Hindawi
Complexity
Volume 2019, Article ID 5734149, 14 pages
https://doi.org/10.1155/2019/5734149

Research Article

A Novel MOEA/D for Multiobjective Scheduling of Flexible Manufacturing Systems

Xinnian Wang,[1] Keyi Xing,[1] Chao-Bo Yan,[1] and Mengchu Zhou[2]

[1]*State Key Laboratory for Manufacturing Systems Engineering, and Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, China*
[2]*Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, USA*

Correspondence should be addressed to Keyi Xing; kyxing@mail.xjtu.edu.cn

Received 22 January 2019; Accepted 7 May 2019; Published 2 June 2019

Guest Editor: Rosario Domingo

This paper considers the multiobjective scheduling of flexible manufacturing systems (FMSs). Due to high degrees of route flexibility and resource sharing, deadlocks often exhibit in FMSs. Manufacturing tasks cannot be finished if any deadlock appears. For solving such problem, this work develops a deadlock-free multiobjective evolutionary algorithm based on decomposition (DMOEA/D). It intends to minimize three objective functions, i.e., makespan, mean flow time, and mean tardiness time. The proposed algorithm can decompose a multiobjective scheduling problem into a certain number of scalar subproblems and solves all the subproblems in a single run. A type of a discrete differential evolution (DDE) algorithm is also developed for solving each subproblem. The mutation operator of the proposed DDE is based on the hamming distance of two randomly selected solutions, while the crossover operator is based on Generalization of Order Crossover. Experimental results demonstrate that the proposed DMOEA/D can significantly outperform a Pareto domination-based algorithm DNSGA-II for both 2-objective and 3-objective problems on the studied FMSs.

1. Introduction

The *multiobjective optimization problem* (MOP, the abbreviations and their meanings are given in Table 1) is an optimization problem that may have a number of conflicting objectives to be considered, and decision-makers need to determine an optimal trade-off among the objectives. This problem presents in many real-life applications [1–4]. A *Pareto optimal* solution to an MOP is a candidate for the optimal trade-off [5]. Most MOPs have a lot of or even an infinite number of Pareto optimal solutions, and the set of all the Pareto optimal solutions in the objective space is called as *Pareto front* (PF). The decision-makers desire a fair approximation to the PF to make final decisions. *Multiobjective evolutionary algorithms* (MOEAs) work with populations of candidate solutions to MOPs and therefore are able to find good approximations to the PF in a single run.

To produce Pareto optimal vectors that are distributed evenly along the PF and therefore can approximate the PF, many MOEAs evaluate the solutions based on Pareto domination like SPEA2 [6] and NSGA-II [7]. Multiobjective evolutionary algorithm based on decomposition (MOEA/D) [8], which decomposes an MOP into a number of single-objective subproblems, is another effective MOEA framework. In an MOEA/D, the objective to each subproblem is a weighted aggregation of some individual objectives. Based on the distances between the weighted aggregation vectors, the neighborhood relationships among the subproblems can be found out. Each subproblem is then solved according to the information mainly from the neighboring subproblems. There are a few significant studies that apply it in dealing with multiobjective problems in different areas. Chen et al. [9] proposed a multiobjective discrete method called MODTLBO/D for solving the community detection problem of complex networks. They adopted a multiobjective decomposition method and introduced a neighbor-based mutation. For multiobjective job shop scheduling problem, Zhao et al. [10] developed an improved multi-objective evolutionary algorithm based on decomposition (IMOEA/D). Experimental results demonstrate that their IMOEA/D can converge better than Pareto dominance-based MOEAs.

Table 1: Descriptions of abbreviations.

Abbreviation	Description	Abbreviation	Description
MOP	Multi-objective optimization problem	DE	Differential evolution
PF	Pareto front	DDE	Discrete differential evolution
MOEA	Multi-objective evolutionary algorithm	GOX	Generalization of order crossover
SPEA2	Improving the strength Pareto evolutionary algorithm	PNS	Petri net for scheduling
NSGA-II	Nondominated sorting genetic algorithm II	DAP	Deadlock avoidance policy
MOEA/D	MOEA based on decomposition	EP	External population
MODTLBO/D	Multi-objective discrete teaching-learning-based optimization with decomposition	S^3PRs	Systems of simple sequential processes with resources
IMOEA/D	Improved MOEA/D	DNSGA-II	Deadlock-free NSGA-II
FMS	Flexible manufacturing system	*NPS*	Number of Pareto solutions
AGV	Automated guided vehicle	*MID*	Mean ideal distance
PN	Petri net	*RAS*	Rate of achievement to objectives simultaneously

A flexible manufacturing system (FMS) is a computer-controlled manufacturing system that is comprised of finite resources and can process multitypes of jobs. Different from the traditional production environments such as job shops and flow shops, FMSs often encounter deadlock problems due to high degrees of route flexibility and resource sharing. Once a deadlock appears, the whole or partial system will be indefinitely blocked and cannot finish manufacturing tasks. Developing effective control and scheduling approaches to avoid deadlocks while optimizing the performance of the system is of paramount importance in practice.

Scheduling of FMSs contains both deadlock control and optimization of objectives and therefore is more difficult. A few works have been published on this area [4, 11–23]. Most of them involve deadlock problems and some also concerns different optimization objectives such as the back-tracking and distance travel of AGVs [4], AGV's fleet size [11], and total energy consumption [21, 22]. However, none of them takes multiobjective optimization into consideration.

This work addresses the multiobjective scheduling problem of deadlock-prone FMSs for the first time and proposes a deadlock-free multiobjective evolutionary algorithm based on decomposition (DMOEA/D) and Petri net (PN) models. Our DMOEA/D can decompose a multiobjective FMS scheduling problem into several single-objective subproblems and optimizes all the subproblems in a single run by evolving a population of solutions. In each generation, a solution for solving a subproblem is reproduced by a new proposed discrete differential evolution (DDE) algorithm. In DDE, the mutation operator is based on the hamming distance between two randomly selected solutions. The Generalization of Order Crossover (GOX) [24] is used as the crossover operator. Two illustrated examples are used to demonstrate the efficiency of the proposed algorithm. Since there is no work reported for the multiobjective scheduling of FMSs considering deadlock situations, we just compare our proposed DMOEA/D with a NSGA-II based MOEA on these examples.

This work presents an effective approach for organizations and production managers to enhance their competitiveness in manufacturing versatile, route-flexible, and time-critical productions concerning two or more different scheduling objectives.

The rest of the paper is organized as follows. Section 2 introduces the FMSs studied in the paper and their Petri net models and defines the considered multiobjective scheduling problem. Section 3 introduces some basics of MOEA/D and DDE and develops a multiobjective scheduling algorithm based on them. A performance comparison between two developed scheduling algorithms is made, and the experimental results are show in Section 4. Section 5 concludes the paper.

2. PN Models of FMSs

PN is a widely used mathematical tool for modeling the dynamic behaviors of FMSs and many other manufacturing systems [15–17, 25–27]. This section briefly introduces some basics of PNs first and then the PN models of FMSs for multiobjective scheduling. For more details of PNs, readers may refer to [28].

2.1. Definitions of Petri Nets. Let $\mathbb{Z}_0 = \{0, 1, 2, \ldots\}$ and $\mathbb{Z}_k = \{1, 2, \ldots, k\}$. A marked PN is a 4-tuple $N = (P, T, F, M_0)$, where P is a finite set of places, T is a finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is the set of directed arcs, and $M_0 : P \longrightarrow \mathbb{Z}_0$ is the initial marking of N. For a given node $x \in P \cup T$, its preset is defined as ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$ and postset $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$. Given a marking M and a place $p \in P$, denote $M(p)$ as the number of tokens in p at M. A string $\alpha = x_1 x_2 \cdots x_k$ is called a *path* in the PN, where $x_i \in P \cup T$ and $(x_i, x_{i+1}) \in F$, $i \in \mathbb{Z}_{k-1}$.

A transition $t \in T$ is enabled at M if $\forall p \in {}^\bullet t$, $M(p) > 0$, denoted as $M[t >$. An enabled transition t can fire at M, yielding M', denoted as $M[t > M'$, where $M'(p) = M(p) - 1$, $\forall p \in {}^\bullet t \backslash t^\bullet$, $M'(p) = M(p) + 1$, $\forall p \in t^\bullet \backslash {}^\bullet t$, and otherwise $M'(p) = M(p)$. A sequence of transitions $\alpha = t_1 t_2 \cdots t_k$ is feasible from M if $M_i[t_i > M_{i+1}$ holds for $i \in \mathbb{Z}_k$, where $M_1 = M$.

2.2. Placed Timed PN Models of FMSs. An FMS studied in our work is comprised of m types of resources, denoted as $R = \{r_i, i \in \mathbb{Z}_m\}$, and is capable to process n types of jobs, denoted as $Q = \{q_j, j \in \mathbb{Z}_n\}$. More definitions and constraints are described as follows:

(1) The number of type-q_j jobs to be processed is $\varphi(q_j)$, and the total number of jobs is $\Phi = \sum_{j \in \mathbb{Z}_n} \varphi(q_j)$.

