一种改进迭代贪婪算法求解跨工厂运输的分布式柔性作业车间调度问题

唐敦兵 崔晓哲 黄 林* 张泽群 杨汝舟 蔡祺祥
南京航空航天大学机电学院,南京,210016

摘要:针对工件可跨工厂运输的分布式柔性作业车间调度问题,以最小化最大完工时间为优化目标,提出了一种改进的迭代贪婪(IIG)算法。IIG算法以解构-重构机制为核心框架:在解构阶段,通过移除关键工件以缩短最大完工时间;在重构阶段,设计了一种基于机器空闲的贪婪插入策略(GIMI)。该策略考虑了当前工序插入导致的调度延迟、剩余工序调度的灵活性以及转运时间3种关键因素。通过融合3种因素设计了全局评估函数进行评估,从而确定工序的插入位置,打破了传统贪婪策略的局部最优限制。此外,基于GIMI还设计了双梯度邻域结构,实现了全局与局部搜索的协同。通过与4种先进算法的对比实验,验证了所提算法的有效性。

关键词:分布式柔性作业车间;迭代贪婪算法;跨工厂运输;最大完工时间
中图分类号:TP301
DOI:10.3969/j.issn.1004-132X.2026.11.018 **开放科学(资源服务)标识码(OSID)**:

An Improved Iterative Greedy Algorithm for Solving Distributed Flexible Job-shop Scheduling Problem with Cross-factory Transportation

TANG Dunbing CUI Xiaozhe HUANG Lin* ZHANG Zequn YANG Ruzhou CAI Qixiang
College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016

Abstract: To solve the distributed flexible job-shop scheduling problem with cross-factory job transportation, an improved iterative greedy (IIG) algorithm was proposed with the optimization objective of minimizing the makespan. The IIG algorithm used a destruction-reconstruction mechanism as its core framework. In the destruction phase, critical jobs were removed to reduce the makespan. In the reconstruction phase, a greedy insertion strategy based on machine idle (GIMI) was designed, considering the scheduling delay caused by inserting the current operation, the flexibility of scheduling the remaining operations, and the transportation time. A global evaluation function was designed by integrating these three factors to determine the insertion position of the operation and overcome the local-optimum limitation of traditional greedy strategies. Additionally, a dual-gradient neighborhood structure was then developed based on GIMI to balance global and local search. The effectiveness of the proposed algorithm was verified through comparative experiments with four advanced algorithms.

Key words: distributed flexible job-shop; iterative greedy algorithm; cross-factory transportation; makespan

0 引言

现代制造业正面临个性化需求增长与市场波动加剧的双重挑战。单一工厂虽能借助柔性设备提高局部响应能力,但仍受产能与资源约束,在订单频繁变化、交付周期压缩的情况下,难以快速调整产出或将订单转移到空闲产线上。为此,企业逐步采用分布式制造,通过多工厂协同实现资源共享与能力互补,以增强系统鲁棒性与整体柔性[1-2]。然而,该模式也带来了分布式调度难题,其中分布式柔性作业车间调度问题(distributed flexible job-shop scheduling problem, DFJSP)是典型代表。DFJSP是柔性作业车间调度问题(FJSP)的扩展,涉及n个工件在多个工厂进行加工,每个工厂属于一个柔性作业车间[3]。

近年来,DFJSP受到广泛关注。MENG等[4]以最小化最大完工时间为目标,提出4种线性规划模型和1种约束规划模型求解DFJSP;LI等[5]提出一种基于Pareto的混合禁忌搜索算法,求解

收稿日期:2025-09-04
基金项目:国家自然科学基金(52305539);江苏省自然科学基金(BK20230880);中央高校基本科研业务费专项资金(NS2024033)

以完工时间、工作负载等为目标的DFJSP模型；魏光艳等[6]针对DFJSP，提出一种多维模因算法来优化完工时间与总能耗。此外，张洪亮等[7]在DFJSP中考虑运输时间，设计了一种改进的非支配排序遗传算法，实现了完工时间与总能耗双目标优化；DU等[8]针对考虑起重机运输的DFJSP，提出一种混合分步估计算法来优化总工期与能耗；余佳林等[9]提出一种改进的延迟接受爬山算法，求解了考虑工件在机器间运输时间的DFJSP。

上述研究虽取得显著进展，但大多默认工件在单一工厂内完成全部加工，忽略了跨工厂协同生产的现实需求。在分布式制造中，不同工厂常配置相似的加工单元，当部分工厂出现生产瓶颈或设备故障时，可将剩余工件转移至其他工厂的同类资源上加工，从而实现跨工厂并行生产与负载均衡。如在航空结构件制造中，同类铣削或钻孔等工序可由不同工厂的并行产线承担，企业可依据产能与交期动态分配任务[10]。在上述场景中，工件的跨工厂运输不可避免，且工厂间转运时间通常长于厂内机器间运输时间，对完工时间影响显著。基于此，本文以最小化最大完工时间为目标，研究工件可跨工厂运输的DFJSP（DFJSP with cross-factory transportation，DFJSPT）。

迭代贪婪（iterative greedy，IG）算法因其高效的解构-重构机制在调度领域应用广泛[11]：在解构阶段删除部分工件，在重构阶段采用贪婪策略将删除的工件重新插回，以实现调度优化。相关研究表明，IG算法在流水车间问题（flow shop problem，FSP）中的应用已取得显著成效[12-15]，但在FJSP中应用较少，原因在于FJSP中允许工序非连续加工，重构阶段需逐个工序进行贪婪插入操作，缺乏高效的重构策略，且传统IG算法的插入决策遵循“局部最优”原则，仅关注当前工序的即时成本最小化，而忽略后续工序的调度灵活性，尤其当插入位置临近完工时间时，剩余工序的调度空间会被大幅压缩，难以保证整体调度质量。

鉴于上述分析，本文针对DFJSPT提出了一种改进的迭代贪婪（IIG）算法。首先，设计了一种基于机器空闲的贪婪插入策略（greedy insertion strategy based on machine idle，GIMI），通过融合3种关键指标来构建全局评估函数，实现了工件的全局最优插入；然后，基于GIMI设计了双梯度邻域结构，实现了工件级全局搜索与工序级局部搜索的协同优化；最后，在40个不同规模的基准实例上与4种先进算法进行对比测试，验证了所提算法的有效性。

1 问题描述

DFJSPT可描述为将工件集合$J=\{J_1,J_2,\cdots,J_n\}$分配到工厂集合$F=\{F_1,F_2,\cdots,F_f\}$内加工，每个工厂F_s包含机器集合$M=\{M_{s\text{-}1},M_{s\text{-}2},\cdots,M_{s\text{-}m}\}$，每个工件$J_i$由工序序列$O_i=\{O_{i\text{-}1},O_{i\text{-}2},\cdots,O_{i\text{-}k}\}$组成，每道工序$O_{i\text{-}j}$均可柔性分配到工厂$F_s$中兼容的机器$M_{s\text{-}a}$上加工，其加工时间$P_{ijsa}$是预先确定的。若相邻工序分配到不同工厂或机器上加工时，则需消耗固定的转运时间T_{trans}，跨工厂转移时间为$T_{\text{F}st}$，厂内跨机器运输时间为T_{Mal}。表1列出了相关参数符号。

