doi: 10.3969/j.issn.1007-7375.230102

考虑序列相关准备时间的分布式柔性作业车间调度研究

王有远[1,2]，董博文[3]

(1. 南昌航空大学 工业工程研究所，江西 南昌 330063；2. 南昌市航空复杂系统与智能科学重点实验室，
江西 南昌 330063；3. 南昌航空大学 飞行器工程学院，江西 南昌 330063)

摘要: 针对考虑序列相关准备时间的分布式柔性作业车间调度问题，提出以最小化最大完工时间为优化目标的混合整数线性规划模型，并提出一种改进遗传算法。采用基于负荷均衡的种群初始化方法提高初始种群质量，根据问题特性构造 6 个局部扰动算子，设计多重局部扰动策略提高算法的局部搜索能力。通过扩展柔性作业车间调度基准生成测试算例，使用正交实验确定算法参数。实验结果表明，所提改进策略能够有效提高算法性能，求解结果优于对比算法，验证了调度模型和所提算法的可行性和有效性。

关键词: 分布式柔性作业车间调度；序列相关准备时间；遗传算法；最大完工时间

中图分类号: F406.2; TP18　　**文献标志码:** A　　**文章编号:** 1007-7375(2024)03-0078-09

Distributed Flexible Job Shop Scheduling with Sequence-Dependent Setup Times

WANG Youyuan[1,2], DONG Bowen[3]

(1. Institute of Industrial Engineering, Nanchang Hangkong University, Nanchang 330063, China;
2. Nanchang Key Laboratory of Aviation Complex Systems and Intelligence Science, Nanchang 330063, China;
3. School of Aircraft Engineering, Nanchang Hangkong University, Nanchang 330063, China)

Abstract: For the distributed flexible job shop scheduling problem considering sequence-dependent setup times, a mixed-integer linear programming model with the optimization objective of minimizing the makespan is proposed. Also, an improved genetic algorithm is developed. A load-balanced population initialization method is used to improve the quality of the initial population. Six local perturbation operators are constructed according to problem characteristics, and a multiple local perturbation strategy is designed to improve the local search capability of the algorithm. Test cases are generated by extending the flexible job shop scheduling benchmark, and the algorithm parameters are determined by orthogonal experiments. Experimental results show that the proposed strategy can effectively improve the performance of the algorithm, with solutions superior to those obtained by the comparison algorithms, thus verifying the feasibility and effectiveness of the scheduling model and the proposed algorithm.

Key words: distributed flexible job shop scheduling; sequence-dependent setup times; genetic algorithm; makespan

经济全球化发展导致市场竞争日益激烈，为提高竞争力，许多制造企业逐步由单一工厂制造转变为多工厂协同制造，制造模式发展为分布式制造。生产调度在分布式制造中进一步延伸，在机械加工等制造场景中，产生分布式柔性作业车间调度问题(distributed flexible job shop scheduling problem, DFJSP)。DFJSP 包含多个分布在不同地理位置的工厂，或同一工厂内的多个车间，不仅要将工件分配给各工厂/车间，同时要解决每一工厂/车间内的调度问题，已被证明为 NP−难问题[1]，其研究具有理论意义。有效解决 DFJSP 有助于缩短制造期，提高机器利用率，因此其研究同样具有应用价值。

国内外学者对 DFJSP 进行了相关研究。DE Giovanni 等[1]提出一种改进遗传算法求解 DFJSP，并将柔性作业车间调度问题 (flexible job shop scheduling problem, FJSP) 基准算例扩展为 DFJSP 算例。Wu 等[2]同样提出一种遗传算法求解 DFJSP，但是使用了一维编码方式，设计启发式规则解码为三维调度

收稿日期: 2023-05-21

基金项目: 国家自然科学基金资助项目 (71761028)；航空基金资助项目 (2022Z069056002)

作者简介: 王有远 (1965—)，男，江西省人，教授，博士，主要研究方向为智能制造与制造业信息化。

解。吴秀丽等[3]设计一种改进差分进化算法求解 DFJSP，利用模拟退火提高算法的局部搜索能力。孟磊磊等[4]则结合变邻域搜索提高算法的局部搜索能力，提出一种混合蛙跳算法求解 DFJSP。Li 等[5]提出一种改进灰狼优化算法求解 DFJSP，使用了基于关键工厂和关键路径的邻域搜索策略。Luo 等[6]认为现实环境中同一工件可能需要在不同工厂加工，提出考虑运输时间的 DFJSP，并设计了模因算法进行求解。Du 等[7]针对需要使用起重机运输工件的制造场景，考虑起重机使用约束及运输时间，使用混合分布估计算法结合变邻域搜索进行求解。Xu 等[8]考虑存在工序外包的情形，同样在 DFJSP 中引入运输时间，设计了混合遗传算法和禁忌搜索进行求解。张洪亮等[9]针对考虑运输时间的 DFJSP，同时优化最大完工时间和总能耗，提出一种改进的非支配排序遗传算法进行求解。唐红涛等[10]针对加工时间不确定的模糊 DFJSP，以最小化最大模糊完工时间为优化目标，提出一种改进灰狼优化算法进行求解。Li 等[11]针对模糊 DFJSP，提出两阶段知识驱动进化算法，根据问题特性设计启发式规则和邻域结构提高求解能力。