(2) The capacity of type-r_i resources is denoted as $C(r_i)$, which is a positive integer and marks the maximum jobs that type r_i resources can handle simultaneously.

(3) A processing route of a type-q_j job, $w_k = o_{k1} o_{k2} \cdots o_{kl(w_k)}$, is a predefined sequence of operations, where $l(w_k)$ is the total number of operations in route w_k and o_{kl} is the lth operation in w_k. A job may be processed on more than one route and can choose its routes while processing. Let $\Omega = \{w_k \mid k \in \mathbb{Z}_{|\Omega|}\}$ be the set of all processing routes and $\Omega_j \subseteq \Omega$ be the set of routes for type-q_j jobs, respectively.

(4) Each operation needs one unit resource, and any two consecutive operations of a job need different resource types. For operation o_{kl}, let $R(o_{kl})$ denote the resource needed for processing o_{kl}.

(5) The processing time $d(o_{kl})$ for operation o_{kl} is predefined. Let $c(o_{kl})$ denote the completion time of o_{kl}.

(6) No pre-emption is allowed.

Now we can establish a PN model for FMS considered in this paper.

For type-q_j jobs, let o_{js} and o_{je} be two virtual operations representing the storages of raw and processed type-q_j jobs, respectively. Operations o_{js} and o_{je} do not need any resource. Then, route w_k with these two fictitious operations for type-q_j jobs can be defined as $w_k = o_{js} o_{k1} o_{k2} \cdots o_{kl(w_k)} o_{je}$. Identical operations shared by different routes are merged as one operation.

A processing route $w_k = o_{js} o_{k1} o_{k2} \cdots o_{kl(w_k)} o_{je}$ in the PN model is modelled as a path of places and transitions $a_k = p_{js} t_{k1} p_{k1} t_{k2} p_{k2} \cdots t_{kl} p_{kl} \cdots t_{kl(w_k)} p_{kl(w_k)} t_{k(l(w_k)+1)} p_{je}$, where p_{js} and p_{je} represent operations o_{js} and o_{je}, respectively, and p_{kl} is an operation place that represents operation o_{kl}; t_{kl} is a transition that indicates the start of o_{kl} and the completion of $o_{k(l-1)}$. A token in operation place p_{kl} means that operation o_{kl} of a job is being processed. In these ways, the marked PN model of processing routes for type-q_j jobs is defined as

$$N_j = \left(P_j \cup \left\{p_{js}, p_{je}\right\}, T_j, F_j, M_{j0}\right) \tag{1}$$

where $P_j = \cup_{1 \le k \le |\Omega_j|} \{p_{k1}, p_{k2}, \ldots, p_{kl(w_k)}\}$, $T_j = \cup_{1 \le k \le |\Omega_j|} \{t_{k1}, t_{k2}, \ldots, t_{k(l(w_k)+1)}\}$, and $F_j = \cup_{1 \le k \le |\Omega_j|} \{(p_{js}, t_{k1}), (t_{k1}, p_{k1}), (p_{k1}, t_{k2}), \ldots, (p_{kl(w_k)}, t_{k(l(w_k)+1)}), (t_{k(l(w_k)+1)}, p_{je})\}$. M_{j0} is the initial marking, where $M_{j0}(p_{js}) = \varphi(q_j)$ and $M_{j0}(p) = 0$, $\forall p \in P_j \cup \{p_{je}\}$. In N_j, $\forall t \in T_j$, $|t^\bullet| = |{}^\bullet t| = 1$. An operation place p is a split place if $p \in P_j$, $|p^\bullet| > 1$. At a split place, a job is able to choose the processing routes.

For type-r_i resources, assign a resource place denoted also by r_i. Tokens in r_i represent the number of available type-r_i resources. Let $C(r_i)$ denote the initial marking of r_i.

Denote $R(p)$ and P_R as the resource needed by operation place p and the set of all resource places, respectively. Then, in our PN model, add arcs from $R(p)$ to each transition in ${}^\bullet p$ denoting the occupation of $R(p)$, and add arcs from each transition in $p^\bullet$ to $R(p)$ denoting the releasing of $R(p)$. The set of all arcs related to resource places is denoted as F_R. Then, the marked PN model of our studied FMS can be defined as

$$N = \left(P \cup P_s \cup P_f \cup P_R, T, F, M_0\right) \tag{2}$$

where $P = \cup_{j \in \mathbb{Z}_n} P_j$, $P_s = \{p_{js} \mid j \in \mathbb{Z}_n\}$, $P_f = \{p_{je} \mid j \in \mathbb{Z}_n\}$, $T = \cup_{j \in \mathbb{Z}_n} T_j$, $F = F_Q \cup F_R$, and $F_Q = \cup_{j \in \mathbb{Z}_n} F_j$. The initial marking M_0 is defined as $M_0(p_{js}) = \varphi(q_j)$, $\forall p_{js} \in P_s$; $M_0(p) = 0$, $\forall p \in P \cup P_f$; and $M_0(r_i) = C(r_i)$, $\forall r_i \in P_R$.

In this paper, processing times needed by operations are described by a place-timed PN. Each operation place p is assigned with a time delay $d(p)$, denoting its processing time, $d(p) = 0$, $\forall p \in P_s \cup P_f \cup P_R$. Such PN model for an FMS is called PNS [17].

Example 1. Consider an FMS shown in Figure 1. It contains four machines r_1, r_2, r_3, and r_4. Machines r_1, r_2, and r_4 can hold one job at the same time while machine r_3 can hold two. Then the resource set is $R = \{r_1, r_2, r_3, r_4\}$, with initial markings $C(r_1) = C(r_2) = C(r_4) = 1$, and $C(r_3) = 2$. This FMS is able to process two types of jobs, q_1 and q_2. Type-q_1 jobs can be processed sequentially on r_1, r_2, and r_4, or on r_1, r_3, and r_4; while type-q_2 jobs are processed on r_4, r_3, and r_1. Thus, there are two processing routes for type-q_1 jobs, $w_1 = o_{1s} o_{11} o_{12} o_{13} p_{1e}$ and $w_2 = o_{1s} o_{21} o_{22} o_{23} o_{1e}$, while type-$q_2$ jobs are processed on one route, $w_3 = o_{2s} o_{31} o_{32} o_{33} o_{2e}$. Note that $o_{11} = o_{21}$ and $o_{13} = o_{23}$ are processed on machines r_1 and r_4, respectively. Then routes w_1 and w_2 can be modelled as $w_1 = p_{1s} t_{11} p_{11} t_{12} p_{12} t_{13} p_{13} t_{14} p_{1e}$ and $w_2 = p_{1s} t_{11} p_{11} t_{22} p_{22} t_{23} p_{13} t_{14} p_{1e}$, respectively, while route w_3 is modelled as $w_3 = p_{2s} t_{31} p_{31} t_{32} p_{32} t_{33} p_{33} t_{34} p_{2e}$. Figure 2(a) shows the PN model of this system, in which the jobs of types q_1 and q_2 are to be processed are 2 and 1, respectively.

When all operations of all jobs are completed, the system reaches its final marking, denoted as M_f, where $M_f(p_{ie}) = M_0(p_{js})$ and $\forall p_{je} \in P_f$; $M_f(p) = 0$, $\forall p \in P \cup P_s$; and $M_f(r_i) = C(r_i)$, $\forall r_i \in P_R$. A sequence of transitions α is complete and feasible if $M_0[a > M_f$. Such a sequence in the PNS is a *feasible schedule* to the considered FMS.

Note that an improper schedule of a PNS may lead to deadlock situations, and the system can thus never reach its final marking. Consider a reachable marking $M_1 = 2p_{22} + p_{31} + r_1 + r_2$ in Figure 2(b). It is clear that a circular wait for resources r_3 and r_4 arises and no transition can be fired.

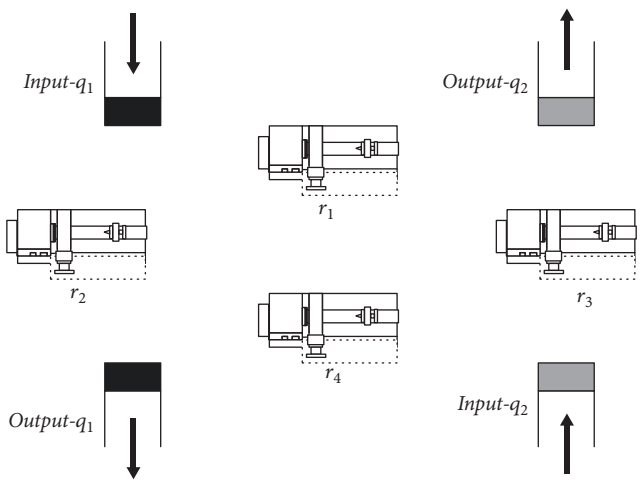

Figure 1: The FMS in Example 1.