表1 DFJSPT参数表

Tab.1 The parameter table of DFJSPT

参数符号	含义
J	工件集合；$J=\{J_1,J_2,\cdots,J_n\}$
n	工件总数
$O_{i\text{-}j}$	工件i的第j个工序；$i=1,2,\cdots,n$；$j=1,2,\cdots,k$
k	工件J_i的工序总数
F	工厂集合；$F=\{F_1,F_2,\cdots,F_f\}$
f	工厂总数
M	机器集合；$M=\{M_{s\text{-}1},M_{s\text{-}2},\cdots,M_{s\text{-}m}\}$
m	机器总数
$M_{s\text{-}a}$	工厂F_s中的第a台机器；$s=1,2,\cdots,f$；$a=1,2,\cdots,m$
P_{ijsa}	工序$O_{i\text{-}j}$在$M_{s\text{-}a}$上的加工时间
S_{ijsa}	工序$O_{i\text{-}j}$在$M_{s\text{-}a}$上的加工开始时间
C_{ijsa}	工序$O_{i\text{-}j}$在$M_{s\text{-}a}$上的加工结束时间；$C_{ijsa}=S_{ijsa}+P_{ijsa}$
T_{trans}	工件的转运时间
$T_{\text{F}st}$	工件从工厂F_s到F_t的转运时间（若$s=t$，则$T_{\text{F}st}=0$）
T_{Mal}	工件从机器$M_{s\text{-}a}$到$M_{s\text{-}l}$的运输时间（若$a=l$，则$T_{Mal}=0$）
$C_{\max}$	所有工件的最大完工时间
X_{ijsa}	决策变量；若工序$O_{i\text{-}j}$在工厂F_s的机器$M_{s\text{-}a}$上加工时为1，否则为0
$Y_{i(h,j)}$	决策变量；当且仅当工序$O_{i\text{-}j}$在工序$O_{i\text{-}h}$之后加工（$j>h$）时为1，否则为0
V	极大值常数

求解DFJSPT需满足如下约束：①所有工厂及机器均零时可用；②单工序同一时间只能在一个工厂内加工，单台机器在任意时刻最多处理一道工序；③工序加工起始到结束过程不能中断；④同一工件需按照既定工艺路线加工；⑤跨工厂或机器的相邻工序需等待转运结束后再开始加工，同机器上同一工件的相邻工序加工无需转运时间；⑥工件跨工厂转移时间和跨机器运输时间

均预先确定。

基于上述约束，可建立如下数学模型：

$$\min C_{\max}=\max\{C_{ijsa}\mid\forall i,j,s,a\} \tag{1}$$

$$\sum_{s=1}^{f}\sum_{a=1}^{m}X_{ijsa}=1 \tag{2}$$

$$S_{ijsa}\geqslant(S_{ihsa}+P_{ihsa}+T_{\text{trans}})Y_{i(h,j)}$$

$$T_{\text{trans}}=\begin{cases}0 & s=t\text{且}a=l\\ T_{Mal} & s=t\text{且}a\neq l\\ T_{Fst} & s\neq t\end{cases} \tag{3}$$

$$\left.\begin{aligned}C_{ihsa}&\leqslant S_{ijtl}+V(1-Y_{i(h,j)})\\ C_{ijtl}&\leqslant S_{ihsa}+V(Y_{i(h,j)})\end{aligned}\right\} \tag{4}$$

$$S_{ijsa}\geqslant 0\qquad\forall i,j,s,a \tag{5}$$

其中，式(1)为本文的目标函数；式(2)为工序分配约束，即每道工序必须分配到某个工厂的某台机器上加工；式(3)为工艺顺序约束；式(4)为机器唯一性约束，即同一机器上的工序时间不重叠；式(5)为开工时间非负约束。

表2展示了一个DFJSPT的5×2×5实例，表示有5个工件可分配至2个工厂共5台机器上加工。每个工件有多道工序，工序分配具有一定的柔性，每道工序存在多台可加工机器。如工件J_1的第一道工序$O_{1\text{-}1}$具有部分柔性，可分别选择在工厂F_1的$M_{1\text{-}1}$、$M_{1\text{-}2}$和工厂F_2的$M_{2\text{-}2}$、$M_{2\text{-}3}$上加工。

表2　DFJSPT 5×2×5实例

Tab.2　The 5×2×5 instance of DFJSPT　　min

工件	工序	F_1		F_2		
		$M_{1\text{-}1}$	$M_{1\text{-}2}$	$M_{2\text{-}1}$	$M_{2\text{-}2}$	$M_{2\text{-}3}$
J_1	$O_{1\text{-}1}$	15	**17**		21	16
	$O_{1\text{-}2}$	**16**		14	19	
J_2	$O_{2\text{-}1}$	**15**				13
	$O_{2\text{-}2}$		**13**	10	15	
	$O_{2\text{-}3}$	12		**11**	9	13
J_3	$O_{3\text{-}1}$	10		**11**		12
	$O_{3\text{-}2}$	11	9	**14**	14	15
	$O_{3\text{-}3}$		12	9		**13**
J_4	$O_{4\text{-}1}$	8	7			**9**
	$O_{4\text{-}2}$		14	21	**15**	17
	$O_{4\text{-}3}$	20	**15**	16		
J_5	$O_{5\text{-}1}$	**12**			15	
	$O_{5\text{-}2}$		14		**13**	11
	$O_{5\text{-}3}$	15	**13**			12

在该实例中，将5个工件随机分配到机器上加工，表2粗体部分为每道工序所在加工机器对应的加工时间，调度方案的甘特图见图1，其中$C_{\max}$=69 min。以工件4为例，$O_{4\text{-}1}$与$O_{4\text{-}2}$在同工厂的不同机器上加工，运输时间为4 min；$O_{4\text{-}2}$与$O_{4\text{-}3}$在不同工厂内加工，转运时间为8 min。

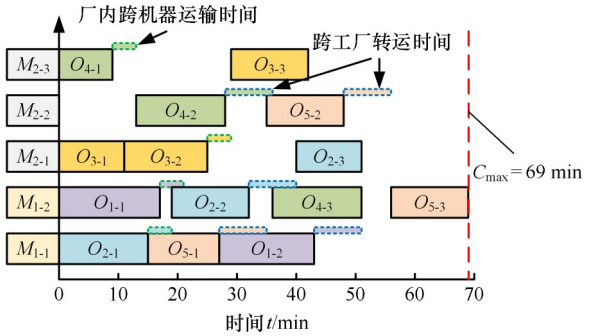


图1　DFJSPT初始调度方案

Fig.1　Initial scheduling plan for DFJSPT

2　基于机器空闲的贪婪插入策略

迭代贪婪算法是一种高效的启发式算法[16]，其核心在于贪婪重构策略——在每一步都选择当下最优的方案。在FJSP中，IG算法更多的是针对工件层的解构和重构[17]。由于每个工件包含多道工序，且每道工序有多个可加工机器，在贪婪策略下，当前工序的插入极易陷入局部最优，从而造成剩余工序调度空间的压缩，因此，如何在保证当前工序高效插入的同时保留后续调度的灵活性，是IG算法亟需解决的问题。