实际生产中通常需要进行工件装夹、刀具调整、模具更换等准备工作，其准备时间通常与工件加工顺序相关，即序列相关准备时间。忽略序列相关准备时间会导致实际状况与调度计划产生偏差，造成机器负荷不均衡、制造期增加等问题，调度计划的适用性变差。然而，目前的分布式柔性作业车间调度研究均未考虑序列相关准备时间，基于此，本文以考虑序列相关准备时间的分布式柔性作业车间调度问题 (DFJSP with sequence-dependent setup times, DFJSP-SDST) 为研究对象，提出以最小化最大完工时间为优化目标的混合整数线性规划模型。由于遗传算法能够很好地对组合优化问题进行编码，并通过设计有效的进化算子保证进化过程解的可行性，因此被广泛应用于求解 DFJSP[1-2, 6, 8-9]。为此，本文提出一种遗传算法求解 DFJSP-SDST，并设计多重局部扰动策略提高算法局部搜索能力。

1　问题描述与建模

1.1　符号定义

本文中使用的符号定义如表 1 所示。

表 1　符号定义

Table 1　Definitions of notations

符号	定义
i	工件索引
j	工序索引
k	机器索引
l	工厂索引
r	机器上的加工位置索引
n	工件数量
q	工厂数量
n_i	工件 i 的工序数
m^l	工厂 l 的机器数
O_{ij}	第 i 个工件的第 j 个工序
r^{lk}	工厂 l 中机器 k 的加工工序总数
R^{lk}	工厂 l 中机器 k 的加工位置集合，$R^{lk}=\{1,2,\cdots,r^{lk}\}$
U	工厂集合，$U=\{U^1,U^2,\cdots,U^l,\cdots,U^q\}$
J	工件集合，$J=\{J_1,J_2,\cdots,J_i,\cdots,J_n\}$
J_i	工件 i 的工序集，$J_i=\{O_{i1},O_{i2},\cdots,O_{ij},\cdots,O_{in_i}\}$
M^l	工厂 l 机器集，$M^l=\{M^{l1},M^{l2},\cdots,M^{lk},\cdots,M^{lm^l}\}$
M_{ij}^l	工厂 l 中工序 O_{ij} 的可用机器集
p_{ij}^{lk}	O_{ij} 在工厂 l 机器 k 上的加工时间
s_{ij}^{lk}	O_{ij} 是工厂 l 机器 k 上第一个加工工序时的准备时间
$s_{iji'j'}^{lk}$	O_{ij} 不是工厂 l 机器 k 上第一个加工工序时的准备时间，此时上一个加工工序为 $O_{i'j'}$
d_i^l	工件 i 在工厂 l 加工完后运输到客户的时间
L	一个足够大的正数
C^{lkr}	工厂 l 中机器 k 的第 r 次加工完成时间
C_{ij}	工序 O_{ij} 的完成时间
C_i	工件 i 的完工时间
$C_{\max}$	最大完工时间
A_{ij}	准备时间不可分离时为 1，可分离时为 0
x_{ij}^{lkr}	O_{ij} 在工厂 l 机器 k 第 r 个位置时为 1，否则为 0
y_i^l	工件 i 在工厂 l 加工时为 1，否则为 0

1.2　问题描述

考虑序列相关准备时间的 DFJSP 描述如下。有 n 个工件 $J=\{J_1,J_2,\cdots,J_i,\cdots,J_n\}$ 要在 q 个工厂 $U=\{U^1,U^2,\cdots,U^l,\cdots,U^q\}$ 加工，工厂 l 有 m^l 台机器 $M^l=\{M^{l1},M^{l2},\cdots,M^{lk},\cdots,M^{lm^l}\}$，工件 i 有 n_i 个工序 $J_i=\{O_{i1},O_{i2},\cdots,O_{ij},\cdots,O_{in_i}\}$。在工厂 l 中，工序 O_{ij} 的

可加工机器集为M_{ij}^{l}，$M_{ij}^{l}\subseteq M^{l}$。工序$O_{ij}$的加工时间为$p_{ij}^{lk}$，在不同工厂的不同机器上可能不同。若工序$O_{ij}$在工厂$l$的机器$k$上第一个加工，则其准备时间为$s_{ij}^{lk}$；若在工序$O_{i'j'}$后加工，则其准备时间为$s_{iji'j'}^{lk}$。准备时间分为可分离和不可分离两种，可分离即在工件上一工序尚未完成时，可提前执行下一工序的准备作业，用 0-1 变量A_{ij}表示准备时间类型，1 表示不可分离，0 表示可分离。DFJSP-SDST 的目标是将工件分配给工厂，选择加工工序的机器和确定各机器上的加工顺序，使得调度的最大完工时间最小。问题的约束和假设条件如下。

1) 一个工件只能被分配给一个工厂，该工件所有工序都在同一工厂加工；

2) 所有工厂、机器和工件都在 0 时刻可用；

3) 一台机器在同一时刻只能加工一个工件；

4) 同一工件在同一时刻只能在一台机器上加工；

5) 工序一旦开始加工便不能中断；

6) 工件间无优先顺序，同一工件的所有工序需满足优先约束；

7) 不考虑机器故障、新工单插入等紧急情况。

1.3 数学模型

本文以最小化最大完工时间为优化目标，构建 DFJSP-SDST 的数学模型 (见式 (1)～(11))。

$$\min\ C_{\max}=\max\left\{C_{in_i}+\sum_{l\in U}d_i^{l}y_i^{l}\right\},\ \forall i\in J。\tag{1}$$

s.t.