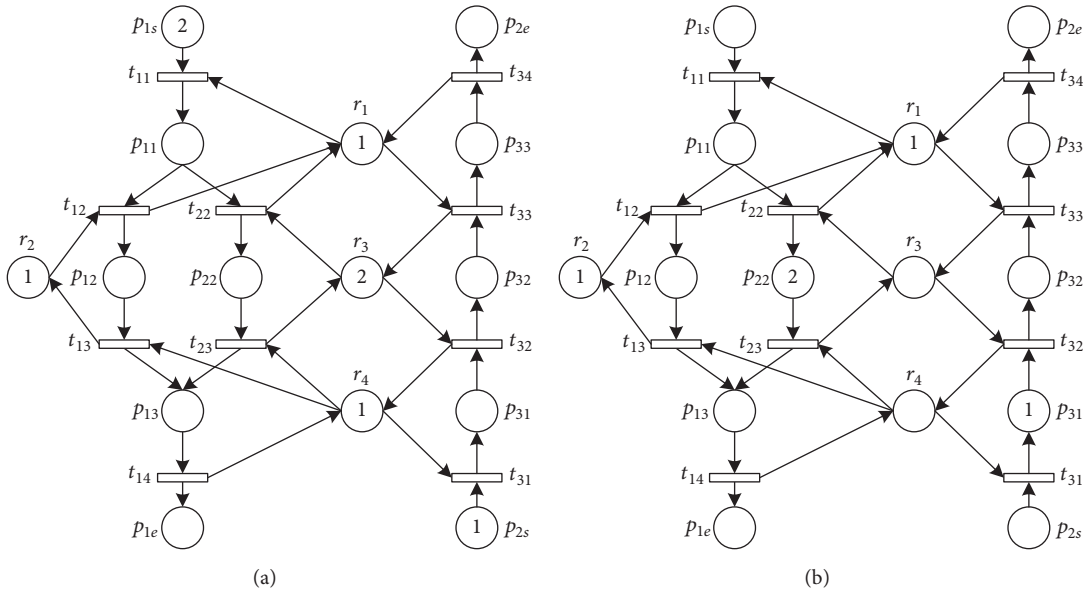

Figure 2: (a) The PNS of an FMS and (b) its deadlock situation.

3. DMOEA/D for the Scheduling of FMSs

A multiobjective optimization problem (MOP) is formally stated as follows:

$$\begin{aligned} &\text{Minimize} \quad F(x) = \left(f_1(x), f_2(x), \ldots, f_L(x)\right)^T \\ &\text{Subject to} \quad x \in \Pi \end{aligned} \tag{3}$$

where Π is the decision (variable) space and $f_i(x)$ is the objective functions, $i \in \mathbb{Z}_L$. A feasible *solution* to MOP can be interpreted into a feasible schedule. Then, the multiobjective scheduling problem studied in this paper is to find a feasible solution x^* so that $F(x^*)$ is as small as possible.

MOEA/D solves an MOP problem by decomposing it into several single-objective optimization subproblems and then solves all the subproblems in a single run by evolving a population of solutions [8]. The population in each generation is composed of best found solutions for these subproblems. The neighbourhood relations among the subproblems are calculated according to the distances between their coefficient vectors. In MOEA/D, each subproblem is then solved based on the information of its neighbouring subproblems.

There are a few studies using MOEA/D in different areas recently [5, 8–10, 29–31]. In this work, we focus on the multiobjective scheduling of FMSs and propose a novel DMOEA/D by combining MOEA/D and DDE. PNSs are used to model the systems and deadlock controllers are embedded to avoid deadlocks. The varieties of DMOEA/D are described as follows.

3.1. Representation, Interpretation, and Reparation. In this paper, a solution x in DMOEA/D consists of two sections $x = (S_r; S_o)$, where S_r stores the route information and S_o is a permutation with repetition of all jobs. Let $l(w_s)$ denote

the length of route w_s. Then, each job J_i appears $l(J_i)$ times in S_o, where $l(J_i) = \max\{l(w_s) \mid w_s \text{ is a route of job } J_i\}$. The jth operation of J_i is represented as the jth J_i in S_o, and S_o is uniquely interpreted as a sequence of operations $\sigma(S_o)$. Then x is rewritten as $x = (S_r; \sigma(S_o))$. By associating each operation to its corresponding transition, a sequence of transitions $\alpha(x)$, which is taken as a schedule in PNS, can be interpreted from solution x.

Example 2. Reconsider the PNS in Figure 1(a). There are three jobs to be processed: two type-q_1 jobs, J_1 and J_2, and one type-q_2 job, J_3. The type-q_1 jobs have two routes, w_1 and w_2, while type-q_2 job has only one, w_3. Suppose that J_1 and J_2 are processed by routes w_1 and w_2, respectively. Then, $S_{r1} = (w_1, w_2, w_3)$ can be taken as the first section of a solution x_1. Note that the route lengths $l(J_i)$ are 5, 5, and 4, respectively. Thus, we can represent section S_o as a permutation with repetition that consists of five J_1's, five J_2's, and four J_3's, for example, $S_{o1} = (J_1, J_1, J_3, J_2, J_3, J_3, J_1, J_2, J_1, J_2, J_2, J_2, J_1, J_3)$. Then x_1 is represented as $x_1 = (S_{r1}; S_{o1}) = (w_1, w_2, w_3; J_1, J_1, J_3, J_2, J_3, J_3, J_1, J_2, J_1, J_2, J_2, J_2, J_1, J_3)$, S_{o1} is interpreted as a sequence of operations $\sigma(S_{o1}) = (O_{11}, O_{12}, O_{31}, O_{21}, O_{32}, O_{33}, O_{13}, O_{22}, O_{14}, O_{23}, O_{24}, O_{25}, O_{15}, O_{34})$, and x_1 is interpreted as a sequence of transitions $\alpha(x_1) = (t_{11}t_{12}, t_{31}, t_{11}, t_{32}, t_{33}, t_{13}, t_{22}, t_{14}, t_{23}, t_{24}, t_{15}, t_{15}, t_{34})$, or for more details, $\alpha(x_1) = (t_{11}[O_{11}], t_{12}[O_{12}], t_{31}[O_{31}], t_{11}[O_{21}], t_{32}[O_{32}], t_{33}[O_{33}], t_{13}[O_{13}], t_{22}[O_{22}], t_{14}[O_{14}], t_{23}[O_{23}], t_{24}[O_{24}], t_{15}[O_{25}], t_{15}[O_{15}], t_{34}[O_{34}])$.

The sequence of transitions interpreted from a solution may be infeasible and contain deadlock situations. The feasibility of each solution must be examined and the infeasible ones are repaired. There are some works addressing the deadlock problem of FMSs [32–36], and deadlock-free operations of FMSs can thus be guaranteed. In this work, an Amending Algorithm proposed by [17] is used to obtain a feasible sequence of transitions from M_0 to M_f. This Amending Algorithm is based on the deadlock avoidance policy (DAP) proposed by [34]. An optimal polynomial complexity DAP for S^3PRs without ξ-resources is developed by a one-step look-ahead approach; for S^3PRs with ξ-resources, a suboptimal DAP is also derived by reducing the net. A deadlock search algorithm is then proposed for prohibiting deadlock situations.

3.2. Objective Functions. Three objective functions are used in the work, makespan, mean completion time, and mean tardiness time.

For a given solution x, let $\alpha(x) = t_1 t_2 \cdots t_n$ be its corresponding sequence of transitions. Let $f(t_k[O_{ij}])$ be the firing time of transition t_k and C_i be the completion time of the last operation of job J_i. Note that $f(t_k[O_{ij}])$ is also the start time of operation O_{ij}. According to the predefined operation sequence and the order in a schedule (sequence of transitions), transition $t_k[O_{ij}]$ ($k \in \mathbb{Z}_n$) can be fired only after transition t_{k-1} is fired and operation $O_{i(j-1)}$ is completed. Assume that t_{k-1} corresponds to operation O_{uv} and t_u corresponds to operation $O_{i(j-1)}$. We have $f(t_k[O_{ij}]) = \max\{f(t_{k-1}[O_{uv}]), f(t_u[O_{i(j-1)}]) + d(O_{i(j-1)})\}$, and the completion time of job J_i can be calculated as $C_i = f(t_v[O_{il(J_i)}]) + d(O_{il(J_i)})$, where $O_{il(J_i)}$ is the last processed operation of job J_i and t_v corresponds to $O_{il(J_i)}$.

