10]. The needed expert skills are diverse to build, validate, and monitor different types of models [11], e.g., internal credit rating models versus financial instrument valuation models. The mandated workflow, including two stages before and after the model implementation, varies significantly based on the model risk levels [6]. The combination of distributed resources, heterogeneous skillsets, and risk-dependent, two-stage workflows makes it a complex scheduling problem to optimize the enterprise-wide model lifecycle. Current practices often rely on manual coordination or simple expert rules, leading to proloned launch time and increased model risk [5].

To address this gap, the optimization of enterprise-wide model lifecycle is formulated as a two-stage variant of Distributed Flexible Jobshop Scheduling Problem (DFJSP) [12], denoted as Bilayered DFJSP (BiDFJSP), which is elaborated in Section II. Recent research has significantly extended the classic DFJSP to better reflect real-world complexities, e.g., multi-objective formulations integrating energy consumption [13], and dynamic machine breakdown [14]. However, DFJSP literature seldom addresses the distinct nature of enterprise-wide lifecycle workflows, including the precedence constraints that define the two-stage workflows, the mandated time windows for post-implementation maintenance at the second stage, and the objective to minimize the makespan of the first stage, i.e., the completion time of model implementation. The BiDFJSP is denoted as $D_F, FJ_c \,|\, r_{ij}, d_{ij} \,|\, C_{\max}$ according to the triplet for classifying scheduling problems [15].

The work was supported by grants from the China Postdoctoral Science Foundation (Grant No. 2025M773693).

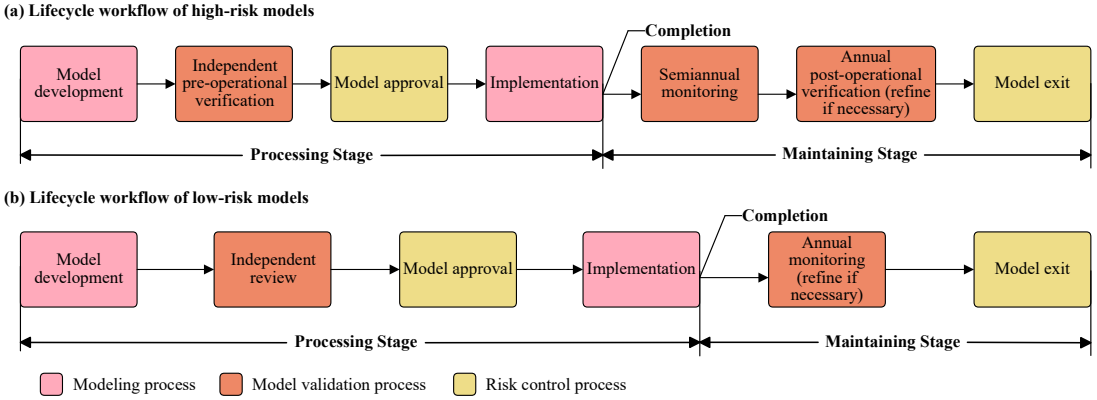


Fig. 1. Risk-tiered model lifecycle workflows.

To solve this NP-hard problem, we first employ a set of heuristics to represent the schedule in Section III. Four frequently used distpaching rules are adopted to solve BiDFJSP [15], as is described in Section V. We also adopt a constructive heuritic algorithm [16], and a set of metaheuristics, including several Simulated Annealing (SA) variants [17] and the Variable Neighborhood Descent (VND) [18]. Performance of these algorithms is comprehensively evaluated in Section VI.

The main contributions of this work are threefold. First, a novel BiDFJSP is proposed and effectively solved to depict the model lifecycle workflow scheduling problem. Second, we construct a comprehensive benchmark suite based on both real operational scenarios and stress ones, providing a testbed for model lifecycle scheduling. Third, we demonstrate the practical efficacy of our approach through experiments on benchmarks and real datasets, showing that the metaheuristics greatly reduce the makespan compared to baseline methods relying on common expert rules.

II. Distributed Flexible Jobshop Scheduling for Enterprise-Wide Model Lifecycle

A. Problem Description

Solving the BiDFJSP requires three key decisions, i.e., allocating jobs to suitable Flexible Manufacturing Units (FMUs), scheduling their operations to specific machines, and arranging the operation precedence on each machine. First, assign n jobs (models) to F distributed FMUs. Each job i has o_i operations ($i = 1, \cdots, n$), denoted as $O_{i1}, \cdots, O_{io_i}$, which must be processed from O_{i1} to O_{io_i}. Each FMU l ($l = 1, \cdots, F$) contains H machines, and the h-th machine in FMU l is denoted as m_{lh}. Second, assign each O_{ij} to a suitable m_{lh}. M_{ij}^l represents the set of machines in FMU l that can process operation O_{ij}. The processing time of O_{ij} on its eligible machine is denoted as p_{ij}^{lh}. Third, determine the processing sequence for all O_{ij} assigned to each m_{lh}.

In BiDFJSP, the following assumptions are made. Once a job is assigned to an FMU, all its operations in both stages must be processed within that FMU. The processing time is machine-dependent, and the same operation may have different p_{ij}^{lh} on different machines. Each machine can process only one operation at a time, and no preemption is allowed. Transportation time between machines, machine maintenance durations, and constraints related to labor, energy, and raw materials are not considered. Upon completion of implementation, a model is launched into use. The primary optimization objective is the makespan of the first stage, defined as the completion time of implementation. The second stage consists of mandatory maintenance operations that must be scheduled within their release and due dates; these constraints are enforced as hard constraints rather than as additional optimization objectives.

B. Mixed-Integer Programming Model

A mixed-integer programming model is established for the BiDFJSP with the objective of minimizing the makespan of the first stage. Table I explains the symbols used in the model.