$$\sum_{l\in U}y_i^{l}=1,\ \forall i\in J;\tag{2}$$

$$y_i^{l}=\sum_{k\in M_{ij}^{l}}\sum_{r\in R^{lk}}x_{ij}^{lkr},\ \forall i\in J,\ j\in J_i,\ l\in U;\tag{3}$$

$$\sum_{i\in J}\sum_{j\in J_i}x_{ij}^{lkr}\leqslant 1,\ \forall l\in U,\ k\in M^{l},\ r\in R^{lk};\tag{4}$$

$$\sum_{i\in J}\sum_{j\in J_i}x_{ij}^{lk(r-1)}\geqslant\sum_{i'\in J}\sum_{j'\in J_{i'}}x_{i'j'}^{lkr},\ \forall l\in U,\ k\in M^{l},\ r\in R^{lk},\ r\neq 1;\tag{5}$$

$$C^{lkr}\geqslant C_{ij}-L\left(1-x_{ij}^{lkr}\right),\ \forall i\in J,\ j\in J_i,\ l\in U,\ k\in M_{ij}^{l},\ r\in R^{lk};\tag{6}$$

$$C^{lkr}\leqslant C_{ij}+L\left(1-x_{ij}^{lkr}\right),\ \forall i\in J,\ j\in J_i,\ l\in U,\ k\in M_{ij}^{l},\ r\in R^{lk};\tag{7}$$

$$C^{lk1}-p_{ij}^{lk}-s_{ij}^{lk}+L\left(1-x_{ij}^{lk1}\right)\geqslant 0,\ \forall i\in J,\ j\in J_i,\ l\in U,\ k\in M^{l};\tag{8}$$

$$C^{lkr}-p_{ij}^{lk}-s_{iji'j'}^{lk}+L\left(2-x_{ij}^{lkr}-x_{i'j'}^{lk(r-1)}\right)\geqslant C^{lk(r-1)},\ \forall i,\ i'\in J,\ j\in J_i,\ j'\in J_{i'},\ l\in U,\ k\in M^{l},\ r\in R^{lk},\ r\neq 1,\ (i,j)\neq(i',j');\tag{9}$$

$$C^{lk1}-p_{ij}^{lk}-A_{ij}s_{ij}^{lk}+L\left(2-x_{ij}^{lk1}-x_{i(j-1)}^{lk'r'}\right)\geqslant C^{lk'r'},\ \forall i\in J,\ j\in J_i,\ l\in U,\ k,k'\in M^{l},\ r'\in R^{lk'},\ j\neq 1,\ (k,1)\neq(k',\ r');\tag{10}$$

$$C^{lkr}-p_{ij}^{lk}-A_{ij}s_{iji'j'}^{lk}+L\left(3-x_{ij}^{lkr}-x_{i'j'}^{lk(r-1)}-x_{i(j-1)}^{lk'r'}\right)\geqslant C^{lk'r'},\ \forall i,i'\in J,\ j\in J_i,\ j'\in J_{i'},\ l\in U,\ k,k'\in M^{l},\ r\in R^{lk},\ r'\in R^{lk'},\ r\neq 1,\ j\neq 1,\ (i,j)\neq(i',j'),\ (k,r)\neq(k',r')。\tag{11}$$

式 (1) 为目标函数；式 (2) 表示一个工件只能分配到一个工厂；式 (3) 表示同一工件的所有工序在同一工厂加工，且同一工序只能被加工一次；式 (4) 表示一台机器在同一时刻只能加工一个工件；式 (5) 表示在机器前一位置未安排工序时不可安排后一位置；式 (6) 和式 (7) 表示工序完工时间和机器加工位置完工时间的关系；式 (8) 和式 (9) 保证在同一机器上加工的工序无重叠，且开始准备时间大于等于 0，结合式 (6) 和式 (7) 可保证各工件的第一个工序开始准备时间大于等于 0；式 (10) 和式 (11) 保证同一工件的工序满足优先约束。

2 改进遗传算法求解 DFJSP-SDST

2.1 编码与解码

为形成问题与解空间的完全映射，使用完全编码方式，染色体包括工厂分配、工序排序和机器选择 3 部分。一个 DFJSP-SDST 实例如表 2 所示。

表 2 DFJSP-SDST 实例

Table 2 DFJSP-SDST instances

工件	工序	A_{ij}	U^1		U^2	
			M^{11}	M^{12}	M^{21}	M^{22}
J_1	O_{11}	1	(15,1)	(10,2)	(20,2)	(10,1)
	O_{12}	0	(20,2)	(15,1)	(10,1)	—
	O_{13}	0	(10,1)	—	(20,2)	—
J_2	O_{21}	1	(15,1)	(10,1)	—	(15,2)
	O_{22}	1	(20,2)	(10,1)	(15,1)	(20,2)
J_3	O_{31}	1	—	(15,2)	(20,2)	(10,1)
	O_{32}	1	(20,2)	(15,2)	(10,1)	(15,1)
	O_{33}	1	(20,2)	—	(15,1)	(10,1)

该实例包含 3 个工件 8 个工序，需要分配到 2 个工厂，每个工厂包含 2 台机器，工序 O_{12} 和 O_{13} 的准备时间可分离。表中括号内数值对表示加工时间和当工序在机器上第一个加工时的准备时间，为简单示范，工序间的序列相关准备时间均设为 3。工件完工后运输到客户的时间包含在最后一道工序加工时间内。表中“—”表示不可在该机器上加工。上述实例的一个编码方案如图 1 所示，工厂分配编码 FA 长度与工件数相同，第 i 个基因表示工件 i 所分配的工厂，如第 1 个基因表示工件 J_1 分配到工厂 2。工序排序编码 OS 长度与总工序数相同，使用工件号编码，工件号出现的次数表示第几个工序，如第 3 个基因为 1，第 2 次出现，表示为工序 O_{12}。机器选择编码 MS 长度同样与总工序数相同，且其对应的工序保持原有顺序，染色体中的数字代表所选择的机器在工厂中的编号，如第 1 个基因对应工序 O_{11}，根据 FA 编码其分配到工厂 2，则数字 2 表示加工 O_{11} 的机器为 M^{22}。