The makespan C_{max} is the completion time of the last job that leaves the system. A minimum C_{max} implies a good utilization of machines. It can be defined as

$$f_1(x) \equiv C_{max} = \max_{1 \le i \le \Phi} \{C_i\} \tag{4}$$

The mean completion time is the average completion time of all the jobs, and its definition is

$$f_2(x) \equiv \overline{C} = \sum_{1 \le i \le \Phi} \frac{C_i}{\Phi} \tag{5}$$

Tardiness time of a job is a time delay after the predefined due time. Jobs completed after the due time may cause compensation for customers and loss of goodwill. Let D_i and T_i be the due time and the tardiness time of job J_i, respectively. In this paper, D_i is defined as $D_i = 1.5 \times \sum_{1 \le j \le l(J_i)} d(O_{ij})$ and T_i is defined as $T_i = \max\{C_i - D_i, 0\}$. The mean tardiness time is the average tardiness time of all the jobs, and it can be expressed as

$$f_3(x) \equiv \overline{T} = \sum_{1 \le i \le \Phi} \frac{T_i}{\Phi} \tag{6}$$

3.3. Decomposition of Multiobjective Optimization. A general MOEA/D must decompose an MOP into K single-objective optimization problems, where K is the population size. Several approaches have been proposed for this task and this paper uses the Tchebycheff approach.

Given a weight vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_L)^T$, $\lambda_i \ge 0$, $\forall i \in \mathbb{Z}_L$, and $\sum_{i \in \mathbb{Z}_L} \lambda_i = 1$. Then, in the Tchebycheff approach, the optimal solution to the single-objective optimization problem below

$$\begin{aligned} &\text{Minimize} \quad g^{te}(x \mid \lambda) = \max_{i \in \mathbb{Z}_L} \{\lambda_i (f_i(x) - z_i^*)\}^T \\ &\text{Subject to} \quad x \in \Pi \end{aligned} \tag{7}$$

is a Pareto optimal solution to (3), where $z^* = (z_1^*, z_2^*, \ldots, z_L^*)^T$ is the reference point, i.e., $z_i^* = \min_{x \in \Pi} f_i(x), \forall i \in \mathbb{Z}_L$. For each optimal solution of (7), there exists a corresponding Pareto optimal solution of (3). Hence, we can obtain different Pareto optimal solutions by setting different weight vectors.

3.4. Discrete Differential Evolution Algorithm. Differential evolution (DE) [37] generally works on a population of candidate solutions, which are represented by floating point numbers. DE generates a mutated solution by adding a weighted difference between two randomly selected solutions to a third one. A trial solution is then generated by crossing the mutated and target solutions. The selection operator in DE determines whether the target solution can be retained in the next generation or not.

From the description above, it can be known that the traditional DE cannot be directly used in the multiobjective

scheduling of FMSs, since it is originally designed for solving continuous optimization problems and can only generate solutions of floating point numbers [38]. In this subsection, we propose a novel discrete DE (DDE) for generating the solutions to the single-objective optimization problems in our DMOEA/D.

The mutated solution at the kth generation x_k^v is constructed based on 3 randomly selected solutions, x_k^a, x_k^b, and x_k^c. Let D denote the hamming distance between solutions x_k^b and x_k^c. In our DDE, the mutation operator is defined as follows:

$$x_k^v = \begin{cases} Mut_1\left(x_k^a, p_m\right), & \text{if } D = 0 \\ Mut_D\left(x_k^a, p_m\right), & \text{if } D > 0 \end{cases} \tag{8}$$

where Mut_i is the mutation with i iterations and p_m is the mutation probability. At each iteration of $Mut_i(x_k^a, p_m)$, the algorithm generates a random number $r \in [0, 1]$. If $r < p_m$, select one job in x_k^a randomly and insert it to another position in x_k^a; otherwise, x_k^a remains unchanged. Note that the mutation operator is iterated at least once, even if there is no difference between x_k^b and x_k^c.

The trial solution x_k^u in DDE algorithm is generated by crossing the target solution x_k^i and the mutated solution x_k^v. GOX is used for achieving this. In our DDE, GOX is only applied in the second section of the solution, i.e., S_o. The length of the crossover string is chosen randomly between $N/4$ and $3N/4$.

Example 3. Assume that $x_1 = (S_r^1; S_o^1) = (w_1, w_2, w_3; J_1, J_1, J_3, J_2, J_3, J_3, J_1, J_2, J_1, J_2, J_2, J_2, J_1, J_3)$ is a target solution and $x_2 = (S_r^2; S_o^2) = (w_1, w_1, w_3; J_1, J_2, J_2, J_1, J_3, J_1, J_2, J_3, J_2, J_1, J_3, J_1, J_2, J_3)$ is a mutated solution. Then, according to Section 3.1, their corresponding sequences of operations are $\sigma(S_o^1) = (O_{11}, O_{12}, O_{31}, O_{21}, O_{32}, O_{33}, O_{13}, O_{22}, O_{14}, O_{23}, O_{24}, O_{25}, O_{15}, O_{34})$ and $\sigma(S_o^2) = (O_{11}, O_{21}, O_{22}, O_{12}, O_{31}, O_{13}, O_{23}, O_{32}, O_{24}, O_{14}, O_{33}, O_{15}, O_{25}, O_{34})$, respectively. Let the length of a crossover string be 7 and the crossover starts at operation O_{31}. So the crossover string in S_o^2 is $\sigma' = (J_3, J_1, J_2, J_3, J_2, J_1, J_3)$ and the corresponding operations are O_{31}, O_{13}, O_{23}, O_{32}, O_{24}, O_{14}, and O_{33}. Then, delete the jobs that correspond to these operations from S_o^1 and insert σ' into S_o^1. The so obtained S_o^3 is $(J_1, J_1, J_2, \boldsymbol{J_3}, \boldsymbol{J_1}, \boldsymbol{J_2}, \boldsymbol{J_3}, \boldsymbol{J_2}, \boldsymbol{J_1}, \boldsymbol{J_3}, J_2, J_2, J_1, J_3)$. The first section of the trial solution is inherited directly from x_1. Then, we have the trial solution $x_k^u = (w_1, w_2, w_3; J_1, J_1, J_2, J_3, J_1, J_2, J_3, J_2, J_1, J_3, J_2, J_2, J_1, J_3)$. Note that the obtained trial solution x_k^u is infeasible and hence should be repaired by Amending Algorithm.

The selection operator in DDE decides whether a trial solution x_k^u should be a member of the population in the next generation. It is stated as follows:

$$x_k^i = \begin{cases} x_k^u, & \text{if } g^{te}\left(x_k^u \mid \lambda, z\right) \le g^{te}\left(x_{k-1}^i \mid \lambda, z\right) \\ x_{k-1}^i, & \text{otherwise} \end{cases} \tag{9}$$

where $z = (z_1, z_2, \ldots, z_L)^T$ and z^i is the best objective f_i found so far.

3.5. Framework of DMOEA/D. Let $\lambda^1, \lambda^2, \ldots, \lambda^K$ be a set of uniform spread weight vectors. DMOEA/D decomposes MOP into K single-objective subproblems with the Tchebycheff approach, and the jth subproblem is

$$\begin{aligned} \text{Minimize} \quad & g^{te}\left(x \mid \lambda^j, z^*\right) \\ & = \max_{i \in \mathbb{Z}_L} \left\{\lambda_i^j \left(f_i(x) - z_i^*\right)\right\}^T \\ \text{Subject to} \quad & x \in \Pi \end{aligned} \tag{10}$$

where $\lambda^j = (\lambda_1^j, \lambda_1^j, \ldots, \lambda_i^j)^T$. Given a weight vector λ^j, define the neighbourhood of λ^j as a set of its closest weight vectors in $\{\lambda^1, \lambda^2, \ldots, \lambda^K\}$. Then, all the subproblems that correspond to these weight vectors in the neighbourhood of λ^j constitute the neighbourhood of the jth subproblem. The population contains the best found solutions for all the subproblems. While DMOEA/D optimizes a subproblem, it only exploits the current solutions to the neighbourhood of the subproblem.

In each generation, our proposed DMOEA/D using a Tchebycheff approach contains a population of K solutions $x^1, \ldots, x^K \in \Pi$, where x^j is the current solution to the jth subproblem and an external population (EP) for storing nondominated solutions found in DMOEA/D. Then, our proposed DMOEA/D can be stated as in Algorithm 1.

From the algorithm, it can be known that $E(j)$ contains the indexes of H closest vectors of λ^j. This paper uses the Euclidean distance to evaluate how close any two weight vectors are. Hence, λ^j itself is its closest vector, $j \in E(j)$. The kth subproblem is in the neighbourhood of the jth subproblem if $k \in E(j)$. While considering the jth subproblem in DMOEA/D, since x^a, x^b, and x^c are the best found solutions to the respective neighbours of the jth subproblem, the reproduced solution x^u based on them should be a promising one to the jth subproblem. Then, amend x^u by using Amending Algorithm [17]. The so-obtained solution x^u is therefore feasible and probably better for the neighbours of the jth subproblem. Then, for each neighbour j^h of the jth subproblem, replace x^{j^h} with x^u if x^u is better for the jth subproblem. The external population EP is also updated based on the new solutions.

Since obtaining the accurate reference point z^* is usually very time-consuming, we use z as a substitute of z^* in g^{te}. z is initialized by a problem-specific method and updated according to the quality of generated trial solutions.