The mixed-integer programming model for BiDFJSP is:

$$\min \max C_i. \tag{1}$$

Constraints include:

$$\sum_{l=1}^{F} X_i^l = 1, \forall i, \tag{2}$$

$$\sum_{h \in M_{ij}^l} Y_{ij}^{lh} = X_i^l, \forall i, j, l, \tag{3}$$

$$\sum_{l=1}^{F} \sum_{h \notin M_{ij}^l} Y_{ij}^{lh} = 0, \forall i, j, \tag{4}$$

$$\begin{aligned} Z_{ij,i'j'} + Z_{i'j',ij} = \sum_{l=1}^{F} \sum_{h=1}^{H_l} Y_{ij}^{lh} \cdot Y_{i'j'}^{lh}, \\ \forall i, i' \in J; j \in 1, \cdots, o_i, j' \in 1, \cdots, o_{i'}; \\ i \neq i' \text{ or } j \neq j', \end{aligned} \tag{5}$$

$$Z_{ij,ij'} = 0, \forall i \in J; j, j' \in 1, \cdots, o_i, j > j', \tag{6}$$

$$\sum_{h \in N_{ij}^l} K_{ij}^{lh} = X_i^l, \forall i, j, l, \tag{7}$$

$$\sum_{l=1}^{F} \sum_{h \notin N_{ij}^l} K_{ij}^{lh} = 0, \forall i, j, \tag{8}$$

$$\begin{aligned} L_{ij,i'j'} + L_{i'j',ij} = \sum_{l=1}^{F} \sum_{h=1}^{H_l} Y_{ij}^{lh} \cdot K_{i'j'}^{lh}, \\ \forall i, i' \in J; j \in 1, \cdots, o_i, j' \in 1, \cdots, o_{i'}; \\ i \neq i' \text{ or } j \neq j', \end{aligned} \tag{9}$$

TABLE I
SYMBOLS AND DEFINITIONS USED IN BIDFJSP

Index	Definition
l	Index of FMU, $l = 1, \cdots, F$
i	Index of job, $i = 1, \cdots, n$
j	Index of operation of job i, $j = 1, \cdots, o_i$ or q_i
h	Index of machine, $h = 1, \cdots, H_l$
Parameter	**Definition**
F	Number of FMUs
n	Total number of jobs
o_i	Total number of operations before implementation
q_i	Total number of operations after implementation
H_l	Total number of machines in FMU l
U	Set of FMUs, $U = \{1, \cdots, F\}$
m_{lh}	The h-th machine in FMU l
J	Set of jobs, $J = \{1, \cdots, n\}$
O_{ij}	The j-th operation of job i before deployment
Q_{ij}	The j-th operation of job i after deployment
M_{ij}^{l}	Set of machine indices in FMU l for O_{ij}
N_{ij}^{l}	Set of machine indices in FMU l for Q_{ij}
p_{ij}^{lh}	Processing time required for operation O_{ij} on m_{lh}
t_{ij}^{lh}	Processing time required for operation Q_{ij} on m_{lh}
Γ	A sufficiently large positive real number
r_{ij}	Release date of O_{ij} after deployment
d_{ij}	Due date of O_{ij} after deployment
State variable	**Definition**
E_{ij}	Start time of operation O_{ij}
S_{ij}	Start time of operation Q_{ij}
C_i	Completion time of the first stage of job i
Decision variable	**Definition**
X_i^l	Binary variable, equals 1 if job i is processed in FMU l, otherwise 0
Y_{ij}^{lh}	Binary variable, equals 1 if operation O_{ij} is processed on machine m_{lh}, otherwise 0
$Z_{ij,i'j'}$	Binary variable, equals 1 if operation $O_{i'j'}$ is processed after O_{ij}, otherwise 0
K_{ij}^{lh}	Binary variable, equals 1 if operation Q_{ij} is processed on machine m_{lh}, otherwise 0
$L_{ij,i'j'}$	Binary variable, equals 1 if operation $Q_{i'j'}$ is processed after Q_{ij}, otherwise 0

$$L_{ij,ij'} = 0, \forall i \in J; j, j' \in 1, \cdots, o_i, j > j', \tag{10}$$

$$E_{i1} \geq 0, \forall i, \tag{11}$$

$$E_{ij} \geq E_{i(j-1)} + \sum\nolimits_{h=1}^{H_l} p_{i(j-1)}^{lh} Y_{i(j-1)}^{lh}, \quad \forall i \in J, j \neq 1, l \in U \tag{12}$$

$$E_{ij'} \geq E_{ij} + p_{ij}^{lh} Y_{ij}^{lh} + \Gamma(Z_{ij,ij'} + Y_{ij}^{lh} + Y_{ij'}^{lh} - 3), \quad \forall i \in J; j, j' = 1, \cdots, o_i, j < j'; \quad l \in U; h \in M_{ij}^{l} \cap M_{ij'}^{l}, \tag{13}$$

$$E_{i'j'} \geq E_{ij} + p_{ij}^{lh} Y_{ij}^{lh} + \Gamma(Z_{ij,i'j'} + Y_{ij}^{lh} + Y_{i'j'}^{lh} - 3), \quad \forall i, i' \in J, i < i'; j = 1, \cdots, o_i, j' = 1, \cdots, o_{i'}; \quad l \in U; h \in M_{ij}^{l} \cap M_{i'j'}^{l}, \tag{14}$$

$$E_{ij} \geq E_{i'j'} + p_{i'j'}^{lh} Y_{i'j'}^{lh} + \Gamma(Y_{ij}^{lh} + Y_{i'j'}^{lh} - Z_{ij,i'j'} - 2), \quad \forall i, i' \in J, i < i'; j = 1, \cdots, o_i, j' = 1, \cdots, o_{i'}; \quad l \in U; h \in M_{ij}^{l} \cap M_{i'j'}^{l}, \tag{15}$$

$$C_i \geq E_{io_i} + \sum\nolimits_{h=1}^{H_l} p_{io_i}^{lh} Y_{io_i}^{lh}, \forall i, l, \tag{16}$$