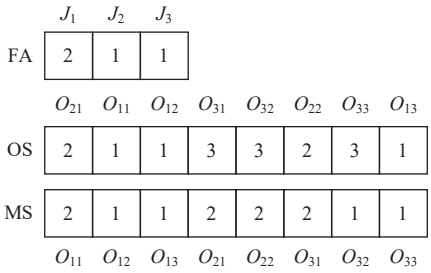

图 1　DFJSP-SDST 实例的一个编码方案

Figure 1　An encoding scheme for a DFJSP-SDST instance

染色体解码依 OS 顺序依次解码，根据 FA 编码确定工序所在工厂，再根据 MS 编码确定加工机器。在将工序分配给机器时，使用贪婪插入策略，即将工序分配到机器上满足约束的最早空闲时间段，这样有利于充分利用机器，减少完工时间。完成整个调度后，按照工序开始加工时间的升序重新生成 OS 编码，以保留贪婪插入策略对染色体的改善。图 1 编码方案经贪婪插入解码后对应的调度甘特图如图 2 所示。每一个方块代表一个工序，方块底部分为两部分，分别表示准备时间和加工时间。注意工序 O_{12} 在工序 O_{11} 还没有完工时就开始执行准备作业，即对应其可分离准备时间的属性。

2.2　种群初始化

初始种群的质量对于算法迭代搜索有一定的影响，好的初始种群能够帮助算法找到更优解。为提高初始种群质量，随机生成 OS 编码，使用基于工厂和机器负荷均衡的种群初始化方法，计算过程如表 3 所示。

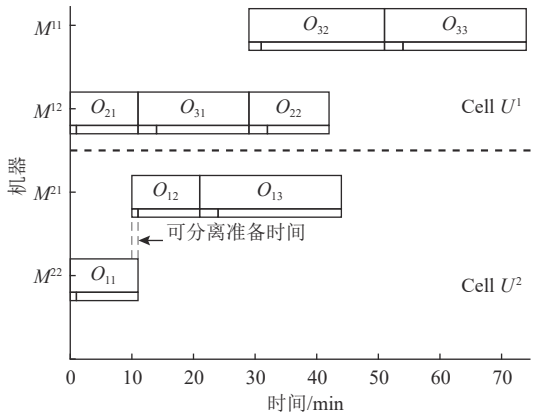

图 2　DFJSP-SDST 实例的一个调度甘特图

Figure 2　A scheduling Gantt chart of a DFJSP-SDST instance

表 3　基于工厂负荷的工厂分配方法

Table 3　Factory assignment method based on factory load

工件	第 1 次		第 2 次		第 3 次	
	U^1	U^2	U^1	U^2	U^1	U^2
J_1	48.2	53.3	81.1	**53.3**	81.1	**53.3**
J_2	**32.9**	38.1	**32.9**	38.1	**32.9**	38.1
J_3	61.0	48.0	93.9	48.0	**93.9**	101.3

首先计算各工序在各工厂中的平均工时，即平均加工时间和平均准备时间之和；再将各工件所有工序的平均工时相加，即得到各工件在各工厂的预估工时，如表 3 第 2、3 列所示。然后根据 OS 编码中各工件第 1 个工序出现的顺序 (即 2→1→3)，依次为工件选择负荷最小的工厂，每次为一个工件分配工厂后更新该工厂负荷。如第 1 个为工件 J_2，U^1 负荷更低，因此选择在 U^1 加工；由于 U^1 已加工工件 2，因此 U^1 的负荷为 32.9。第 2 个为工件 J_1，将其放到 U^1 或 U^2 加工时，U^1 和 U^2 负荷分别为 81.1 和 53.3。此时对于 J_1，U^2 的负荷更小，因此将 J_1 分配到 U^2。同理将 J_3 分配到 U^1，形成对应的 FA 编码为 211。

确定 MS 编码与此类似，通过计算工序可加工机器上的预估工时，即加工时间和平均准备时间之和，选择负荷最小的机器进行加工，每次选择后更新机器负荷。基于负荷均衡的初始化方法能够尽可能均衡工厂之间、机器之间的负荷，有利于降低最大完工时间，因此可以获得高质量初始种群。为提高种群多样性，初始种群中 80% 个体基于负荷生

成，20% 个体随机生成。

2.3 进化算子

进化算子包括选择、交叉和变异，是遗传算法的关键部分，促使种群向好的方向发展，即实现对问题的迭代寻优。选择算子用于模拟适者生存的原则，能够帮助种群向最优个体收敛，算法使用具有较好收敛性的二元锦标赛选择方法。

交叉算子各部分编码有所不同，FA 编码使用单点交叉，MS 编码使用两点交叉，对于 OS 编码，为保证交叉后解的可行性，使用 POX 交叉[5]，如图 3 所示。首先将工件集分为两个子集，如 J_{s1} = {1}，J_{s2} = {2, 3}，子代 C_1 保留父代 P_1 中 J_{s1} 的基因，剩余空位按照父代 P_2 中 J_{s2} 的顺序依次填入，得到的子代 C_1 各基因出现的次数仍与各工件工序数相同，依然是可行解，且同时继承了两个父代的基因信息。

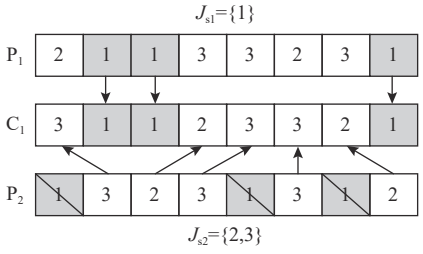

图 3 工序排序 POX 交叉

Figure 3 POX crossover for operation sequencing

FA 编码和 MS 编码均使用单点突变变异，对于 FA 编码，随机选择一个基因，重新分配一个工厂。对于 MS 编码，随机选择一个基因，重新分配一台可加工该工序的机器。OS 编码使用交换变异，随机选择两个位置交换其基因，如图 4 所示。