4. Illustrative Examples

To the authors' knowledge, there is still no work reported for the multiobjective scheduling of deadlock-prone FMSs. Thus, to demonstrate the effectiveness of DMOEA/D, we compare it with a NSGA-II (one of the best known MOEAs) based MOEA. Since the DAP used in Section III-A is also embedded, we rename the compared algorithm as DNSGA-II.

Input
K: the number of sub-problems used in DMOEA/D;
$\lambda^1, \lambda^2, \ldots, \lambda^K$: the set of uniform spread weight vectors;
H: the size of neighborhood of each weight vector;
Initialize
Set EP $= \varnothing$, initialize population $x^1, \ldots, x^K \in \Pi$;
Set $F^j = F(x^j)$, $j \in \mathbb{Z}_K$, initialize $z = (z_1, z_2, \ldots, z_L)^T$, where $z_i = \min_{x \in \Pi} f_i(x)$;
Compute the Euclidean distances between any two weight vectors and figure out the H closest ones of each weight vector;
Set $E(j) = \{j^1, j^2, \ldots, j^H\}$, where $\lambda^{j^1}, \lambda^{j^2}, \ldots, \lambda^{j^H}$ are the H closest weight vectors to λ^j.
While(the stopping criterion is not met) {
  For ($j = 1, 2, \ldots, K$) {
    Randomly select three different neighbors x^a, x^b, and x^c from $E(j)$;
    Generate the mutated solution x^v from x^a, x^b, and x^c;
    Generate the trial solution x^u from x^j and x^v; Amend x^u;
    For ($i = 1, 2, \ldots, L$) {
      If ($z_i < f_i(x^u)$) {
        Set $z_i = f_i(x^u)$; }
    } // End **For**
    For (each index $j^h \in E(j)$) {
      If($g^{te}(x^u \mid \lambda^{j^h}, z) \leq g^{te}(x^{j^h} \mid \lambda^{j^h}, z)$) {
        Set $x^{j^h} = x^u$; $F^{j^h} = F(x^u)$; }
    } // End **For**
    Remove all the vectors dominated by $F(x^u)$ from EP;
    If(there is no vector dominates $F(x^u)$ and $F(x^u)$ do not exist in EP) {
      Add $F(x^u)$ into EP; }
  } // End **For**
} // End **While**
Output EP

Algorithm 1: Algorithm DMOEA/D.

The stop criteria of two tested algorithms are all set as 1000 generations. The population size is set as $K = 100$. For DMOEA/D, the weight vectors are determined by a parameter I. Each weight vector chooses a value from $\{0/I, 1/I, \ldots, I/I\}$ and the number of the weight vectors is $K = C_{I+L-1}^{L-1}$. For problems with 2-objective, I is set to 99 since $C_{99+2-1}^{2-1} = K = 100$; for problems with 3-objective, I is set to 13 since $C_{13+3-1}^{3-1} = 105 > K = 100$ and DMOEA/D randomly chooses 100 values out of 105. The number of neighbourhoods is $H = 20$. The cross rate p_c and the mutation rate p_m are set as 0.8 and 0.2, respectively. All the algorithms are coded by C++ and simulated on a desktop PC with 3.2 GHz processor and 8 GB memory.

4.1. Performance Metrics. The number of Pareto solutions (*NPS*) obtained in a run is an important metric for MOEA. More Pareto solutions mean more candidate schedules for decision makers and therefore have better performance.

Mean ideal distance (*MID*) evaluates how close the solutions on a PF to the ideal point (often referred to point (0, 0)). It can be defined as

$$MID = \sum_{1 \leq j \leq NPS} \frac{\Delta_j}{NPS} \tag{11}$$

where $\Delta_j = \sqrt{\sum_{1 \leq i \leq L} f_{ji}^2}$ and f_{ji} is the ith objective of the jth Pareto solution. Less value of *MID* indicates better performance.

The rate of achievement to objectives simultaneously (*RAS*) evaluates the closeness of objectives. It can be represented as

$$RAS = \sum_{1 \leq j \leq NPS} \frac{\left(\sum_{1 \leq i \leq L} \left(f_{ji} / \min_{1 \leq i \leq L} f_{ji} - 1\right)\right)}{NPS} \tag{12}$$

The lower *RAS* value there is, the better solution quality we have.

4.2. Case 1. In this subsection, the FMS example in *Example* 1 is used to test the performance of the algorithms. The processing time of operations is listed in Table 2. 20 instances (FMS01- FMS20) designed by [39] are tested.

Firstly, we study the 2-objective problem with respect to makespan and mean completion time. The scheduling results of FMS01- FMS20 are shown in Table 3. The algorithms make 10 independent runs for each instance, and the metrics are averaged.

From Table 3, it can be seen that DMOEA/D outperforms DNSGA-II on all 3 metrics for the 2-objective problem. DMOEA/D achieves more *NPS* in 16 instances out of 20.

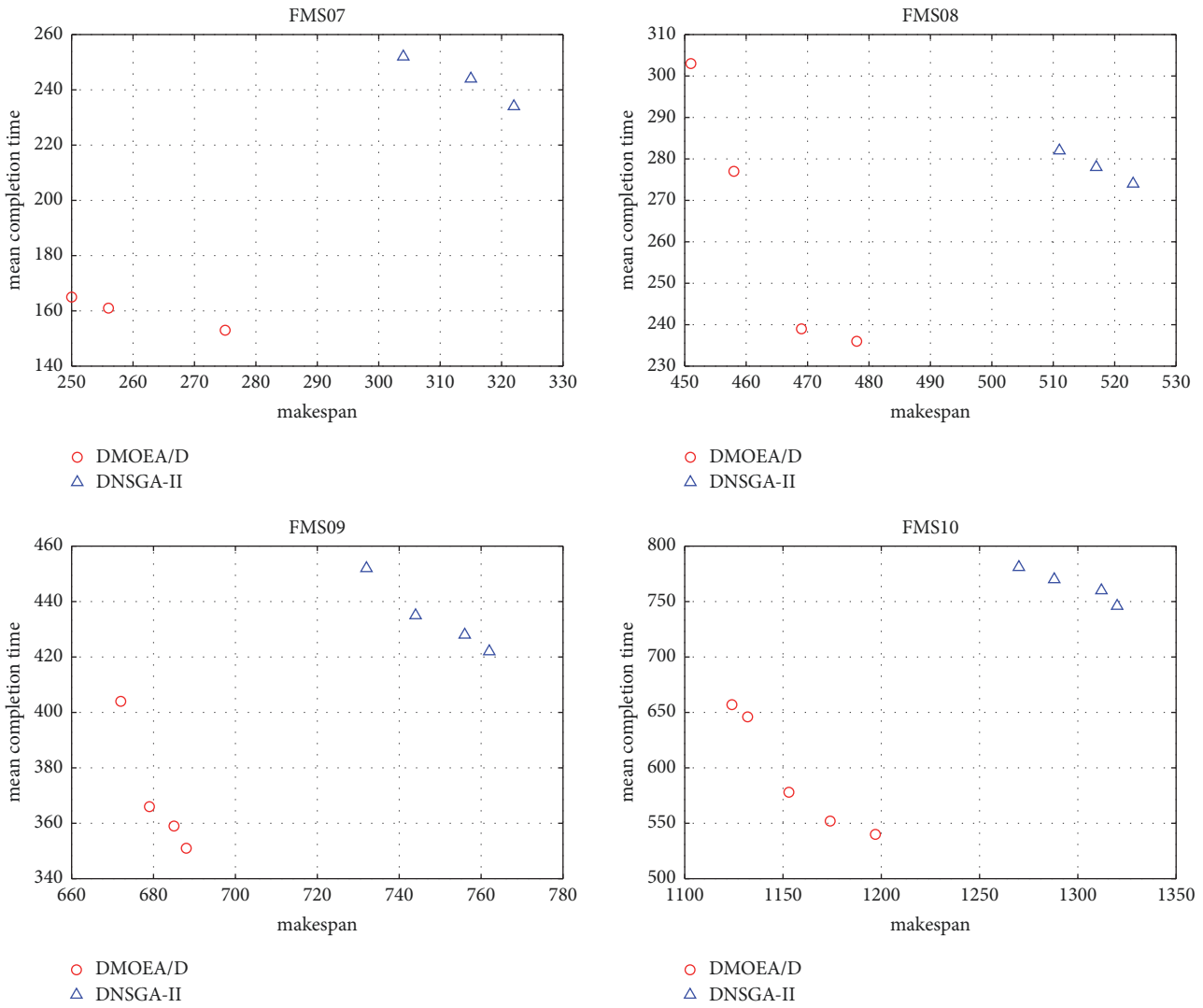

Figure 3: Nondominated solutions of 2-objective problem on instances FMS07-FMS10.