$$S_{ij} \geq r_{ij} + C_i, \forall i, j, \tag{17}$$

$$S_{ij} \leq d_{ij} + C_i - t_{ij}^{lh} K_{ij}^{lh}, \forall i, j, l, h, \tag{18}$$

$$S_{ij} \geq S_{i(j-1)} + \sum\nolimits_{h=1}^{H_l} t_{i(j-1)}^{lh} K_{i(j-1)}^{lh}, \quad \forall i \in J, j \neq 1, l \in U \tag{19}$$

$$S_{ij'} \geq S_{ij} + t_{ij}^{lh} Y_{ij}^{lh} + \Gamma(L_{ij,ij'} + K_{ij}^{lh} + K_{ij'}^{lh} - 3), \quad \forall i \in J; j, j' = 1, \cdots, o_i, j < j'; \quad l \in U; h \in N_{ij}^{l} \cap N_{ij'}^{l}, \tag{20}$$

$$S_{i'j'} \geq S_{ij} + t_{ij}^{lh} K_{ij}^{lh} + \Gamma(L_{ij,i'j'} + K_{ij}^{lh} + K_{i'j'}^{lh} - 3), \quad \forall i, i' \in J, i < i'; j = 1, \cdots, o_i, j' = 1, \cdots, o_{i'}; \quad l \in U; h \in N_{ij}^{l} \cap N_{i'j'}^{l}, \tag{21}$$

$$S_{ij} \geq S_{i'j'} + t_{i'j'}^{lh} K_{i'j'}^{lh} + \Gamma(K_{ij}^{lh} + K_{i'j'}^{lh} - L_{ij,i'j'} - 2), \quad \forall i, i' \in J, i < i'; j = 1, \cdots, o_i, j' = 1, \cdots, o_{i'}; \quad l \in U; h \in N_{ij}^{l} \cap N_{i'j'}^{l}, \tag{22}$$

$$X_i^l \in \{0, 1\}, \forall i, l, \tag{23}$$

$$Y_{ij}^{lh} \in \{0, 1\}, \forall i, j, l, h, \tag{24}$$

$$Z_{ij,i'j'} \in \{0, 1\}, \forall i, i', j, j', \tag{25}$$

$$K_{ij}^{lh} \in \{0, 1\}, \forall i, j, l, h, \tag{26}$$

$$L_{ij,i'j'} \in \{0, 1\}, \forall i, i', j, j', \tag{27}$$

where (1) represents the scheduling objective of minimizing the makespan of the first stage. Equation (2) indicates that each job i can only be assigned to one FMU. Equations (3) and (4) state that O_{ij} can only be assigned to one eligible machine $m_{lh} \in M_{ij}^{l}$. Equation (5) indicates that two operations processed on the same machine must satisfy a precedence order. Equation (6) states that different operations of the same job must satisfy the predecessor-successor relationship. Equations (7)-(10) states the constraints of the decision variables of the second stage, which are the same as those of the first stage. Equation (11) indicates that all jobs can start processing from time 0. Equation (12) states that the interval between the start times of two adjacent operations $O_{i(j-1)}$ and O_{ij} is not less than the processing time of $O_{i(j-1)}$ on m_{lh}. Equation (13) indicates that when two operations of the same job are processed on the same machine, start time of successor $O_{ij'}$ is not earlier than start time of predecessor O_{ij}, and the time interval is not less than the processing time of O_{ij} on m_{lh}. Equation (14) indicates that when O_{ij} and $O_{i'j'}$ from different jobs are processed on the same m_{lh}, if O_{ij} is processed before $O_{i'j'}$, the start time of $O_{i'j'}$ is not earlier than that of O_{ij}, and the time interval is not less than processing time of the former O_{ij} on m_{lh}; otherwise, if $O_{i'j'}$ is processed before O_{ij}, Equation (15) should be satisfied. Equation (16) states that the completion time of job i should not be less than the sum of the start time and processing time of its last operation O_{io_i}. Equation (17) states the start time of operations in the second stage is no earlier than the release date counting from the completion time of the fist stage, and (18) states this start time no later than a due date. Equations (17) and (18) states that Q_{ij}s are subject to hard time window constraints, which

must be satisfied for a schedule to be feasible. Equations (19)-(22) states the start time of the second stage follows the same logic as the first stage. Equations (23)-(27) indicate that the decision variables are all 0-1 variables.

III. Scheduling Solution Encoding and Decoding Strategies Based on Problem Characteristics

Inspired by the S_{OP} encoding and heuristic rule-based decoding strategy [19], we propose encoding and decoding strategies for the BiDFJSP, ensuring solution feasibility and algorithm performance. A BiDFJSP example in Table II is designed to demonstrate the S_{OP} encoding and decoding strategy. The first two columns of Table II show the jobs and the operations in each job. Columns 3 and 4 show the processing time of each operation on different machines, where "-" indicates that the operation cannot be processed on that machine. Columns 5 and 6 show the release date and due date of each operations in the second stage.

TABLE II
An Example of BiDFJSP

Jobs	Operations	Processing time m_{11} & m_{21}	Processing time m_{12} & m_{22}	Release date	Due date
Job 1	O_{11}	1	2	-	-
	O_{12}	4	3	-	-
	O_{13}	2	-	-	-
	Q_{11}	1	2	2	4
	Q_{12}	2	1	4	6
Job 2	O_{21}	3	1	-	-
	O_{22}	2	3	-	-
	Q_{21}	2	3	2	4
	Q_{22}	3	1	4	6
Job 3	O_{31}	-	1	-	-
	O_{32}	6	3	-	-
	Q_{31}	2	1	2	5

A. Operation Sequence Encoding Based on Job Vector

The job vector is adopted as the encoding of the solution for BiDFJSP [19], denoted as S_{OP}. Each element in S_{OP} represents a job index, and the occurrence order of the same job index indicates the operation indices sequentially corresponding to that job. An example of S_{OP} encoding for this problem is shown in the upper part of Fig. 2. Jobs 1, 2, and 3 have 3, 2, and 2 operations in the first stage, and 2, 2, and 1 operation(s) in the second stage, respectively, so S_{OP} contains five "1"s, four "2"s, and three "3"s. The operation sequence corresponding to this S_{OP} is $O_{11} \rightarrow O_{21} \rightarrow O_{12} \rightarrow O_{31} \rightarrow O_{22} \rightarrow O_{13} \rightarrow Q_{11} \rightarrow Q_{21} \rightarrow Q_{22} \rightarrow O_{32} \rightarrow Q_{31} \rightarrow Q_{12}$.