为此，本文设计了一种基于机器空闲的贪婪插入策略(GIMI)。首先，设计三属性评估指标，初步筛选工序的可插入位置；其次，融合三种指标构建全局评估函数，进一步评估筛选最优的插入位置。上述两者构成GIMI主体，可实现工件各工序的全局寻优插入，GIMI框架如图2所示。

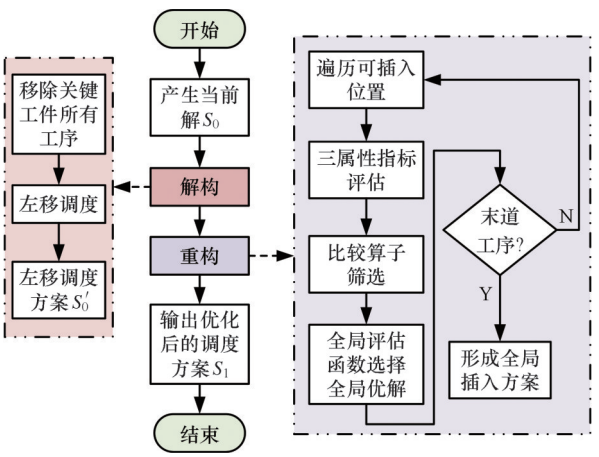


图2　GIMI整体框架

Fig.2　Overall framework of GIMI

2.1　解构

解构阶段下需要在当前调度中移除部分工件。图1中从起始到终止节点的最长路径为关键路径，而删除该路径上的工件可直接压缩最大完工时间，为后续优化创造条件。如图3所示，以删除关键工件J_5为例，将剩余工序左移生成左

移调度方案，此时最大完工时间由69 min缩减为51 min。

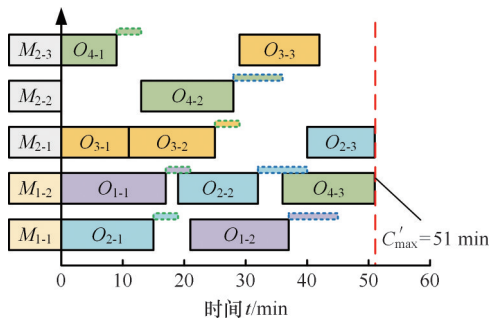


图3 左移调度方案

Fig.3 Left shift scheduling scheme

2.2 重构

重构时需要基于GIMI将移除的工件重新整合至左移调度方案中，该过程在满足各工件工艺顺序的同时还应考虑如下目标：①为各工序选择合适的机器加工；②通过评估筛选出工序在机器上的可加工位置；③综合考虑工件转运时间因素，以确保最大完工时间的最小化变动。

在上述过程中，基于全局视角对潜在插入位置进行评估并完成最优方案筛选，是GIMI的核心决策环节。

2.2.1 评估插入位置

工件各工序在选择插入位置时，需要考虑两个关键问题：一是从可选机器中确定加工位置，二是后续工序的转运时间。其中，转运操作仅在相邻工序使用不同机器加工时产生，是影响最大完工时间的关键参数，需要进行分类评估：安排初始工序时直接进入对应机器加工队列，转运时间为零；安排非初始工序时需考虑当前工序与前序工序是否处于同一机器，不同则考虑转运时间，否则为零。

2.2.1.1 初始工序评估

按照工件的工艺顺序，首先安排J_5的第一道工序$O_{5\text{-}1}$插入。由表2可知，$O_{5\text{-}1}$可选择在$M_{1\text{-}1}$、$M_{2\text{-}2}$上加工，加工时间分别为12、15 min。如图4所示，根据当前机器的加工情况，$O_{5\text{-}1}$共有5个可插入位置。由图5可以直观发现，位置③和⑤的空闲时间长于$O_{5\text{-}1}$的加工时间，对当前调度的完工时间影响最小，其余插入位置会导致调度的延迟。为此，根据传统的贪婪策略[18]会从位置③或⑤中选择一个插入位置。然而，选择位置③或⑤会显著压缩剩余工序（$O_{5\text{-}2}$和$O_{5\text{-}3}$）的调度空间，易导致算法陷入局部最优。

针对上述问题，本文设计了改进的插入策略。在工件插入阶段评估当前最优的同时考虑后续工序的插入区间，通过全局评估确定工序插入

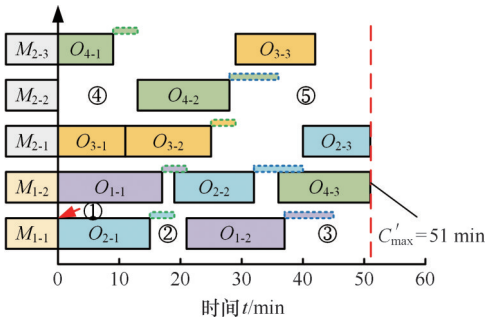


图4 $O_{5\text{-}1}$可选择的位置

Fig.4 Selectable positions of $O_{5\text{-}1}$

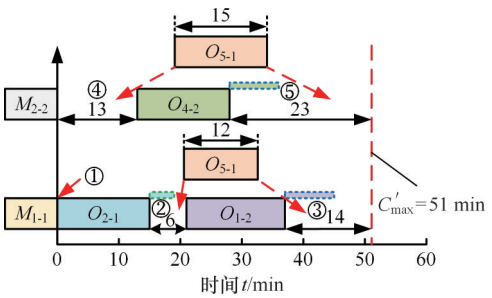


图5 贪婪策略下的$O_{5\text{-}1}$插入

Fig.5 Insertion of $O_{5\text{-}1}$ under the greedy strategy

位置。基于此，提出了两种评价指标用于初始工序的插入评估：

1）当前节点延迟D。由图4已知，$O_{5\text{-}1}$在当前调度方案中有5个可插入位置，若逐个尝试插入并计算最大完工时间，则工作量较大，因此设计延迟指标来评估当前插入位置的优劣。延迟值D越小表明越优，当D为0时，说明当前位置可以完全放置该工序，反之D越大则越差，其表达式为

$$D_{jp}=\max([P_{ijsa}-(E_p-S_p)],0) \tag{6}$$

其中，D_{jp}为工序j在位置p的延迟，S_p、E_p分别为插入位置p区间的起始和结束时刻。以$O_{5\text{-}1}$插入位置①为例，由于位置①起止区间为[0，0]，即S_p=E_p=0，$O_{5\text{-}1}$的加工时间P_{5111}=12 min，故当前节点延迟D_{11}=12 min。同理，可得位置②～⑤的延迟值，如表3所示。