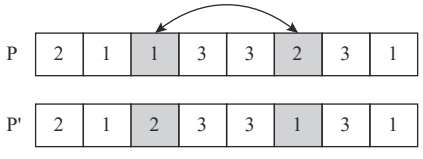

图 4 工序排序交换变异

Figure 4 Swap mutation for operation sequencing

2.4 多重局部扰动策略

DFJSP-SDST 中完工时间最大的工厂称为关键工厂，直接与优化目标相关联。为提高遗传算法的性能，使用多重局部扰动策略进行改进，设计如下 6 个局部扰动算子。

扰动 1 将关键工厂中的一个工件分配到完工时间最小的工厂中 (FA 扰动)；

扰动 2 将关键工厂的工序分成 k 段，每一段第一个基因与其他基因交换 (OS 扰动)；

扰动 3 将关键工厂的工序分成 k 段，重排 k 段的顺序 (OS 扰动)；

扰动 4 随机选择关键工厂中的一个工序，与其他工序交换顺序 (OS 扰动)；

扰动 5 随机选择关键工厂中的一个工序，插入到工序序列的某个位置 (OS 扰动)；

扰动 6 给关键工厂中负荷最高的机器上的一个工序分配新的加工机器 (MS 扰动)。

邻域 2 和 3 改编自 Zhu 等[12]提出的部分成对交换和简单插入，将均匀分为 k 段改为随机点位分段。邻域 4 和 5 改编自 Valente 等[13]提出的邻近非邻近交换和插入，将随机选择一个工序改为选择一个关键工厂的工序。

多重局部扰动过程如下。

步骤 1 随机选择一个局部扰动算子。

步骤 2 根据扰动算子生成一个新解。

步骤 3 若新解优于原解，将新解赋予原解，返回步骤 1；否则，返回步骤 2，直到没有新解生成。

2.5 算法流程

改进遗传算法执行步骤如下。

步骤 1 设置参数。确定种群规模 N、最大迭代次数 G、交叉概率 P_c、变异概率 P_m、精英个数 e。

步骤 2 种群初始化。80% 个体由基于工厂和机器负荷方法生成，20% 个体随机生成。

步骤 3 计算适应度。使用解码方法计算种群中个体的适应度。

步骤 4 选择。使用二元锦标赛选择个体进入交配池，直到达到种群规模 N。

步骤 5 对所选择的个体进行配对，依交叉概率进行交叉。

步骤 6 依变异概率对个体进行变异。

步骤 7 计算子种群个体适应度。

步骤 8 对 10% 最优个体执行多重局部扰动。

步骤 9 保留精英个体。

步骤 10 判断当前代数是否达到最大迭代次数，若达到，则输出最优解；否则，返回步骤 4。

3 计算实验

为了测试改进遗传算法求解 DFJSP-SDST 的性

能，设计基准算例进行计算实验。使用 Matlab R2020b 编程，每个基准独立运行 30 次。

3.1　算例生成

目前尚未有 DFJSP-SDST 的基准算例，参考 DE Giovanni 等[1]将 FJSP 算例扩展为 DFJSP 算例的方法，使用 FJSP-SDST 算例扩展为 DFJSP-SDST 算例。所使用的 FJSP-SDST 算例由 Defersha 等[14-15]设计，除包含序列相关准备时间外，还包括机器释放时间和工序滞后时间，文献[14]的算例记为 Problem1，文献[15]的算例记为 Problem2。将 Problem1 和 Problem2 扩展为 2/3/4/5 个工厂时的情形，每个工厂机器配置相同，工件数增加相应的倍数，不考虑其机器释放时间和工序滞后时间。新生成的算例命名为“原名称_工厂数”，如 Problem1 扩展为 3 个工厂情形生成的新算例，命名为 Problem1_3。所有算例的基本信息如表 4 所示。

表 4　测试算例基本信息

Table 4　Basic information of test cases

算例名称	类型	工厂数	工件数	机器数
Problem1	FJSP-SDST	1	5	4
Problem2	FJSP-SDST	1	5	4
Problem1_2	DFJSP-SDST	2	10	8
Problem2_2	DFJSP-SDST	2	10	8
Problem1_3	DFJSP-SDST	3	15	12
Problem2_3	DFJSP-SDST	3	15	12
Problem1_4	DFJSP-SDST	4	20	16
Problem2_4	DFJSP-SDST	4	20	16
Problem1_5	DFJSP-SDST	5	25	20
Problem2_5	DFJSP-SDST	5	25	20

3.2　参数设置

改进遗传算法包含 5 个关键参数：种群规模 N、最大迭代次数 G、交叉概率 P_c、变异概率 P_m、精英个数 e。为维持精英比例，精英个数取种群规模的 5%，向上取整，即 $e=\lceil N\times 5\% \rceil$，其余参数由正交实验确定。根据经验给定各参数的 4 个水平，如表 5 所示。为找到合适的参数组合，选用正交表 $L_{16}(4^4)$ 进行正交实验，随机选取一个算例 Problem1_4，每组参数独立运行 30 次，取平均值，如表 6 所示。

使用 Minitab 19.1 制作因子水平趋势图，如图 5 所示，得到最优的参数组合为 $N=100$，$G=100$，$P_c=0.9$，$P_m=0.01$。

表 5　参数水平

Table 5　Levels of parameters

参数水平	N	G	P_c	P_m
1	50	50	0.80	0.01
2	100	100	0.85	0.02
3	150	150	0.90	0.03
4	200	200	0.95	0.04