S_{OP} only encodes the operation sequence decision of BiDFJSP, while the job-factory assignment decision and operation-machine assignment decision will be obtained by decoding S_{OP} using two heuristic rules described below.

B. Decoding: Job-Factory Assignmente

Given an S_{OP}, this paper proposes a heuristic rule based on minimum average processing time to assign the jobs to FMUs. The heuristic rule mainly includes the following three steps.

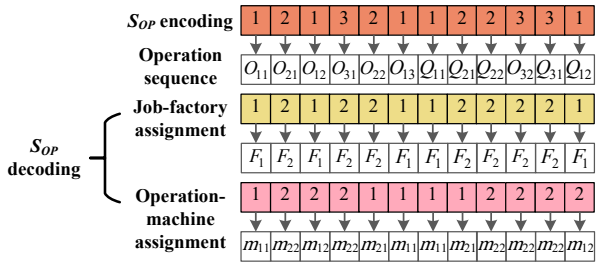


Fig. 2. An illustration of job-factory and operation-machine assignments.

Step 1. The order in which different job numbers first appear in S_{OP} is taken as the order for assigning jobs to FMUs;

Step 2. Calculate the average processing time of each job i in each FMU l as $p_{li} = \sum_{j=1}^{o_i} \sum_{h=1}^{|M_{ij}^l|} p_{ij}^{lh}/|M_{ij}^l| + \sum_{j=1}^{q_i} \sum_{h=1}^{|M_{ij}^l|} t_{ij}^{lh}/|N_{ij}^l|$, where $|M_{ij}^l|$ (or $|N_{ij}^l|$) denotes the number of elements in set M_{ij}^l (or N_{ij}^l), i.e., the number of eligible machines for operation O_{ij} (or Q_{ij}) in FMU l;

Step 3. According to the order in Step 1, find the job i to be assigned. Define its expected completion time in FMU l as $p_{li} + \sum_{i' \in L_l} p_{li'}$, where L_l represents the set of jobs already assigned to FMU l, and assign job i to the FMU with the smallest expected completion time. If job i has equal expected completion times in multiple FMUs, select the FMU l with the smallest average processing time p_{li}; if multiple FMUs have the same p_{li}, randomly select an FMU l.

Using the S_{OP} example from Section II-A, the job-factory assignment decision for the problem shown in Table II is demonstrated. First, the order of the first appearance of the three job numbers in S_{OP}, $1 \rightarrow 2 \rightarrow 3$, is taken as the order for assigning jobs to FMUs. Then, calculate the average processing time of each job in each FMU: $p_{11} = 10$, $p_{12} = 9$, $p_{13} = 7$, $p_{21} = 10$, $p_{22} = 9$, $p_{23} = 7$; job 1 has equal p_{li} in both FMUs and assigned to FMU 1 selected randomly; job 2 has expected completion time of 19 and 9 in the two FMUs, and assigned to FMU 2 with smaller expected completion time; job 3 is also assigned to FMU 2 with the smaller expected completion time. The assignment result is job $1 \rightarrow$ FMU 1, job $2 \rightarrow$ FMU 2, job $3 \rightarrow$ FMU 2, as is shown in Fig. 2, where F_l in the figure represents FMU l.

C. Decoding: Operation-Machine Assignment

Given an S_{OP}, this paper proposes another heuristic rule based on minimum completion time to assign the operations to machines with the following three steps.

Step 1. Extract all operations in both stages of the jobs assigned to each FMU, and sort these operations according to their order in the S_{OP} encoding, obtaining the operation subsequence L_{op}^l for each FMU l;

Step 2. For each FMU l, assign operations to machines in the order of L_{op}^l. When assigning operation O_{ij}, calculate its completion time on each eligible machine m_{lh} in FMU l as $C_{ij}^{lh} = \max\left\{C_{i'j'}^{lh}, C_{i(j-1)}^{lh'}\right\} + p_{ij}^{lh}$, where $C_{i'j'}^{lh}$ denotes the completion time of the previous operation $O_{i'j'}$

(or $Q_{i'(j'-o_i)}$) on machine m_{lh}, and $C^{lh'}_{i(j-1)}$ denotes the completion time of the predecessor operation $O_{i(j-1)}$ of job i on its assigned machine $m_{lh'}$. As for Q_{ij}, its completion time $C^{lh}_{i(j+o_i)} = \max\left\{C^{lh}_{i'j'}, C^{lh'}_{i(j+o_i-1)}, C_i + r_{ij}\right\} + t^{lh}_{ij}$, where $C^{lh}_{i'j'}$ denotes the completion time of the previous operation $O_{i'j'}$ (or $Q_{i'(j'-o_i)}$) on machine m_{lh}, $C^{lh'}_{i(j+o_i-1)}$ denotes the completion time of the predecessor operation O_{io_i} for $j = 1$ or $Q_{i(j-1)}$ for others on its assigned machine $m_{lh'}$, and C_i is defined in Table I, which is the completion time of O_{io_i}.