表3 针对$O_{5\text{-}1}$的插入情况

Tab.3 Regarding the insertion situation of $O_{5\text{-}1}$

可插入位置p/机器M	当前节点延迟D_{jp}/min	后续工序可行区间I_p/跨度L_{jp}	是否保留
①[0，0]/$M_{1\text{-}1}$	12	[0，51]/51	是
②[15，21]/$M_{1\text{-}1}$	6	[21，51]/30	否
③[37，49]/$M_{1\text{-}1}$	0	[49，51]/2	否
④[0，13]/$M_{2\text{-}2}$	2	[13，51]/38	是
⑤[28，43]/$M_{2\text{-}2}$	0	[43，51]/8	是

2）后续工序可行跨度L。为保证最大完工时间的最小化变动，以当前最大完工时间C'_{max}为界限，设计可行跨度指标评估后续工序可插入范围

的优劣。L表示后续工序可插入区间的长度，受到当前工序插入位置的影响，理论上L越大，意味着后续工序调度灵活性越高，其表达式为

$$L_{jp}=C'_{\max}-\min(E_p,(S_p+P_{ijsa})) \tag{7}$$

其中，L_{jp}为插入位置p后工序j后续工序的可行跨度，$\min(E_p,(S_p+P_{ijsa}))$表示后续工序可插入区间的理论起始时刻，若工序j在区间p内完成($S_p+P_{ijsa}\leqslant E_p$)，则后续工序最早可从时刻$S_p+P_{ijsa}$开始安排；若超出区间($S_p+P_{ijsa}>E_p$)，则为避免与后续确定工序出现时间冲突，保守地将后续工序的最早起始时刻设为原区间的结束时刻E_p。故min函数确保了无论何种情况下，L_{jp}始终是一个可行且保守的评估值，为解码顺序提供可靠依据。以图5为例，$C'_{\max}$=51 min为当前调度的最大完工时间，由于位置①起止区间为[0,0]，因此后续工序可行区间$I_1\in[0,51]$，可得L_{11}=51。类似地可得出其余节点的后续可行跨度，如表3所示。

由于工序具有加工柔性，每次安排都存在多个插入位置，因此需要筛选以减少冗余，上述D、L两种指标的协同特性与非支配排序思想[19]相契合。本文采用该思想来保留插入位置，若某个插入位置被其他位置所支配，则丢弃；否则将其保留。依托两种指标和非支配排序思想定义比较算子，其表达式如下：

$$D_{jx}<D_{jy}\text{且}L_{jx}\geqslant L_{jy}$$
$$D_{jx}=D_{jy}\text{且}L_{jx}>L_{jy}$$

以上两点满足其一即认定插入位置x优于y，因此可得出位置④支配位置②，位置⑤支配位置③，位置①与位置④和⑤互不支配。根据相互支配关系可决定各位置的保留权，如表3所示。

2.2.1.2　非初始工序评估

安排非初始工序时，需结合相邻工序的机器对应关系考虑是否执行转运操作，因此在双属性指标的基础上，设计转运时间相关度R指标协同评估。根据约束条件，自插入工件的第2道工序起，相邻工序在不同机器加工时存在转运操作，其中工厂间转移时间长于机器间运输时间，为此设计三个层次的转运时间相关度，以进一步评估非初始工序插入位置的优劣。R值越小，表明所选插入位置越优。定义：R=0时，当前工序与前序工序加工于同机器；R=1时，当前工序与前序工序加工于同工厂非同机器；R=2时，当前工序与前序工序加工于非同工厂。

由表2可知，$O_{5\text{-}2}$可在$M_{1\text{-}2}$、$M_{2\text{-}2}$、$M_{2\text{-}3}$三台机器上加工，根据D、L指标评估可得出$O_{5\text{-}2}$有$M_{1\text{-}2}$[0,0]、$M_{2\text{-}2}$[0,13]、$M_{2\text{-}3}$[9,20]三个较优的可插入位置。当$O_{5\text{-}1}$选择$M_{1\text{-}1}$时，由R评估可知$O_{5\text{-}2}$选择$M_{1\text{-}2}$[0,0]为优；同理，当$O_{5\text{-}1}$选择$M_{2\text{-}2}$时，由R评估可知$O_{5\text{-}2}$选择$M_{2\text{-}2}$[0,13]和$M_{2\text{-}3}$[9,20]为优。具体结果需要进一步评估筛选。

2.2.2　**全局评估函数**

每道工序经过三种评估指标筛选后仍会保留多个插入位置，若遍历保留的位置来评估构建工件的插入方案则过于繁琐，因此，本文设计全局评估函数V对保留的插入位置开展进一步评估。V值越小意味着当前插入方案越优，其表达式如下：

$$V_{ijp}=S_{jp}+U_{jp}+T_{ij}R \tag{8}$$

$$S_{jp}=\sum_{q=1}^{j}D_{qp} \tag{9}$$

$$U_{jp}=|\min(L_{jp}-\sum_{q=j+1}^{k}\min P_{iq},\ 0)| \tag{10}$$

$$T_{ij}=\begin{cases}0 & j=1\\ 1 & j\neq 1\end{cases} \tag{11}$$

其中，V_{ijp}为$O_{i\text{-}j}$在插入位置p的评估值；S_{jp}为$O_{i\text{-}j}$及其前序工序节点延迟之和；U_{jp}为$O_{i\text{-}j}$后续所有工序的最小偏移指数之和；L_{jp}为工序j后续工序的可行跨度；$\sum_{q=j+1}^{k}\min P_{iq}$表示$O_{i\text{-}j}$后续工序集合中(从工序$j$+1到末道工序$k$)，每道工序的最小加工时间之和，用于计算后续工序非负的最小偏移指数；T_{ij}为工序决策变量。以$O_{5\text{-}1}$为例，评估$M_{1\text{-}1}$[0,0]时，S_{11}=12，由于剩余两道工序的最小加工时间分别为11和12，因此U_{11}=|min(51−(11+12),0)|=0，由于当前工序j=1，故T_{i1}=0，可得V_{511}=12。

2.3　全局寻优插入

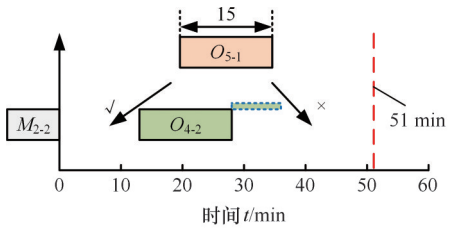
GIMI旨在实现全局寻优插入，本文设计三属性评估指标以及全局评估函数，通过对比各工序可插入位置的评估值择优保留，算法在运行时不仅考虑当前工序的最优插入，同时考虑后续工序的评估，以确保全局最优解的产生。GIMI伪代码如下面算法1所示(其中4～14行表示关键工件的完整重构过程，5～9行表示对各工序的初步评估，10～13行表示对各工序的全局评估：