Table 2: Processing time of operations for the FMS in *Example* 1.

q_1		q_2
w_1	w_2	w_3
$d(O_{11})$: 25	$d(O_{21})$: 25	$d(O_{31})$: 26
$d(O_{12})$: 23	$d(O_{22})$: 20	$d(O_{32})$: 21
$d(O_{13})$: 27	$d(O_{23})$: 27	$d(O_{33})$: 24

This means that DMOEA/D can obtain more Pareto solutions than DNSGA-II for the studied problem. For the metric *MID*, DMOEA/D significantly outperforms DNSGA-II since it obtains less *MID* values for all 20 instances. This indicates that DMOEA/D gets a better convergence than DNSGA-II. For the metric *RAS*, it seems that the solutions of DMOEA/D have better closeness of objectives as it obtains lower *RAS* values than DNSGA-II in 17 instances out of 20.

Figure 3 shows a part of nondominated solutions of 2-objective on FMS02-FMS05. We can see that most solutions found by DMOEA/D dominate the ones found by DNSGA-II. DMOEA/D achieves a better approximation to PF than DNSGA-II does.

Then, the 3-objective problem is studied. The scheduling results are listed in Table 4. Each instance makes 10 independent runs and the metrics are averaged.

As seen in Table 4, the results for the 3-objective problem seem in accordance with the conclusions for 2-objective problem. DMOEA/D generally obtains more Pareto solutions (higher *NPS* values in 18 of 20 instances) than DNSGA-II for the studied problem. DMOEA/D also has a better convergence than DNSGA-II, since it obtains lower *MID* values in all 20 instances. The solutions obtained by DMOEA/D have better closeness of objectives as it obtains lower *RAS* values in 19 instances out of 20.

Note that the *RAS* values of instances with very few jobs (FMS01, FMS06, FMS11, and FMS16) are much higher than the ones of other instances. This is due to the fact that a schedule with very few jobs usually has very little tardiness

Table 3: Scheduling results of 2-objective problem (FMS01-FMS20).

Instances	*NPS*		*MID*		*RAS*	
	DNSGA-II	DMOEA/D	DNSGA-II	DMOEA/D	DNSGA-II	DMOEA/D
FMS01	**2.1**	2.0	218.33	**199.24**	0.80	**0.76**
FMS02	2.0	**2.7**	412.45	**350.20**	0.84	**0.79**
FMS03	2.3	**2.4**	657.22	**584.42**	0.91	**0.87**
FMS04	**2.6**	**2.6**	990.17	**875.01**	1.04	**0.98**
FMS05	2.8	**3.1**	1644.12	**1436.08**	**1.08**	1.09
FMS06	2.0	**2.3**	190.50	**165.82**	0.57	**0.55**
FMS07	2.8	**3.9**	361.71	**302.64**	0.61	**0.57**
FMS08	3.3	**3.8**	602.29	**537.69**	0.72	**0.65**
FMS09	3.8	**4.2**	917.27	**827.27**	1.05	**0.97**
FMS10	3.9	**4.8**	1525.18	**1354.03**	1.10	**1.04**
FMS11	**2.2**	1.9	158.93	**132.09**	0.55	**0.41**
FMS12	2.4	**2.5**	323.24	**257.43**	**0.56**	0.58
FMS13	3.1	**4.0**	568.74	**516.87**	0.72	**0.70**
FMS14	4.2	**4.6**	858.91	**779.25**	0.88	**0.83**
FMS15	4.5	**4.7**	1437.07	**1320.08**	0.98	**1.04**
FMS16	**1.8**	**1.8**	129.86	**118.01**	0.52	**0.43**
FMS17	2.4	**3.0**	242.64	**199.14**	0.55	**0.54**
FMS18	3.0	**3.2**	547.16	**508.87**	**0.72**	0.80
FMS19	3.4	**4.2**	827.83	**753.60**	0.90	**0.92**
FMS20	3.6	**4.3**	1358.92	**1182.49**	0.97	**0.99**

time, and the objective function mean tardiness time is thus much smaller than the other two.

The computational times of DNSGA-II and DMOEA/D on FMS01-FMS20 are almost at the same level. For the 2-objectives problem, the computational times for instances with 10, 20, 40, 60, and 100 jobs are around 5s, 10s, 25s, 40s, and 60s, respectively. For the 3-objectives problem, the computational times for instances with 10, 20, 40, 60, and 100 jobs are around 8s, 15s, 30s, 40s, and 65s, respectively.

4.3. Case 2. In this subsection, we use a widely researched FMS example to test the performance of our algorithms. This FMS consists of 3 robots r_1-r_3, 4 machines m_1-m_4, and can process 3 types of jobs q_1-q_3. Its PNS with the initial marking of In01 is shown in Figure 4 and the processing time of operations is listed in Table 5. 16 instances (In01-In16) designed by [18] are tested.

The 2-objective problem is studied first. The scheduling results of 16 instances are shown in Table 6. The algorithms make 10 independent runs for each instance and the metrics are averaged.

From Table 6, we can see that DMOEA/D outperforms DNSGA-II on all 3 metrics. It can obtain more Pareto solutions than DNSGA-II in 15 instances out of 16. DMOEA/D also gets a better convergence than DNSGA-II since it obtains less *MID* values in 15 instances out of 16. Moreover, the solutions of DMOEA/D have better closeness of objectives than DNSGA-II, since DMOEA/D obtains lower *RAS* values in 14 instances out of 16.

Figure 5 shows a part of nondominated solutions of 2-objective on instances In01-In04. From Figure 5, we can see that the solutions found by DMOEA/D can dominate the ones obtained by DNSGA-II.

For the 3-objective problem, the scheduling results of In01-In16 are listed in Table 7. Each instance makes 10 independent runs and the metrics are averaged.

As seen in Table 7, DMOEA/D obtains more Pareto solutions than DNSGA-II in 11 of 16 instances. DMOEA/D also has a better convergence than DNSGA-II, since it obtains lower *MID* values in all 16 instances. The solutions obtained by DMOEA/D have better closeness of objectives since it obtains lower *RAS* values in all 16 instances.

The computational times of DNSGA-II and DMOEA/D on In01-In16 are almost at the same level. For the 2-objectives problem, the computational times for instances with 28, 40, 50, and 60 jobs are around 20s, 55s, 90s, and 120s, respectively. For the 3-objective problem, the computational times for instances with 28, 40, 50, and 60 jobs are around 25s, 65s, 95s, and 130s, respectively.

5. Conclusions

This paper studies the multiobjective scheduling problem of deadlock-prone FMSs. Combining decomposition approaches and DDE, a novel scheduling algorithm called DMOEA/D is proposed based on the PN model of the studied system. DMOEA/D can decompose a multiobjective scheduling problem into a certain number of single-objective subproblems and solves all the subproblems in a single run. The solutions of the subproblems are reproduced by a new proposed DDE algorithm. The mutation operator of DDE is based on the hamming distance between two selected

Table 4: Scheduling results of 3-objective problem (FMS01-FMS20).

Instances	*NPS*		*MID*		*RAS*	
	DNSGA-II	DMOEA/D	DNSGA-II	DMOEA/D	DNSGA-II	DMOEA/D
FMS01	3.2	**4.4**	246.15	**218.61**	10.47	**8.54**
FMS02	3.9	**5.5**	486.50	**423.76**	4.42	**2.18**
FMS03	**4.3**	4.1	812.69	**703.04**	1.89	**1.41**
FMS04	4.5	**4.7**	1184.47	**1035.12**	2.24	**1.79**
FMS05	4.4	**5.2**	1982.92	**1802.27**	2.37	**2.02**
FMS06	3.5	**4.8**	207.96	**174.29**	21.26	**19.82**
FMS07	4.5	**5.0**	398.74	**341.40**	3.10	**2.87**
FMS08	4.4	**6.3**	739.06	**621.45**	1.92	**1.27**
FMS09	5.8	**5.9**	1115.63	**965.83**	2.01	**1.61**
FMS10	5.5	**6.0**	1870.84	**1685.44**	2.24	**1.96**
FMS11	**3.3**	2.5	170.02	**137.77**	30.36	**28.23**
FMS12	3.5	**3.6**	320.57	**263.89**	3.72	**3.47**
FMS13	3.9	**4.9**	671.11	**589.06**	1.82	**1.08**
FMS14	5.8	**6.0**	1054.72	**946.01**	1.87	**1.62**
FMS15	5.7	**6.2**	1759.81	**1594.49**	2.19	**1.94**
FMS16	2.2	**2.6**	144.56	**112.42**	**40.66**	41.47
FMS17	3.2	**2.5**	247.30	**191.23**	12.02	**10.76**
FMS18	4.0	**4.8**	617.54	**542.48**	1.14	**0.93**
FMS19	5.1	**5.2**	977.95	**856.20**	2.06	**1.47**
FMS20	5.0	**5.9**	1668.03	**1576.65**	2.38	**2.04**

Table 5: Processing times of operations for the FMS in Figure 4.

q_1	q_2		q_3
w_1	w_2	w_3	w_4
$d(O_{11})$: 8	$d(O_{21})$: 4	$d(O_{21})$: 4	$d(O_{41})$: 5
$d(O_{12})$: 34	$d(O_{22})$: 32	$d(O_{32})$: 23	$d(O_{42})$: 22
$d(O_{13})$: 5	$d(O_{23})$: 8	$d(O_{33})$: 6	$d(O_{43})$: 4
	$d(O_{24})$: 38	$d(O_{34})$: 20	$d(O_{44})$: 17
	$d(O_{25})$: 5	$d(O_{25})$: 5	$d(O_{45})$: 6

Table 6: Scheduling results of 2-objective problem (In01-In16).