Step 3. Assign O_{ij} and Q_{ij} to the machine with the smallest completion time. If there are multiple machines with the same smallest completion time, record these schemes, and assign subsequent operations based on each scheme. Select the scheme that minimizes the completion time of subsequent operations. If there are some assignment schemes with equal completion time after assigning all operations, select the scheme that minimizes the total completion time in that FMU. If there are multiple such schemes, randomly select one.

The operation-machine assignment decision on FMU 2 is demonstrated using the S_{op} example from Section II-A. From Section III-B, jobs 2 and 3 are assigned to FMU 2. From Step 1, L^2_{op} is $O_{21} \to O_{31} \to O_{22} \to Q_{21} \to Q_{22} \to O_{32} \to Q_{31}$. Table III shows the assignment of operations in L^2_{op} to machines in FMU 2 according to Steps 2 and 3.

TABLE III
AN ILLUSTRATION OF OPERATION-MACHINE ASSIGNMENT ON FMU 2

L^2_{op}	C^{lh}_{ij} of tentative scheme		Selected machine	Updated C^{lh}_{ij} of the last operation		Available date
	m_{21}	m_{22}		m_{21}	m_{22}	
O_{21}	3	1	m_{22}	0	1	0
O_{31}	-	2	m_{22}	0	2	0
O_{22}	3	5	m_{21}	3	2	0
Q_{21}	7	8	m_{21}	7	2	5
Q_{22}	10	8	m_{22}	7	8	7
O_{32}	13	11	m_{22}	7	11	0
Q_{31}	15	14	m_{22}	7	14	13

Similarly, the assignment result on FMU 1 is: $O_{11} \to m_{11}$, $O_{12} \to m_{12}$, $O_{13} \to m_{11}$, $Q_{11} \to m_{11}$, $Q_{12} \to m_{12}$.

D. Decoding: Operation Processing Sequence

After finishing the operation-machine assignment, the processing sequence of operations assigned to each machine can be determined according to the S_{OP} encoding. Through these methods, a feasible scheduling for BiDFJSP can be decoded from S_{OP}. The gantt chart of the feasible solution decoded from the S_{OP} in Fig. 2 is shown in Fig. 3.

IV. BENCHMARK DESIGN

A. Benchmark instances for BiDFJSP

As there are no standard benchmark for BiDFJSP, we construct 10 benchmark instances from the standard DFJSP benchmark la01-10 [12], denoted as Bila01-10. In these instances, we set $F = 2$, $H_l = 5$ for $\forall l$, $n = 10$, $o_i = q_i = 5$ for $\forall i$. p^{lh}_{ij} and t^{lh}_{ij} are randomly generated from $\{1, 2, \cdots, 99\}$ for $\forall i, j$. r_{ij} and d_{ij} are randomly generated from $\{1, 2, \cdots, 99\}+100\cdot(j-1)$ and $\{1, 2, \cdots, 99\}+100\cdot p_i \cdot j$ for $\forall i$. Processing time, release and due dates of each operation are the same on all eligible machines. The eligible machines for O_{ij} and Q_{ij} are the same, i.e., $M^l_{ij} = N^l_{ij}$ for $\forall i, j, l$.

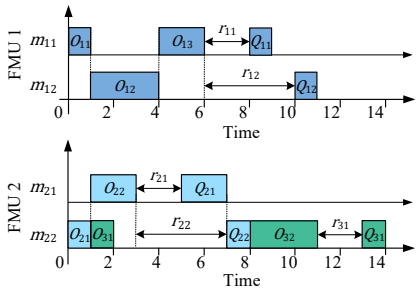


Fig. 3. Gantt chart of the feasible solution decoded from S_{OP} in Fig. 2.

B. Real case of Enterprise-Wide Model Lifecycle Workflow

We further propose a realistic case to develop the BiDFJSP within practical operational context, involving heterogeneous resources distributed across locations for disaster recovery, and workflows vary based on model risk tiering.

In the first stage from model development to implementation, the workflow of models differs based on their risk level. For high-risk models, the sequence is shown in Fig.1(a) as

$$O_{i1}\ (Development) \to O_{i2}\ (Independent\ Verification) \to O_{i3}\ (Model\ Approval) \to O_{i4}\ (Implementation).$$

For low-risk models, the sequence is shown in Fig.1(b) as

$$O_{i1}\ (Development) \to O_{i2}\ (Independent\ Review) \to O_{i3}\ (Model\ Approval) \to O_{i4}\ (Implementation).$$

If a model is purchased from a vendor rather than built in house, its development operation is omitted, and the workflow begins from the verification/review. Set the processing time $p^{lh}_{ij} \in \{n \in \mathbb{Z} \mid 60 \le n \le 90\}$ days for high-risk models, $p^{lh}_{ij} \in \{n \in \mathbb{Z} \mid 15 \le n \le 45\}$ days for low-risk models, and all p^{lh}_{ij} of O_{ij} are kept the same for all eligible machines in any FMU. The eligible machine set M^l_{ij} is randomly generated and kept the same for all FMUs, $|M^l_{ij}| \in \{1, 2, 3\}$.

In the second stage, all models are assumed to have the same service life as 3 years. Models require periodic maintenance operations, and time windows are determined based on the risk level. High-risk models require semiannual monitoring and annual ongoing verification, resulting in 6 post-deployment operations. Set the release date $\{r_{ij}\}^6_{j=1} = \{6, 12, 18, 24, 30, 36\}$ months, the due date $\{d_{ij}\}^6_{j=1} = \{9, 15, 21, 27, 33, 39\}$ months, the processing time $t^{lh}_{ij} \in \{n \in \mathbb{Z} \mid 40 \le n \le 60\}$ days for any high-risk model and $\forall j, l, h$. Low-risk models require annual monitoring, i.e., 3 operations in the second stage. We set $\{r_{ij}\}^3_{j=1} = \{12, 24, 36\}$ months, $\{d_{ij}\}^3_{j=1} = \{18, 30, 42\}$ months, $t^{lh}_{ij} \in \{n \in \mathbb{Z} \mid 20 \le n \le 40\}$ days. The eligible machine set N^l_{ij} is randomly generated, $|N^l_{ij}| \in \{1, 2, 3\}$, and N^l_{ij} are kept the same for all FMUs. Suppose 1 month has 30 days.