算法1 GIMI伪代码

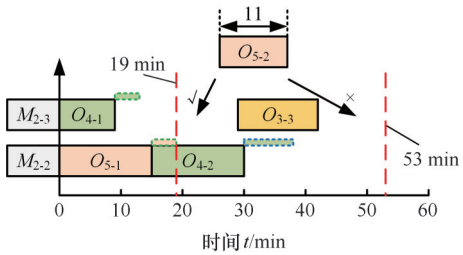
```
1   输入：初始调度方案S0，关键工件Ji的工序集{Oi-1，Oi-2，…，Oi-k}，列表L0，L1
2   输出：最优插入方案BIO
3   S0'←删除关键工件Ji，左移调度
4   While (Oi-j非末道工序) do
5     If (j=1)                                  //初始工序
6       Oi-1，p/M←(D，L)∪比较算子
7     Else                                      //非初始工序
```

$O_{i\text{-}j}$, p/M←(D, L, R)∪比较算子
End
L_0←筛选较优位置{$O_{i\text{-}j}$, p/M}存入列表
L_1←评估函数V,从L_0中筛选,存入新列表
j←j+1
L_0←清空列表L_0
End
BIO←选择$O_{i\text{-}j}$最优位置∈L_1,生成最优插入方案
Return BIO

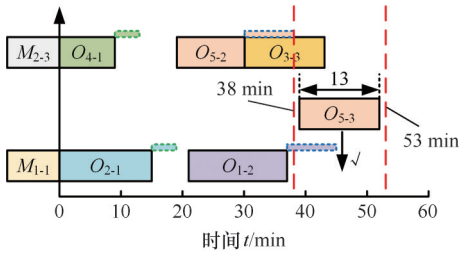
图6展示了关键工件J_5各工序$O_{5\text{-}1}$、$O_{5\text{-}2}$、$O_{5\text{-}3}$的插入情况。



(a)$O_{5\text{-}1}$插入



(b)$O_{5\text{-}2}$插入



(c)$O_{5\text{-}3}$插入

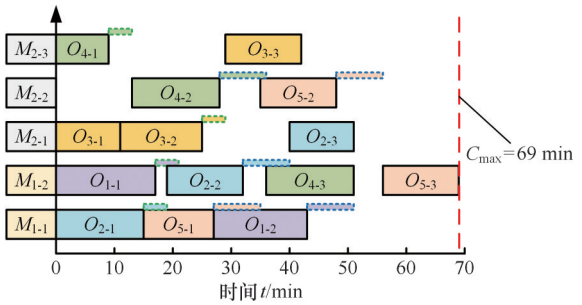
图6 J_5各工序插入情况

Fig.6 Insertion situation of each operation of J_5

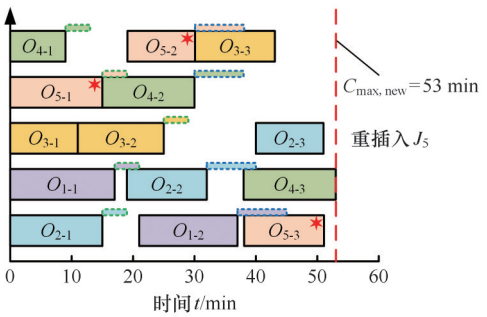
图7为针对J_5各工序(星号标记)优化前后的调度甘特图,对比优化前的最大完工时间C_{max}=69 min,可得优化后的最大完工时间$C_{max,new}$缩短了16 min。这种全局评估策略有效平衡了插入过程的深度探索与广度搜索,最终筛选出综合性能最优的工序序列。

3 IIG求解DFJSPT

调度算法的设计需要在全局寻优与局部搜索之间达到平衡,为此,本文提出了改进的迭代贪婪(IIG)算法以求解DFJSPT。在重构阶段,基于GIMI设计了一种双梯度邻域结构,通过工件级循



(a)优化前调度甘特图



(b)优化后调度甘特图

图7 J_5经过一次解构和重构的对比

Fig.7 J_5 has undergone a comparison between deconstruction and reconstruction

环搜索和工序级局部选择生成邻域解。双梯度结构包含G1、G2策略,集成了工件转运时间因素,确保算法在搜索过程中能动态维护全局最优解,从而进一步增强算法性能。IIG算法涵盖编解码、初始化、解构-重构搜索、迭代寻优等模块,其整体框架如下面算法2所示:

算法2 IIG整体框架

输入:工厂数据:F,机器数据:M,工件数据:J,列表$LIST$
输出:最优调度方案:$Best$
S_0←产生初始解(F, M, J);
当前迭代次数N=1,最大迭代次数I_{max}
While (N≤I_{max}) do
  [J_c, J_d]←选择关键工件J_c、G1(J—J_c)　//G1见3.3节
  For j = [J_c, J_d]
    S_0'←移除工件并左移
    S_0'←评估插入位置(D, L, R, V)
    S_j=G2(J, S_1')　// G2见3.3节
    $LIST$←将S_j存入空列表
    If makespan(S_j)<makespan($Best$)　//更新
      $Best$←S_j,
      S_0←$Best$
    Else
      S_0←从$LIST$中随机选择一个解
    End
  End
  N←N+1
End
Return $Best$

3.1　编码与解码

在算法框架中，编码与解码是首要问题。针对DFJSPT特有的工序、机器和工厂三维特性，本文采用了三维编码结构（图8）：①工序排序（OS），采用基于工序的编码方式，数值1～5代表工件J_1～J_5；②机器选择（MS），采用基于机器的编码方式，编码序列对应各工件工序的执行顺序，其中每个编码值代表对应工序所选用的机器号，且预绑定了不同的工厂（其中1、2分别表示工厂F_1的机器$M_{1\text{-}1}$和$M_{1\text{-}2}$；3～5表示工厂F_2的机器$M_{2\text{-}1}$～$M_{2\text{-}3}$）；③工厂分配（FA），建立机器-工厂的映射关系（其中1、2分别表示工厂F_1和F_2），明确每台选定机器的所属工厂。

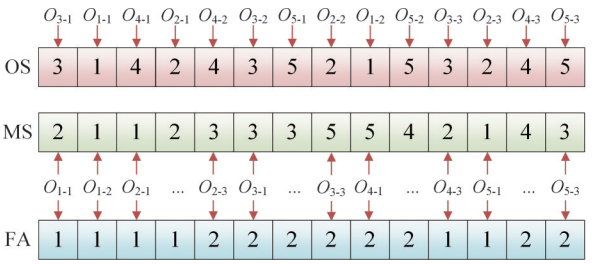


图8　三维编码方式

Fig.8　Three-dimensional encoding method

解码分如下两个阶段进行：①依据OS确定工序优先级序列；②根据MS和FA协同计算每道工序的加工起止时间。本文采用全主动解码方法，旨在维持编码优先级顺序的同时优化工序时序，从而缩短算法收敛时间。