表 6　正交表及实验结果

Table 6　Orthogonal array and experimental results

实验号	N	G	P_c	P_m	平均值
1	50	50	0.80	0.01	2 514.25
2	50	100	0.85	0.02	2 473.58
3	50	150	0.90	0.03	2 473.67
4	50	200	0.95	0.04	2 503.67
5	100	50	0.85	0.03	2 498.58
6	100	100	0.80	0.04	2 468.42
7	100	150	0.95	0.01	2 473.92
8	100	200	0.90	0.02	2 503.67
9	150	50	0.90	0.04	2 491.50
10	150	100	0.95	0.03	2 499.58
11	150	150	0.80	0.02	2 511.17
12	150	200	0.85	0.01	2 477.92
13	200	50	0.95	0.02	2 511.92
14	200	100	0.90	0.01	2 473.25
15	200	150	0.85	0.04	2 498.75
16	200	200	0.80	0.03	2 473.58

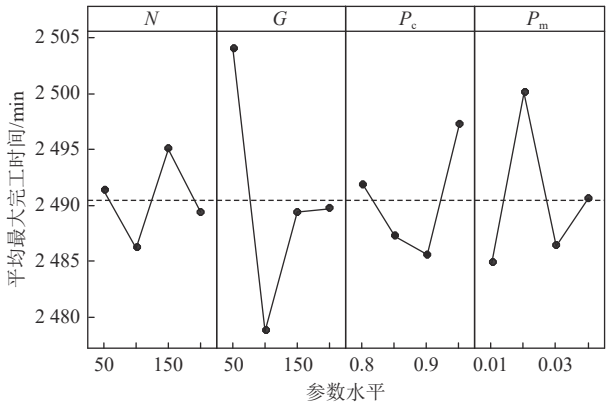

图 5　因子水平趋势图

Figure 5　Trend of factor levels

3.3　种群初始化实验

为验证基于负荷均衡的初始化方法的有效性，以 Problem2_5 为例，分别使用随机初始化、基于工厂负荷初始化、基于工厂和机器负荷初始化 3 种

方法各生成 1 000 个个体，绘制直方图并拟合正态密度函数，如图 6 所示。

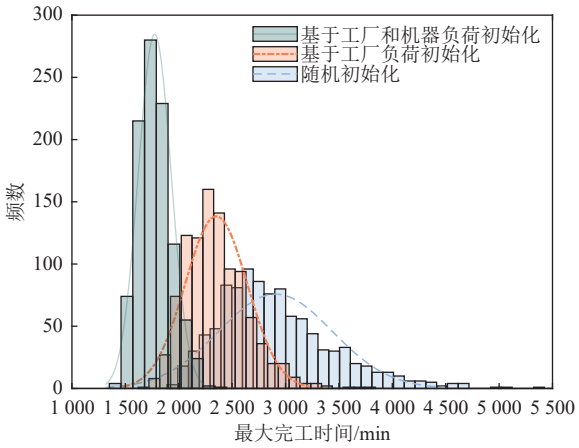

图 6 不同种群初始化方法对比

Figure 6 Comparison of different population initialization methods

从图 6 中可以明显看出，基于工厂和机器负荷的初始化方法得到的初始种群质量最高，基于工厂负荷初始化次之，随机初始化最差，表明基于工厂负荷和基于机器负荷的方法均起到提高初始种群质量的作用。但基于工厂和机器负荷的初始化方法得到的种群相对随机初始化更集中，种群多样性相对差些，因此在算法设计中使用混合初始化方法，均衡种群质量与多样性。

3.4 FJSP-SDST 计算实验

首先测试所提算法在 FJSP-SDST 算例上的性能，并与文献[14]和文献[15]中的算法进行对比，实验结果如表 7 所示。IGA 为完整算法，GA 表示不使用多重局部扰动策略，其他部分与 IGA 相同，Best 表示 30 次计算结果中最优值，Avg 表示 30 次计算结果的平均值。

表 7 FJSP-SDST 实验结果

Table 7 Experimental results of FJSP-SDST min

算例名称	文献[14]	文献[15]	GA		IGA	
			Best	Avg	Best	Avg
Problem1	2 035.0		2 027.5	2 054.3	1 990.0	2 032.3
Problem2		1 637.5	1 341.3	1 383.2	1 331.3	1 362.7

从表 7 中可知，GA 与 IGA 所求得的最优解均优于原文献，且 IGA 在两个算例上的计算结果均优于 GA，初步验证了所提算法及多重局部扰动策略的有效性。值得注意的是，无多重局部扰动策略的 GA 所求得的结果也优于原文献，分析认为是使用了贪婪插入解码策略的原因，在染色体原有调度顺序上尝试贪婪插入，得到更优调度。Problem1 的最优调度甘特图如图 7 所示。

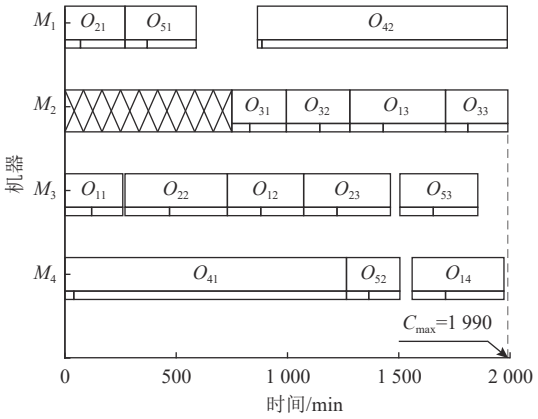

图 7 Problem1 的最优调度甘特图

Figure 7 Gantt chart for the optimal scheduling of Problem1

3.5 DFJSP-SDST 计算实验

为验证所提算法求解考虑序列相关准备时间的 DFJSP 的有效性，对其余算例进行测试，计算结果如表 8 所示。从表 8 数据可知，除了 Problem1_2 算例 GA 和 IGA 求得的最优解相同，其余算例最优解及平均值 IGA 均优于 GA，再次验证所提算法和多重局部扰动策略的有效性。