The resource environment consists of three identical FMUs at distinct locations, reflecting a disaster recovery setup. Three resilience scenarios are considered: the base scenario with all FMUs fully operational, a moderate stress scenario with two available FMUs, and a severe stress scenario with only one FMU accessible. Each FMU has 5 machines. 3 in house high-risk models, 3 in house low-risk models, 2 purchased high-risk models, and 2 pruchased low-risk models are to be processed.

V. Algorithms

We compare three categories of algorithms to solve the proposed BiDFJSP: (i) baseline dispatching rules commonly used in practice, (ii) a constructive heuristic, and (iii) meta-heuristic algorithms. The novelty consists in adaptations of the standard methods to the BiDFJSP using the proposed S_{OP}-based representation and the associated decoding heuristics.

Dispatching Rules. Four dispatching rules are adopted from classic scheduling theory [15] and adapted to BiDFJSP using the decoding methods proposed in Section III. These rules serve as baselines representing prevailing tools in practice. The Shortest Processing Time first (SPT) and Longest Processing Time first (LPT) rules sort jobs based on the minimum average processing time across all FMUs, defined as $\min_{l\in U} p_{li}$, where p_{li} is described in Section III-B. In SPT, jobs are processed in ascending order of this metric; LPT follows the opposite order. Incorporating the Earliest Due Date first (EDD), the SPT and LPT are extended as the SPT-EDD and LPT-EDD: first-stage operations are scheduled using SPT or LPT, and second-stage operations are subsequently sorted in ascending order of their due dates. For all dispatching rules, the S_{OP} encoding is generated according to the sorted job order, and the decoding strategies from Sections III-B and III-C are applied to obtain a complete schedule. To illustrate the proposed solution framework, we provide the pseudocode for the LPT-EDD in Algorithm 1 as an example.

Algorithm 1: LPT-EDD for BiDFJSP.

Data: Instance data: J, U, p_{ij}^{lh}, t_{ij}^{lh}, r_{ij}, and d_{ij}.
Output: Schedule and its makespan.
// Phase 1: First-stage ordering
for *each job* $i \in J$ *and each FMU* $l \in U$ **do**
  Compute p_{li} as defined in Section III-B;
end
Sort jobs in descending order of $\min_{l\in U} p_{li}$. Get a job list L_{LPT};
Generate partial solution S_{OP}^{1st} by concatenating operations in the order of L_{LPT}. All operations of Job i have higher precedence than those of Job i' if $\min_{l\in U} p_{li} < \min_{l\in U} p_{li'}$;
Apply heuristics in Section III to obtain C_i, $\forall i \in J$;
// Phase 2: Second-stage ordering
for *each operation* Q_{ij} **do**
  Determine due dates $D_{ij} = C_i + d_{ij}$;
end
Sort $Q_{ij}, \forall i \in J$ and $j \in q_i$ in ascending order of D_{ij} to obtain an operation list L_{EDD};
Append L_{EDD} after S_{OP}^{1st} to obtain S_{OP};
Apply heuristics in Section III to obtain the schedule, and check whether the schedule is feasible.

Constructive Heuristic. The modified Nawaz-Enscore-Ham (NEH) algorithm [16] is employed. It generates five random S_{OP} sequences as initial candidates, decodes each using the proposed decoding strategies, and selects the solution with the minimal makespan as its final output.

Metaheuristics. Three variants of Simulated Annealing (SA) are implemented using standard neighborhood operators: SA-INV (inverse), SA-INS (insert), and SA-SWP (swap) [17]. Additionally, a Variable Neighborhood Descent (VND) [18] is implemented. For these metaheuristics, the initial solution is set as the best solution obtained from the dispatching rules and the modified NEH. All metaheuristics adopt the S_{OP} encoding and the decoding strategies proposed in this paper. The other parameter settings of these metaheuristics follow those in [20].

VI. Experiments

A. Experiment settings

To ensure fairness in comparative experiments, all algorithms are implemented in MATLAB R2020a and run on a laptop equipped with Intel Core i7 with a 1.10 GHz processor and 16 GB of RAM. The solutions of dispatching rules are deterministic and there is no need for repeated experiments. Each metaheuristic is independently run for 10 times on each instance. The termination condition for metaheuristics is when the CPU time reaches $6no$ seconds, where n is the number of jobs and o is the maximum number of operations.

B. Results on benchmark

The Average Relative Error (ARE) of dispatching rules and meta-heuristic algorithms to solve the benchmark instances are summarized in Table IV, with the best results highlighted in bold. The calculation for ARE of algorithm f on l-th instance is $ARE_{l,f} = \frac{1}{I}\sum_{i=1}^{I}\left(V_{l,i}^{f} - V_{l}^{*}\right)/V_{l}^{*}$, where $V_{l,i}^{f}$ represents the makespan obtained by algorithm f in its i-th independent run on instance l, V_{l}^{*} denotes the makespan of the best known solution for instance l, and I is the number of independent runs per algorithm on each instance.

From Table IV, metaheuristics collectively outperformed dispatching rules, as evidenced by their lower average ARE by 318.1%, and the constructive heuristic was superior than dispatching rules with lower average ARE by 278.4%.

Among dispatching rules, LPT-EDD yielded the best average result. However, its applicability was limited, as it failed to generate feasible solutions for Bila06-09. LPT, while effective for smaller instances, also suffered from infeasibility on Bila07-10. SPT and SPT-EDD showed higher AREs. The results highlighted a limitation of dispatching rules.