3.2　初始化

DFJSPT基于编码-解码生成初始解。初始调度方案的构建过程包含如下两个环节：①工序排序，在确保工件各工序无时序冲突的前提下，采用随机排列方式生成候选序列；②机器分配，将工序任务分配至不同工厂中具备加工兼容性的机器上。三维编码机制与全主动解码策略的协同作用能提高初始解质量，为后续优化奠定基础。

3.3　解构-重构搜索

解构-重构是算法的核心。IIG双梯度邻域结构的设计能够进一步降低最大完工时间，双梯度邻域结构具体包括：

1）第一梯度（G1）：工件级循环搜索插入。在GIMI中移除关键工件的同时另随机选择$(n-1)/2$个剩余工件（n为工件总数）预移除，后续将所选择的工件依次进行重插入操作。

2）第二梯度（G2）：工序级局部择优插入。在进行工件非初始工序重插入操作时，在所筛选的较优可插入位置中以一定概率将后续工序对照其前序工序按照同机器优先、同工厂次级、非同工厂末级的优先级安排加工，以此选择工件转运时间因素影响较小的插入位置，实现进一步寻优。

融入G1、G2后整体操作步骤如下：

1）根据初始调度方案选择一个关键工件，同时G1随机选择$(n-1)/2$个剩余工件预移除。

2）将所选工件从调度方案中移除，执行左移调度生成新调度方案。

3）按照工序顺序，根据三属性评估指标和评估函数对被移除工件的每道工序逐一判断，从中选择全局最优的可插入位置。

4）通过G2对非初始工序所选择的可插入位置进一步寻优，选择转运时间影响较小的位置插入以进一步降低最大完工时间，生成新的可行解。

5）完成单次插入操作后进行寻优更新，对G1选定的剩余工件序列依次执行步骤2）～步骤4）。在优化过程中持续对比新解与当前最优解，若新解更优，则替换并更新调度方案；否则维持当前最优，直至遍历所有候选工件。

3.4　迭代寻优

尽管IIG算法在单次运行时可获得全局最优解，但由于G1策略的随机性可能导致目标函数过早收敛，从而错失潜在更优解。为此引入迭代寻优机制，通过扩展搜索空间来提高全局寻优能力。在预设迭代次数内，设置空列表并存入当前最优解，在后续迭代中持续对比新解与当前最优解。若新解质量远低于当前解则排除；若新解质量略低于当前解则共同存入列表，并随机从列表中选取一个解继续迭代；若新解质量优于当前解则更新最优解。迭代终止后，选择列表中的最优解作为全局最优解。

4　实验对比及结果

针对DFJSPT，本文通过实验来评估所提IIG算法的性能，算法采用MATLAB-2023b编程实现，测试环境为Intel Core i7，2.20 GHz，16 GB RAM。为减小计算过程中的随机误差，对每个基准样本均进行10次独立重复试验。

4.1　基准算例及对比算法

本文采用文献[20]设计的40个基准算例（DFM01～DFM40）。该组算例由FJSP基准算例扩展而来，按照工厂数量、机器数量、每道工序在单个工厂内可用机器数量3项参数划分为3种规模，取值依次为[2，4，6]、[6，8，10]、[2，3，4]，如表4所示。此外，考虑工件转运时间的影响，设定不同工厂间转运时间为60 min，同工厂不同机

器间运输时间为30 min(基准算例获取网址：GitHub-hnulq191817880/mygit2019：statistics for paper)。

表4 算例数据

Tab.4 Instance data

规模	数据集	工件×工序×工厂×机器	单工厂每道工序可用机器数量
小型算例	DFM01～DFM05	10×5×2×6	2
	DFM06～DFM10	15×5×2×6	
	DFM11～DFM15	20×5×2×6	
中型算例	DFM16～DFM20	10×10×4×8	3
	DFM21～DFM25	15×10×4×8	
	DFM26～DFM30	20×10×4×8	
大型算例	DFM31～DFM35	30×10×6×10	4
	DFM36～DFM40	15×15×6×10	

本文设计了两组实验来开展评估测试。第一组为消融实验，将本文所提出的IIG、GIMI算法与改进的遗传算法(improved genetic algorithm，IGA)[21]进行对比，以验证GIMI及IIG算法所设计的双梯度结构的有效性；第二组为对比实验，将IIG与高效模因算法(efficient memetic algorithm，EMA)[20]、混合分布估计与可变邻域搜索算法(estimation of distribution algorithm and variable neighborhood search，EDA-VNS)[8]、改进的延迟接受爬山算法(improved late acceptance hill climbing algorithm，ILAHC)[9]以及IG[11]四种算法进行对比，以进一步验证本文算法的有效性。选择上述对比算法的依据如下：①对比算法均针对考虑运输时间的分布式调度问题展开研究；②EMA、EDA-VNS、ILAHC等算法均源自该领域近几年的代表性研究成果；③通过与IG算法进行对比，可进一步体现IIG算法改进的必要性。

4.2 参数设置

在测试算法的性能前，首先需要确定算法的参数。IIG算法包含最大迭代次数I_{max}、邻域规模η、G1移除的工件个数ρ三个参数。I_{max}值越大，表明优化能力越好，但过大会导致耗时显著增加；η值越大，表明局部搜索能力越强，然而，对局部空间的过度探索也会使算法更易陷入局部最优；适当的ρ值可以保证较优的重构优化结果。参照文献[7]，采用正交试验确定最佳参数组合，随机选取实例DFM16用于参数设置，它涵盖4个工厂、8台机器和10个工件，在G1阶段，剩余可移除工件总数为$n-1=9$。表5列出了3个参数的合理设定值，其中$\lfloor\cdot\rfloor$表示向下取整。表6给出了参数组合的正交阵列，每个参数组合独立运行10次，取平均最大完工时间$C_{max,av}$作为比较指标。图9显示了上述3个参数的趋势图，由此得到如下最佳参数设置：$I_{max}=100$，$\eta=25$，$\rho=\lfloor(n-1)/2\rfloor$。

表5 参数水平设置

Tab.5 Parameter level setting

参数	水平			
	1	2	3	4
I_{max}	80	100	120	140
η	20	25	30	35
ρ	2	3	4	5)

表6 参数的正交阵列

Tab.6 Orthogonal array of parameters

组合	参数			$C_{max,av}$/min
	I_{max}	η	ρ	
1	80	20	2	757.3
2	80	25	3	752.8
3	80	30	4	749.4
4	80	35	5	741.3
5	100	20	3	745.1
6	100	25	2	735.7
7	100	30	5	745.9
8	100	35	4	737.6
9	120	20	4	736.5
10	120	25	5	741.9
11	120	30	2	748.5
12	120	35	3	748.9
13	140	20	5	756.8
14	140	25	4	742.3
15	140	30	3	746.8
16	140	35	2	751.2