表 8 DFJSP-SDST 实验结果

Table 8 Experimental results of DFJSP-SDST min

算例名称	GA		IGA	
	Best	Avg	Best	Avg
Problem1_2	2 325.0	2 387.7	2 325.0	2 380.8
Problem2_2	1 127.5	1 243.5	1 065.0	1 193.6
Problem1_3	2 427.5	2 493.0	2 325.0	2 452.2
Problem2_3	1 193.8	1 317.5	1 131.3	1 241.7
Problem1_4	2 447.5	2 538.8	2 427.5	2 496.0
Problem2_4	1 298.8	1 361.5	1 205.0	1 273.9
Problem1_5	2 497.5	2 585.3	2 447.5	2 525.8
Problem2_5	1 288.8	1 388.3	1 175.0	1 294.4

为对比 GA 和 IGA 的收敛性，取表 8 中两算法所得结果相同的算例 Problem1_2 的计算结果，绘制每代 30 次实验的平均最优解收敛曲线如图 8 所示。从图 8 中看出，两算法均在大约 20 代处收敛，但 IGA 的平均最优解均优于 GA，说明 IGA 在求解 DFJSP-SDST 时性能更优。Problem1_2 的最优调度甘特图如图 9 所示。

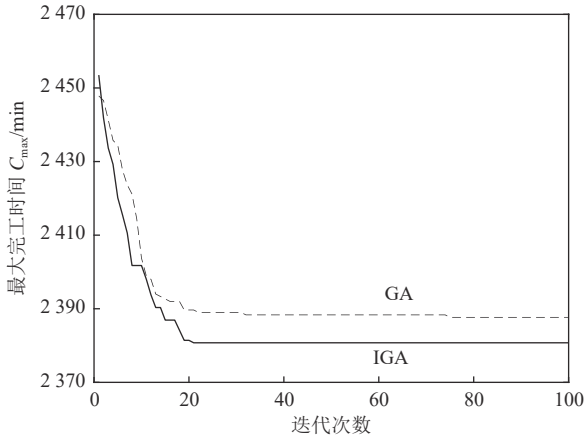

图 8　Problem1_2 的平均最优解迭代曲线

Figure 8　Iteration curve of the average optimal solution for Problem1_2

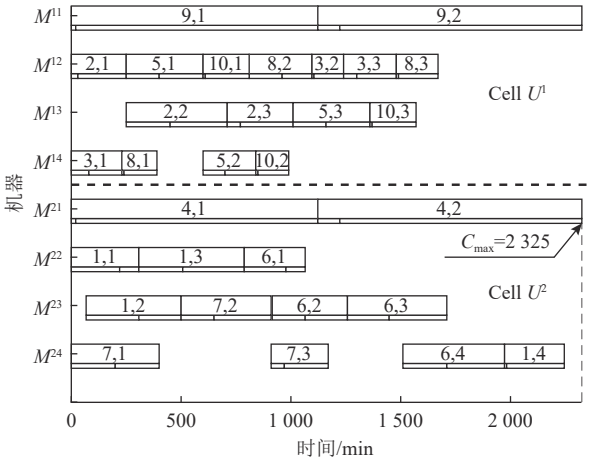

图 9　Problem1_2 的最优调度甘特图

Figure 9　Gantt chart for the optimal scheduling of Problem1_2

4　结束语

本文首次将序列相关准备时间考虑到分布式柔性作业车间调度研究中，提出以最小化最大完工时间为优化目标的混合整数线性规划模型，并提出一种改进遗传算法。实验结果表明，考虑准备时间的贪婪插入解码方法能够得到高质量调度解；考虑负荷均衡的种群初始化方法能够提高初始种群质量；多重局部扰动策略能够提高算法的局部搜索能力。实验验证了改进策略的有效性，表明所提算法能够有效求解 DFJSP-SDST。通过本文研究可以进一步满足实际问题需求，所得调度计划更加贴近实际情况，有助于均衡机器负荷、缩短制造期，为企业节约成本、缩短交期。

在实际生产环境中，可能涉及到刀具、模具甚至是人力资源的使用约束，因此未来可研究考虑准备时间和资源约束的 DFJSP。此外，随着绿色制造理念的兴起，生产能耗和碳排放开始引起决策者的关注，同时优化最大完工时间和总能耗的考虑准备时间的分布式柔性作业车间绿色调度也是今后的研究方向。

参考文献：

[1] DE GIOVANNI L, PEZZELLA F. An improved genetic algorithm for the distributed and flexible job-shop scheduling problem[J]. European Journal of Operational Research, 2010, 200(2): 395-408.

[2] WU M C, LIN C S, LIN C H, et al. Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems[J]. Computers & Operations Research, 2017, 80: 101-112.

[3] 吴秀丽, 刘夏晶. 差分进化算法求解分布式柔性作业车间调度问题[J]. 计算机集成制造系统, 2019, 25(10): 2539-2558.
WU Xiuli, LIU Xiajing. Differential evolution algorithm for solving distributed flexible job shop scheduling problem[J]. Computer Integrated Manufacturing Systems, 2019, 25(10): 2539-2558.

[4] 孟磊磊, 张彪, 任亚平, 等. 求解分布式柔性作业车间调度的混合蛙跳算法[J]. 机械工程学报, 2021, 57(17): 263-272.
MENG Leilei, ZHANG Biao, REN Yaping, et al. Hybrid shuffled frog-leaping algorithm for distributed flexible job shop scheduling[J]. Journal of Mechanical Engineering, 2021, 57(17): 263-272.