Among meta-heuristics, VND demonstrated the best overall performance across all benchmark instances. The three SAs also performed well, with SA-SWP and SA-INV showing strong competitiveness. To sum up, the experiment validated the significant advantage of employing meta-heuristic algorithms, particularly VND to solve the BiDFJSP.

TABLE IV
ARE OF ALGORITHMS ON BIDFJSP BENCHMARK INSTANCES.

Instance	Dispatching rules				Constructive	Metaheuristic algorithms			
	SPT	SPT-EDD	LPT	LPT-EDD	NEH	SA-INV	SA-INS	SA-SWP	VND
Bila01	3.930	3.199	1.010	0.659	0.404	0.053	0.108	**0.000**	**0.000**
Bila02	4.043	4.805	0.599	0.708	0.307	**0.000**	0.041	0.061	**0.000**
Bila03	5.241	4.198	1.269	0.630	0.126	0.017	0.088	0.024	**0.000**
Bila04	4.748	4.390	0.672	0.537	0.558	**0.000**	0.077	0.024	**0.000**
Bila05	4.145	6.153	0.866	0.997	0.029	0.042	**0.000**	0.028	**0.000**
Bila06	7.336	6.867	1.243	-[a]	1.209	0.069	0.274	0.156	**0.057**
Bila07	5.987	8.797	-	-	0.737	0.216	0.347	0.142	**0.106**
Bila08	6.130	6.711	-	-	0.498	0.136	0.332	0.106	**0.097**
Bila09	5.484	6.913	-	-	0.573	0.129	0.317	0.215	**0.114**
Bila10	8.345	7.038	-	1.117	0.582	0.324	0.354	0.093	**0.066**
Average	5.539	5.907	0.943	**0.753**	0.502	0.099	0.194	0.085	**0.044**
	328.6%				50.2%	10.5%			

[a]"-" represents that the dispatching rule cannot generate a feasible solution.

TABLE V
MAKESPAN (IN DAYS) OF CASE STUDY.

Stress scenarios	Dispatching rules				Constructive	Metaheuristic algorithms (Average of 10 repetitions)			
	SPT	SPT-EDD	LPT	LPT-EDD	NEH	SA-INV	SA-INS	SA-SWP	VND
Base	-[a]	420	-	**324**	-	**324**	**324**	**324**	**324**
Moderate	-	**461**	-	-	-	**324**	391.2	**324**	**324**
Severe	-	**692**	-	-	-	534.8	605.5	**499**	516.2

[a]"-" represents that the algorithm cannot generate a feasible solution.

C. Case Study

We further tested the algorithms on the designed real cases. The same algorithms were compared across three scenarios. The performance metric was the makespan. Results were summarized in Table V, and the best-found schedules for stress scenarios were visualized in Fig. 4.

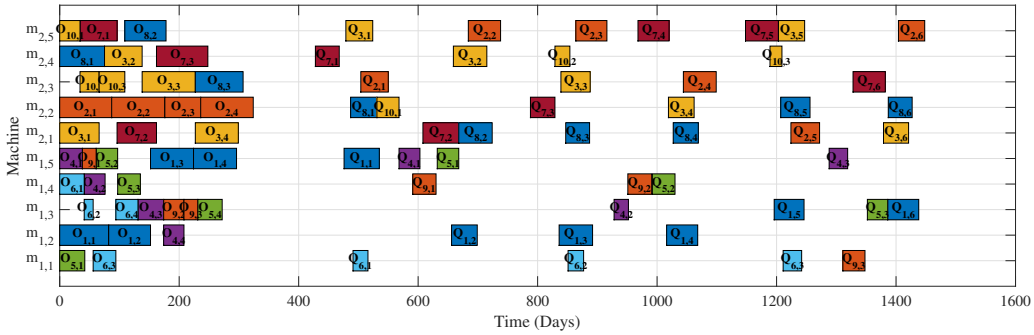
The experimental findings revealed several key insights. First, the performance of metaheuristics matched or surpassed that of dispatching rules in all scenarios. In the base scenario with 3 FMUs, all metaheuristics and LPT-EDD achieved the best-so-far makespan of 324 days. However, as resource constraints tightened, the limitations of static rules became apparent. In the moderate stress scenario, only SPT-EDD found a feasible schedule, while all metaheuristics significantly improved upon this, with VND, SA-INV, and SA-SWP achieving the same best-so-far makespan of 324 days. In the most challenging severe stress scenario with only 1 FMU available, SPT-EDD yielded a makespan of 692 days, whereas SA-SWP performed the best among all algorithms, with a thrift of average makespan by 27.9% compared with SPT-EDD.

The Gantt charts in Fig. 4 visualized the scheduling effectiveness. Fig. 4(a) showed that if one FMU was invalid, the metaheuristics could manage to rebalance the load and find a compact schedule, with the makespan unchanged compared with the base scenario. Fig. 4(b) illustrated the schedule with only one FMU left, where SA-SWP found the best-so-far solution with a makespan of 459 days. Despite the severe resource bottleneck, the algorithm sequenced operations to meet maintenance deadlines, demonstrating the practical value of scheduling for business continuity planning. In addition, the increase in makespan was a quantitative index to evaluate the contribution of resource supplements, which provided a decision-making basis to balance investments in disaster recovery resources against cost saving objectives. This case study confirmed that metaheuristics was superior than static dispatching rules under disaster recovery situations. The reduced makespan represented for faster model implementation, improved resource allocation, and enhanced risk control.

VII. CONCLUSIONS

This paper addressed the challenge of resource scheduling for the enterprise-wide model lifecycle in commercial banks by proposing a novel BiDFJSP. This formulation effectively captured the distributed and flexible resource environment, risk-tiered two-stage workflows, and mandatory post-operational maintenance time windows. The objective is defined as minimizing the makespan of the first-stage model implementation. We designed tailored encoding and decoding strategies to represent the scheduling solutions, and evaluated the performance of multiple algorithms to solve BiDFJSP. Experiments demonstrated that metaheuristics outperformed dispatching rules commonly adopted in practice. The designed benchmark and real case for BiDFJSP could serve as standard testbed for future research.