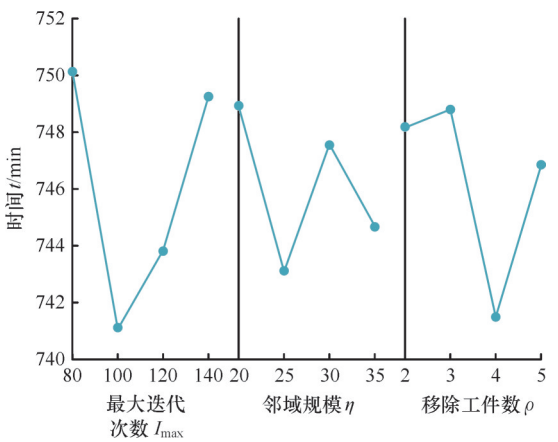


图9 参数趋势图

Fig.9 Parameter trend chart

4.3 消融实验

GIMI是一种将关键工件分配到空闲机器上加工以降低最大完成时间的插入策略，因此评估GIMI时仅将IIG算法中的初始化以及迭代寻优

过程保留，然后与GIMI融合构成一个完整算法，将该测试算法记为GIMI*。IIG算法相较于GIMI新设计了双梯度结构以及工件范围内的寻优更新策略，双梯度结构扩大了搜索解空间，同时降低了转运时间因素的影响，寻优更新策略旨在寻找更优解。本文通过消融实验来验证IIG和GIMI*二者对DFJSPT优化的有效性。

在不同规模的算例中各挑选一个算例进行测试，分别选取DFM04、DFM06、DFM11、DFM18、DFM21、DFM29、DFM32和DFM38这8个代表性算例。参数设置参照4.2节，最大迭代次数设为100，邻域规模设为25，IIG算法中G1所移除的工件数设为$\lfloor (n-1)/2 \rfloor$。对每个算例进行10次测试，记录迭代过程中的最优解C_{max}、运行10次所得最大完工时间的平均解$C_{max,av}$、达到最优解所需的平均运行时间T_{av}，上述3个参数分别表征测试算法的核心性能、稳定性以及收敛性能。具体测试结果如表7所示。

表7　算法对八个算例优化对比

Tab.7　Optimization comparison of the algorithm on eight instances

算例	规模	IIG			GIMI*			IGA	
		C_{max}/min	$C_{max,av}$/min	T_{av}/s	C_{max}/min	$C_{max,av}$/min	T_{av}/s	C_{max}/min	$C_{max,av}$/min
DFM04	10×5×2×6	451	456.2	0.91	465	478.4	1.92	494	509.2
DFM06	15×5×2×6	473	483.1	1.52	490	502.3	2.54	712	723.5
DFM11	20×5×2×6	572	584.0	5.26	599	612.4	4.12	860	884.2
DFM18	10×10×4×8	853	861.7	6.35	950	995.6	4.77	967	1018.7
DFM21	15×10×4×8	950	956.2	10.89	1083	1103.5	5.21	1354	1396.3
DFM29	20×10×4×8	890	907.9	9.45	1149	1177.5	7.51	1302	1341.4
DFM32	30×10×6×10	962	980.6	10.65	1260	1281.3	6.15	1527	1558.9
DFM38	15×15×6×10	1286	1313.6	14.69	1534	1575.7	12.51	1711	1758.3

由表7可知，在上述8个不同规模算例的求解优化中，GIMI*对比IGA在C_{max}上具有明显优势，在$C_{max,av}$上GIMI*的稳定性优于IGA，且根据$C_{max,av}$值可以看出GIMI*在收敛性能上表现出色。同时，IIG在C_{max}以及$C_{max,av}$的测试中均优于GIMI*优化的结果，表明增加双梯度结构以及寻优更新策略后，算法在性能上得到了进一步改善；在T_{av}方面，由于IIG对比GIMI*新设计了双梯度结构，使得算法更完善，因此增加的耗时是可以接受的。

此外，分别将IIG和GIMI*两种算法针对上述算例运行20次，根据得到的结果绘制出箱线图，如图10所示，每组箱线图对应一个算例，由结果可知，针对DFM04和DFM06等小规模算例，两种算法的优化效果近似，但IIG仍优于GIMI*；而针对DFM32和DFM38等大规模算例，IIG优化效果显著。整体来看，与GIMI*相比，IIG对DFJSPT的求解性能更优，从而验证了IIG双梯度结构的有效性。

4.4　IIG与其他算法的对比

本文将IIG分别与GIMI*和另外4种算法作进一步对比。测试采用DFM中40个不同规模的算例，其中W表示算法对所有算例优化的相对偏差，$W=(C_{max}-C^*)/C^*$，C^*为算法所求解的最优值。

表8中将IIG与GIMI*的C_{max}值、$C_{max,av}$值以及W值优化指标进行了对比，可以看出，IIG在整体优化性能上优于GIMI*。

表9中将IIG与4种算法进行了对比，其中W_{av}为算法的平均相对偏差；K_{opt}为取得的最优解数量。可以看出，在K_{opt}值上，IIG、EMA、EDA-VNS、ILAHC、IG算法获得的最优解个数分别为40、1、1、3、0。其中，对比EMA，IIG在40个算例优化中获得39个最优解，且近30个算例的优化效果得到显著增强；对比EDA-VNS，IIG多获得39个优解；对比ILAHC，IIG多获得37个优解；对比IG，所有算例均取得最优。对比W以及W_{av}值可知，IIG相比于其他算法具有更优的稳定性。

图11给出了IIG对DFM15优化后的调度甘特图，其最大完工时间降低至571 min，对比EMA（1377 min）、EDA-VNS（864 min）、ILAHC（754 min）以及IG（889 min）等算法效果显著。由此可得，与同类型的优化算法相比，IIG的效果更优，能够实现更高质量的解，可为求解DFJSPT提供更优的解决方案。

接下来，开展不同运输时间参数的对比实验，以进一步验证本文算法贴合实际场景时的有效性。为确保可靠性，测试选取4.3节中8个不同规模的算例，并保持厂内跨机器运输时间、对比算法以及相应参数与前文一致，仅梯度调整跨工厂运输时间。本研究通过构建三组场景来覆盖多样化的跨厂运输耗时需求。：