[5] LI X Y, XIE J, MA Q J, et al. Improved gray wolf optimizer for distributed flexible job shop scheduling problem[J]. Science China Technological Sciences, 2022, 65(9): 2105-2115.

[6] LUO Q, DENG Q, GONG G, et al. An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers[J]. Expert Systems with Applications, 2020, 160: 113721.

[7] DU Y, LI J, LUO C, et al. A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations[J]. Swarm and Evolutionary Computation, 2021, 62: 100861.

[8] XU W, HU Y, LUO W, et al. A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission[J]. Computers & Industrial Engineering, 2021, 157: 107318.

[9] 张洪亮, 徐公杰, 鲍蔷, 等. 考虑运输时间的分布式柔性作业车间绿色调度[J]. 中国机械工程, 2022, 33(21): 2554-2563.
ZHANG Hongliang, XU Gongjie, BAO Qiang, et al. Distributed flexible job shop green scheduling with transportation time[J]. China Mechanical Engineering, 2022, 33(21): 2554-2563.

[10] 唐红涛, 李悦, 王磊. 模糊分布式柔性作业车间调度问题的求解算法[J]. 华中科技大学学报 (自然科学版) , 2022, 50(6): 81-

88.
TANG Hongtao, LI Yue, WANG Lei. An improved GWO algorithm for fuzzy distributed flexible job shop scheduling problem[J]. Journal of Huazhong University of Science and Technology (Natural Science Edition) , 2022, 50(6): 81-88.
[11] LI R, GONG W, WANG L, et al. Two-stage knowledge-driven evolutionary algorithm for distributed green flexible job shop scheduling with type-2 fuzzy processing time[J]. Swarm and Evolutionary Computation, 2022, 74: 101139.
[12] ZHU J, LI X, WANG Q. Complete local search with limited memory algorithm for no-wait job shops to minimize makespan[J]. European Journal of Operational Research, 2009, 198(2): 378-386.
[13] VALENTE J M S, GONÇALVES J F, ALVES R A F S. A hybrid genetic algorithm for the early/tardy scheduling problem[J]. Asia-Pacific Journal of Operational Research, 2006, 23(3): 393-405.
[14] DEFERSHA F M, CHEN M. A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups[J]. International Journal of Advanced Manufacturing Technology, 2010, 49(1-4): 263-279.
[15] DEFERSHA F M, ROOYANI D. An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time[J]. Computers & Industrial Engineering, 2020, 147: 106605.

(责任编辑: 郑穗华)

(上接第 53 页)

[8] JI B, ZHANG D, YU S S, et al. An exact approach to the generalized serial-lock scheduling problem from a flexible job-shop scheduling perspective[J]. Computers & Operations Research, 2021, 127: 105164.
[9] 邓伟, 邝祝芳, 余绍军, 等. 基于遗传算法的三峡–葛洲坝船闸闸室编排算法[J]. 人民长江, 2016, 47(24): 55-59.
DENG Wei, KUANG Zhufang, YU Shaojun, et al. Study on lock chamber scheduling algorithm of Three Gorges and Gezhouba ship locks based on genetic algorithm[J]. Yangtze River, 2016, 47(24): 55-59.
[10] 曾非凡. 三峡升船机与船闸联合调度闸室编排算法研究[D]. 长沙: 中南林业科技大学, 2016.
[11] 吴小涛, 袁晓辉, Muhammad Adnan Ikram. 基于拟人策略的三峡永久闸室编排新算法[J]. 水电能源科学, 2015, 33(5): 148-151.
WU Xiaotao, YUAN Xiaohui, Muhammad Adnan Ikram. A new allocation algorithm of permanent chamber of Three Gorges Dam based on personification strategy[J]. Water Resources and Power, 2015, 33(5): 148-151.
[12] ZHAO X, LIN Q, YU H. An improved mathematical model for green lock scheduling problem of the Three Gorges Dam[J]. Sustainability, 2019, 11(9): 1-23.
[13] WEI W, OUYANG H, LI S, et al. A modified fireworks algorithm with dynamic search interval based on closed-loop control[J]. Mathematics and Computers in Simulation, 2022, 200(C): 329-360.
[14] 张晓盼. 三峡工程两坝联合通航调度算法研究[D]. 武汉: 华中科技大学, 2007.
[15] ALSATTAR H A, ZAIDAN A A, ZAIDAN B B. Novel meta-heuristic bald eagle search optimisation algorithm[J]. Artificial Intelligence Review, 2020, 53(6): 2237-2264.

(责任编辑: 郑穗华)

(上接第 77 页)

[17] 张蓝天, 石宇强. 考虑特征学习的 IPSO-LSTM 晶圆加工周期预测[J]. 工业工程, 2023, 26(3): 143-150.
ZHANG Lantian, SHI Yuqiang. Wafer cycle time prediction of IPSO-LSTM considering feature learning[J]. Industrial Engineering Journal, 2023, 26(3): 143-150.
[18] LUO J, ZHANG X. Convolutional neural network based on attention mechanism and Bi-LSTM for bearing remaining life prediction[J]. Applied Intelligence, 2022, 52: 1-16.
[19] PHM Society. PHM data challenge 2010[C/OL]//Annual Conference of the Prognostics and Health Management Society 2015. (2015-10-18) . Coronado, California: PHM Society. https: // www. phmsociety. org/ competition/phm/10 .
[20] WANG L, LONG J, LI X, et al. Industrial units modeling using self-attention network based on feature selection and pattern classification[J]. Chemical Engineering Research and Design, 2023, 200: 176-185.

(责任编辑: 刘敏仪)