(a) The best-so-far solution in the moderate stress scenario with two FMUs available



(b) The best-so-far solution in the severe stress scenario with one FMU available

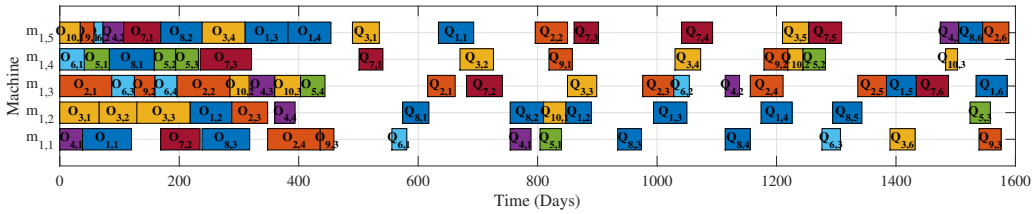


Fig. 4. Gantt chart of the feasible solutions in different stress scenarios.

Several promising directions remain for future work. First, more realistic constraints can be incorporated with BiDFJSP, e.g., machine setup time and transportation time. The model can also be extended to multi-objective optimization. Second, intelligent optimization methods can be adopted to solve the BiDFJSP. Third, banks can consider integrating the proposed scheduling engine with existing MRM platforms.

ACKNOWLEDGMENT

The first author would like to express special thanks to Professor Bo Liu at the Chinese Academy of Sciences, whose insights on distributed scheduling have been greatly beneficial. Gratitude is also extended to Vice President Hong Shi, Director Guangli Hu, and Senior Manager Yang Liu of the Risk Management Department at the Bank of China for their valuable advice on model risk management.

REFERENCES

[1] Monetary Authority of Singapore, "Artificial intelligence (ai) model risk management," Monetary Authority of Singapore, Information Paper, 2024, monographs/Information Papers series. [Online]. Available: https://www.mas.gov.sg/publications/monographs-or-information-paper/2024/artificial-intelligence-model-risk-management

[2] W. Sun, M. Li, X. H. Chen, and Y. Wang, "Enhancing exchange rate prediction and risk management under uncertainty shocks: An ai-driven ensemble prediction model based on metaheuristic optimization," *Ann Oper Res*, 2024.

[3] Board of Governors of the Federal Reserve System, "Guidance on model risk management," Federal Reserve System, Washington, D.C., SR Letter SR 11-7, 2011.

[4] Bank of England Prudential Regulation Authority, "Model risk management principles for banks," Bank of England, Supervisory Statement SS1/23, 2023.

[5] L. Silotto, M. Scaringi, and M. Bianchetti, "Xva modelling: Validation, performance and model risk management," *Ann Oper Res*, vol. 336, no. 1-2, pp. 183–274, 2024.

[6] N. Kiritz, M. Ravitz, and M. Levonian, "Model risk tiering: An exploration of industry practices and principles," *Journal of Risk Model Validation*, vol. 13, no. 2, pp. 47–77, 2019.

[7] M. Cummins, F. Gogolin, F. Kearney, G. Kiely, and B. Murphy, "Practice-relevant model validation: Distributional parameter risk analysis in financial model risk management," *Ann Oper Res*, vol. 330, no. 1-2, pp. 431–455, 2023.

[8] J. Kerkhof, B. Melenberg, and H. Schumacher, "Model risk and capital reserves," *Journal of Banking & Finance*, vol. 34, no. 1, pp. 267–279, 2010.

[9] M. Hu, Y. Zhang, X. Feng, and X. Xiong, "How technological innovation influence operational risk: Evidence from banks in china," *International Review of Financial Analysis*, vol. 95, p. 103480, 2024.

[10] D. Wu and D. L. Olson, "Enterprise risk management: Coping with model risk in a large bank," *Journal of the Operational Research Society*, vol. 61, no. 2, pp. 179–190, 2010.

[11] T. Berg and P. Koziol, "An analysis of the consistency of banks' internal ratings," *Journal of Banking & Finance*, vol. 78, pp. 27–41, 2017.

[12] L. De Giovanni and F. Pezzella, "An improved genetic algorithm for the distributed and flexible job-shop scheduling problem," *European Journal of Operational Research*, vol. 200, no. 2, pp. 395–408, 2010.

[13] Z.-Q. Zhang, Y. Li, B. Qian, R. Hu, and J.-B. Yang, "A multidimensional probabilistic model based evolutionary algorithm for the energy-efficient distributed flexible job-shop scheduling problem," *Engineering Applications of Artificial Intelligence*, vol. 135, p. 108841, 2024.

[14] K. Zhu, G. Gong, N. Peng, L. Zhang, D. Huang, Q. Luo, and X. Li, "Dynamic distributed flexible job-shop scheduling problem considering operation inspection," *Expert Systems with Applications*, vol. 224, p. 119840, 2023.

[15] M. L. Pinedo, *Scheduling*. Cham: Springer International Publishing, 2016.

[16] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the M-machine, N-job flowshop sequencing problem," *Omega-International Journal of Management Science*, vol. 11, no. 1, pp. 91–95, 1983.

[17] A. Franzin and T. Stützle, "A landscape-based analysis of fixed temperature and simulated annealing," *European Journal of Operational Research*, p. S0377221722003125, 2022.

[18] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[19] M.-C. Wu, C.-S. Lin, C.-H. Lin, and C.-F. Chen, "Effects of different chromosome representations in developing genetic algorithms to solve dfjs scheduling problems," *Computers & Operations Research*, vol. 80, pp. 101–112, 2017.

[20] K. Wang and B. Liu, "A syntactic problem solver learning landscape structures for clinical scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 4, pp. 1313–1327, 2025.