Instances	*NPS*		*MID*		*RAS*	
	DNSGA-II	DMOEA/D	DNSGA-II	DMOEA/D	DNSGA-II	DMOEA/D
In01	1.4	**1.6**	465.13	**384.32**	**0.69**	0.73
In02	1.3	**2.0**	646.31	**563.24**	0.81	**0.78**
In03	1.5	**1.7**	886.41	**752.05**	0.99	**0.81**
In04	1.7	**2.1**	1089.87	**920.63**	1.08	**0.83**
In05	1.4	**1.7**	396.37	**366.64**	**0.68**	**0.68**
In06	1.6	**1.6**	526.92	**523.22**	0.86	**0.72**
In07	1.5	**1.5**	**691.50**	699.44	0.95	**0.83**
In08	1.5	**1.9**	882.66	**839.75**	0.99	**0.82**
In09	1.6	**1.8**	339.47	**298.18**	0.65	**0.58**
In10	1.5	**1.6**	476.35	**451.60**	0.77	**0.62**
In11	1.7	**2.0**	585.71	**559.66**	0.84	**0.68**
In12	1.6	**1.8**	719.93	**655.98**	0.83	**0.69**
In13	1.4	**1.5**	328.30	**263.18**	0.61	**0.54**
In14	1.6	**1.7**	465.32	**416.40**	0.71	**0.61**
In15	1.6	**1.5**	550.88	**514.23**	0.79	**0.66**
In16	**1.7**	**1.7**	687.50	**654.51**	0.85	**0.72**

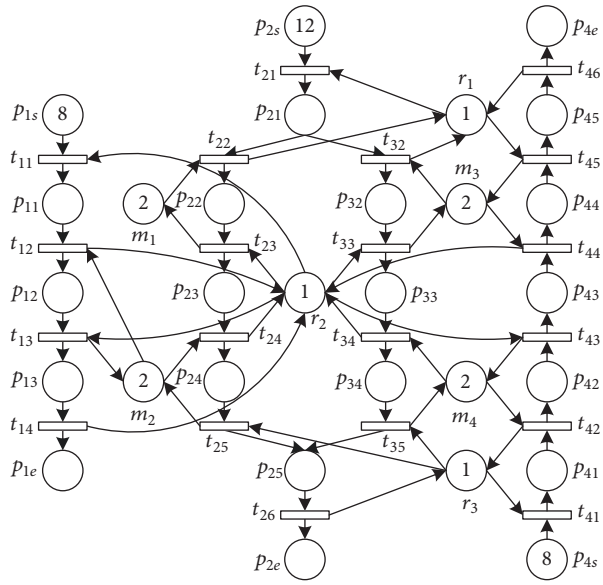

FIGURE 4: PNS model of the studied FMS (In01).

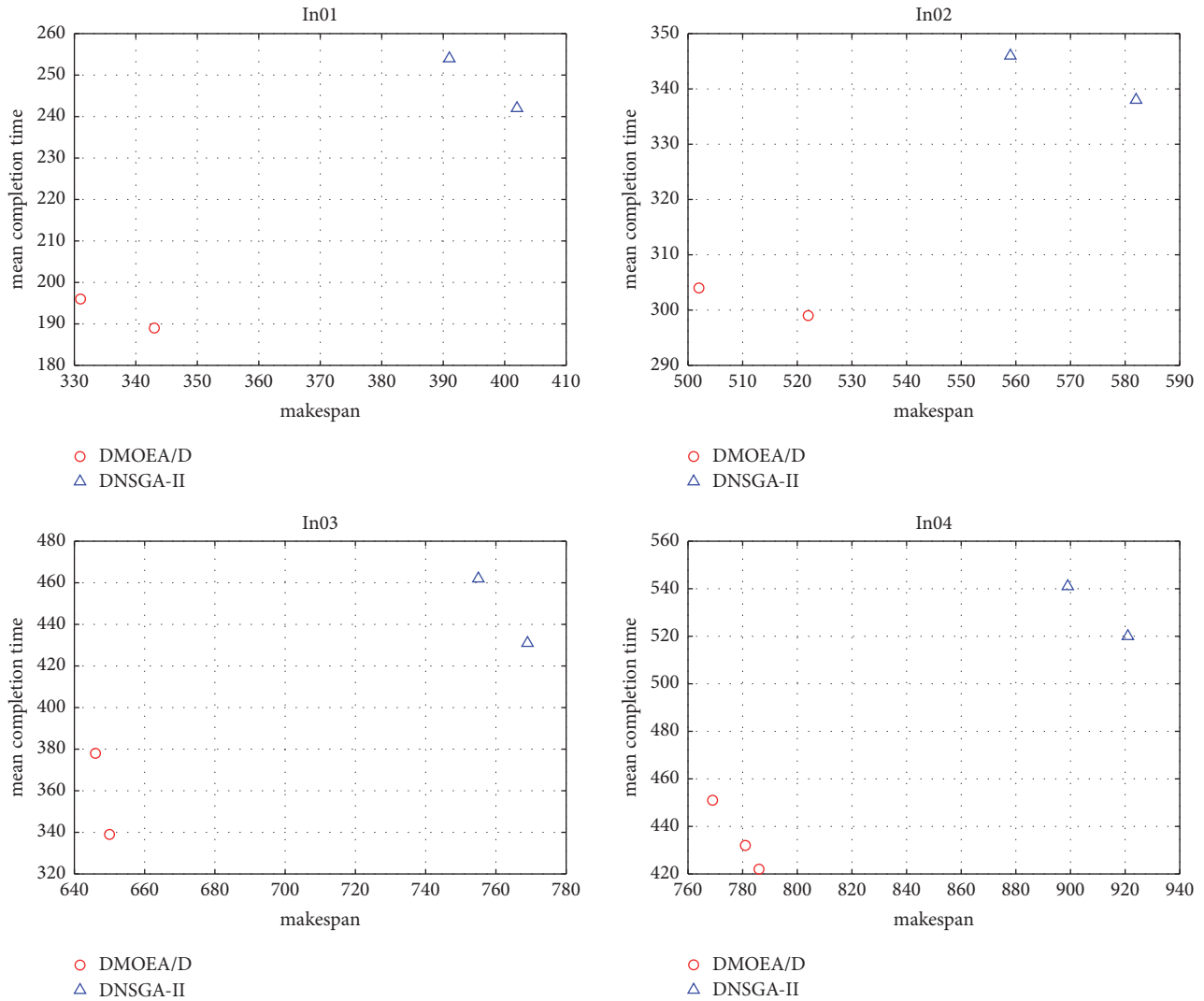

FIGURE 5: Non-dominated solutions of 2-objective problem on instances In01-In04.

Table 7: Scheduling results of 3-objective problem (In01-In16).

Instances	*NPS*		*MID*		*RAS*	
	DNSGA-II	DMOEA/D	DNSGA-II	DMOEA/D	DNSGA-II	DMOEA/D
In01	1.8	**2.2**	467.16	**412.22**	4.32	**3.53**
In02	1.6	**1.8**	793.22	**662.40**	3.80	**1.98**
In03	**2.1**	2.0	1061.19	**847.26**	3.52	**1.79**
In04	**2.1**	**2.1**	1299.54	**1056.08**	3.06	**1.43**
In05	1.6	**1.8**	423.02	**372.25**	4.86	**3.08**
In06	**2.0**	**2.0**	674.95	**577.32**	4.42	**2.06**
In07	2.2	**2.4**	956.23	**725.51**	3.93	**1.90**
In08	2.3	**2.5**	1129.71	**909.89**	3.69	**1.58**
In09	**1.9**	**1.9**	354.21	**319.51**	5.23	**3.22**
In10	2.5	**2.6**	562.76	**460.27**	4.97	**2.38**
In11	1.9	**2.1**	785.41	**636.38**	4.48	**1.76**
In12	2.5	**2.8**	977.70	**761.75**	4.01	**1.61**
In13	**2.1**	1.8	309.61	**263.67**	5.82	**4.58**
In14	2.3	**2.5**	500.26	**432.11**	4.78	**2.49**
In15	2.5	**2.7**	729.43	**572.32**	4.73	**2.01**
In16	2.1	**2.4**	891.41	**681.56**	4.24	**1.79**

solutions. GOX is used as the crossover operator. Two benchmark examples are used to test the DMOEA/D. Computational results demonstrate that our proposed DMOEA/D can outperform DNSGA-II for both 2-objective and 3-objective problems on the studied FMSs.

The advantage of DMOEA/D over DNSGA-II may attribute to the following: (1) DMOEA/D optimizes several single-objective optimization problems rather than directly solving the MOP, and hence the diversity of the population is easier to maintain, and more evenly distributed Pareto solutions can thus be obtained; and (2) coevolution mechanism between subproblems is used in DMOEA/D, and, hence, DMOEA/D can converge faster and obtain better Pareto solutions than DNSGA-II.

For the future work, one research direction is to introduce other efficient local optimization strategies or heuristics to improve the search capability of the proposed algorithm. Another direction is to extend the proposed method to solve the scheduling problem of FMSs with other objectives or constraints, such as energy consumptions, unreliable resources, and maintenance activities, which may be proved both interesting and useful.

Data Availability

(1). All data generated by the simulation in this study are included within the article in Tables 3, 4, 6, and 7. (2). Previously reported data in [17, 39] are used to support our study, and they are included in Tables 2 and 5. (3). No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Science Foundation of PR China [grant number 61573278].

References

[1] A. Alvarado-Iniesta, J. L. García-Alcaraz, M. Piña-Monarrez, and L. Pérez-Domínguez, "Multiobjective optimization of torch brazing process by a hybrid of fuzzy logic and multiobjective artificial bee colony algorithm," *Journal of Intelligent Manufacturing*, vol. 27, no. 3, pp. 631–638, 2016.

[2] J. C. Leyva López, J. J. Solano Noriega, J. L. García Alcaraz, and D. A. Gastélum Chavira, "Exploitation of a medium-sized fuzzy outranking relation based on multi-objective evolutionary algorithms to derive a ranking," *International Journal of Computational Intelligence Systems*, vol. 9, no. 4, pp. 745–764, 2016.

[3] R. Domingo, B. de Agustina, and M. M. Marín, "A multiresponse optimization of thrust forces, torques, and the power of tapping operations by cooling air in reinforced and unreinforced polyamide PA66," *Sustainability*, vol. 10, no. 3, p. 889, 2018.

[4] V. K. Chawla, A. K. Chanda, and S. Angra, "Sustainable multiobjective scheduling for automatic guided vehicle and flexible manufacturing system by a grey wolf optimization algorithm," *International Journal of Data and Network Science*, vol. 2, no. 1, pp. 27–40, 2018.

[5] W. K. Mashwani and A. Salhi, "Multiobjective memetic algorithm based on decomposition," *Applied Soft Computing*, vol. 21, pp. 221–243, 2014.

[6] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE*

Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.

[8] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[9] D. Chen, F. Zou, R. Lu, L. Yu, Z. Li, and J. Wang, "Multi-objective optimization of community detection using discrete teaching-learning-based optimization with decomposition," *Information Sciences*, vol. 369, pp. 402–418, 2016.

[10] F. Zhao, Z. Chen, J. Wang, and C. Zhang, "An improved MOEA/D for multi-objective job shop scheduling problem," *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 6, pp. 616–640, 2017.

[11] V. K. Chawla, A. K. Chanda, and S. Angra, "Automatic guided vehicles fleet size optimization for flexible manufacturing system by grey wolf optimization algorithm," *Management Science Letters*, vol. 8, no. 2, pp. 79–90, 2018.

[12] S. Angra, A. K. Chanda, and V. K. Chawla, "Comparison and evaluation of job selection dispatching rules for integrated scheduling of multi-load automatic guided vehicles serving in variable sized flexible manufacturing system layouts: a simulation study," *Management Science Letters*, vol. 8, no. 4, pp. 187–200, 2018.

[13] G. Demesure, M. Defoort, A. Bekrar, D. Trentesaux, and M. Djemai, "Decentralized motion planning and scheduling of AGVs in an FMS," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1744–1752, 2018.

[14] C. Su, X. Guan, Y. Du, X. Huang, and M. Zhang, "Toward capturing heterogeneity for inferring diffusion networks: a mixed diffusion pattern model," *Knowledge-Based Systems*, vol. 147, pp. 81–93, 2018.

[15] H. H. Xiong and M. Zhou, "Scheduling of semiconductor test facility via petri nets and hybrid heuristic search," *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no. 3, pp. 384–393, 1998.

[16] I. B. Abdallah, H. A. ElMaraghy, and T. ElMekkawy, "Deadlock-free scheduling in flexible manufacturing systems using Petri nets," *International Journal of Production Research*, vol. 40, no. 12, pp. 2733–2756, 2002.

[17] K. Xing, L. Han, M. Zhou, and F. Wang, "Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 3, pp. 603–615, 2012.

[18] L. Han, K. Xing, X. Chen, H. Lei, and F. Wang, "Deadlock-free genetic scheduling for flexible manufacturing systems using Petri nets and deadlock controllers," *International Journal of Production Research*, vol. 52, no. 5, pp. 1557–1572, 2014.

[19] O. T. Baruwa, M. A. Piera, and A. Guasch, "Deadlock-free scheduling method for flexible manufacturing systems based on timed colored petri nets and anytime heuristic search," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 5, pp. 831–846, 2015.

[20] J. Luo, K. Xing, M. Zhou, X. Li, and X. Wang, "Deadlock-free scheduling of automated manufacturing systems using petri nets and hybrid heuristic search," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 530–541, 2015.

[21] X. Li, K. Xing, Y. Wu, X. Wang, and J. Luo, "Total energy consumption optimization via genetic algorithm in flexible manufacturing systems," *Computers & Industrial Engineering*, vol. 104, pp. 188–200, 2017.

[22] X. Li, K. Xing, M. Zhou, X. Wang, and Y. Wu, "Modified dynamic programming algorithm for optimization of total energy consumption in flexible manufacturing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 691–705, 2019.

[23] X. Wang, K. Xing, X. Li, and J. Luo, "An estimation of distribution algorithm for scheduling problem of flexible manufacturing systems using Petri nets," *Applied Mathematical Modelling: Simulation and Computation for Engineering and Environmental Systems*, vol. 55, pp. 776–788, 2018.

[24] C. Bierwirth, "A generalized permutation approach to job shop scheduling with genetic algorithms," *OR Spectrum*, vol. 17, no. 2-3, pp. 87–92, 1995.

[25] J.-I. Latorre-Biel, E. Jiménez-Macías, J. Blanco-Fernández, and J. C. Sáenz-Díez, "Optimal design of an olive oil mill by means of the simulation of a petri net model," *International Journal of Food Engineering*, vol. 10, no. 4, pp. 573–582, 2014.

[26] J. I. Latorre-Biel, E. Jiménez-Macías, J. Blanco-Fernández, E. Martínez-Cámara, J. C. Sáenz-Díez, and M. Pérez-Parte, "Decision support system, based on the paradigm of the petri nets, for the design and operation of a dairy plant," *International Journal of Food Engineering*, vol. 11, no. 6, pp. 767–776, 2015.

[27] J.-I. Latorre-Biel, E. Jiménez-Macías, M. Pérez-De-La-Parte, J. C. Sáenz-Díez, E. Martínez-Cámara, and J. Blanco-Fernández, "Compound petri nets and alternatives aggregation petri nets: two formalisms for decision-making support," *Advances in Mechanical Engineering*, vol. 8, no. 11, pp. 1–12, 2016.

[28] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[29] S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 442–446, 2012.

[30] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: a multiobjective evolutionary algorithm using decomposition and ant colony," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1845–1859, 2013.

[31] C. Wang, Z. Ji, and Y. Wang, "A novel memetic algorithm based on decomposition for multiobjective flexible job shop scheduling problem," *Mathematical Problems in Engineering*, vol. 2017, Article ID 2857564, 20 pages, 2017.

[32] Y. Huang, M. Jeng, X. Xie, and S. Chung, "Deadlock prevention policy based on Petri nets and siphons," *International Journal of Production Research*, vol. 39, no. 2, pp. 283–305, 2001.

[33] L. Piroddi, R. Cordone, and I. Fumagalli, "Selective siphon control for deadlock prevention in Petri nets," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 38, no. 6, pp. 1337–1348, 2008.

[34] K. Y. Xing, M. C. Zhou, H. X. Liu, and F. Tian, "Optimal Petri-net-based polynomial-complexity deadlock-avoidance policies for automated manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics: Part A: Systems and Humans*, vol. 39, no. 1, pp. 188–199, 2009.

[35] Y. Feng, K. Xing, M. Zhou, X. Wang, and H. Liu, "Robust deadlock prevention for automated manufacturing systems with unreliable resources by using general Petri nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2018.

[36] Y. X. Feng, K. Y. Xing, M. C. Zhou, F. Tian, and H. X. Liu, "Structural liveness analysis of automated manufacturing systems modeled by S^4PRs," *IEEE Transactions on Automation Science and Engineering*, 2019.

[37] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[38] Q.-K. Pan, M. F. Tasgetiren, and Y.-C. Liang, "A discrete differential evolution algorithm for the permutation flowshop scheduling problem," *Computers & Industrial Engineering*, vol. 55, no. 4, pp. 795–816, 2008.

[39] L. Han, K. Xing, X. Chen, and F. Xiong, "A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 29, no. 5, pp. 1083–1096, 2018.