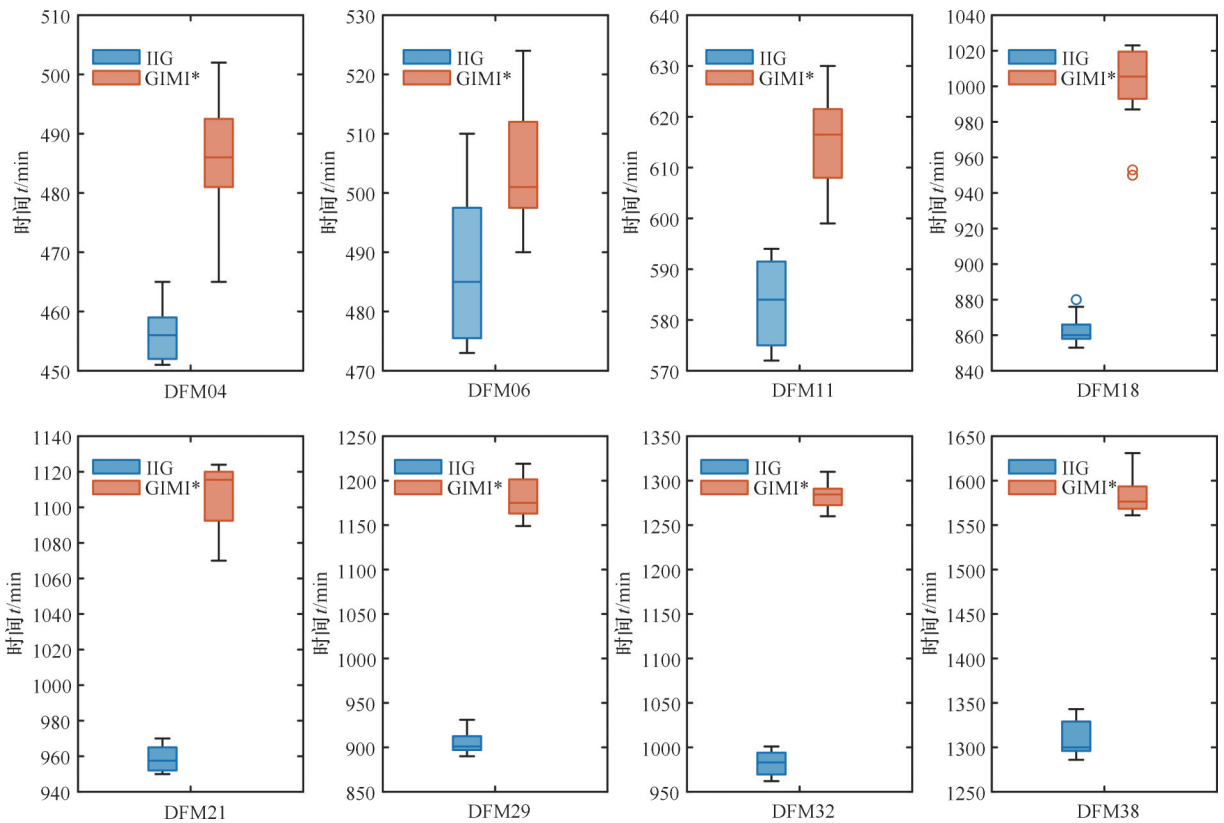


图10 IIG和GIMI*箱线图对比

Fig.10 Boxplot comparison between IIG and GIMI*

表8 IIG与GIMI*对比

Tab.8 Comparison between IIG and GIMI*

算例	IIG			GIMI*			算例	IIG			GIMI*		
	C_{max}/min	$C_{max,av}$/min	W	C_{max}/min	$C_{max,av}$/min	W		C_{max}/min	$C_{max,av}$/min	W	C_{max}/min	$C_{max,av}$/min	W
DFM01	**470**	473.6	0	478	483.1	0.017	DFM22	**858**	865.3	0	885	902.8	0.031
DFM02	**477**	482.2	0	485	494.3	0.017	DFM23	**903**	916.3	0	939	974.8	0.040
DFM03	**412**	415.9	0	434	442.7	0.053	DFM24	**858**	873.9	0	891	934.6	0.038
DFM05	**394**	399.6	0	415	426.5	0.053	DFM25	**850**	860.9	0	884	912.6	0.040
DFM07	**465**	470.5	0	501	506.2	0.077	DFM26	**910**	929.0	0	958	1010.9	0.053
DFM08	**463**	469.6	0	496	507.9	0.071	DFM27	**951**	966.0	0	997	1037.9	0.048
DFM09	**504**	515.5	0	536	547.1	0.063	DFM28	**940**	957.7	0	1012	1061.0	0.077
DFM10	**508**	514.1	0	532	550.8	0.047	DFM30	**974**	985.4	0	1175	1205.1	0.210
DFM12	**522**	528.7	0	547	559.3	0.048	DFM31	**976**	983.5	0	1163	1181.4	0.190
DFM13	**560**	565.5	0	601	616.2	0.073	DFM33	**920**	931.9	0	1189	1220.6	0.290
DFM14	**585**	591.3	0	613	627.8	0.048	DFM34	**892**	908.7	0	1154	1200.4	0.290
DFM15	**571**	581.6	0	607	634.7	0.063	DFM35	**909**	924.4	0	1176	1218.1	0.290
DFM16	**786**	801.6	0	880	901.7	0.12	DFM36	**1326**	1346.7	0	1527	1585.0	0.150
DFM17	**734**	753.4	0	749	803.1	0.02	DFM37	**1354**	1364.2	0	1601	1627.5	0.180
DFM19	**789**	803.6	0	878	908.7	0.11	DFM39	**1295**	1317.1	0	1568	1630.2	0.210
DFM20	**862**	876.6	0	902	931.7	0.046	DFM40	**1317**	1339.2	0	1621	1683.5	0.230

注：粗体表示IIG求得最优解。

1)场景S_1：跨工厂转运时间90 min，跨机器运输时间30 min，运输时间比3∶1；

2)场景S_2：跨工厂转运时间120 min，跨机器运输时间30 min，运输时间比4∶1；

3)场景S_3：跨工厂转运时间150 min，跨机器运输时间30 min，运输时间比5∶1；

表9　IIG与其他算法优化对比

Tab.9　Optimization comparison between IIG and other algorithms

算例	C^*/min	IIG			EMA		EDA-VNS		ILAHC		IG	
		C_{max}/min	$C_{max,av}$/min	W	C_{max}/min	W	C_{max}/min	W	C_{max}/min	W	C_{max}/min	W
DFM01	470	**470**	473.6	0	470	0	492	0.047	481	0.023	493	0.049
DFM02	477	**477**	482.2	0	545	0.14	513	0.075	489	0.025	537	0.130
DFM03	412	**412**	415.9	0	511	0.24	492	0.190	481	0.170	508	0.230
DFM04	451	**451**	456.2	0	493	0.093	489	0.084	473	0.049	519	0.150
DFM05	394	**394**	399.6	0	416	0.056	462	0.170	484	0.230	471	0.195
DFM06	473	**473**	483.1	0	724	0.53	701	0.480	684	0.450	632	0.340
DFM07	465	**465**	470.5	0	771	0.66	733	0.580	695	0.490	613	0.320
DFM08	463	**463**	469.6	0	767	0.66	728	0.570	683	0.480	642	0.390
DFM09	504	**504**	515.5	0	794	0.58	751	0.490	702	0.390	716	0.420
DFM10	508	**508**	514.1	0	926	0.82	779	0.530	691	0.360	711	0.400
DFM11	572	**572**	584.0	0	1137	0.99	871	0.520	798	0.400	917	0.600
DFM12	522	**522**	528.7	0	1106	1.12	859	0.650	770	0.480	861	0.650
DFM13	560	**560**	565.5	0	1288	1.3	856	0.530	789	0.410	908	0.620
DFM14	585	**585**	591.3	0	1033	0.77	795	0.360	712	0.220	912	0.560
DFM15	571	**571**	581.6	0	1377	1.41	864	0.510	754	0.320	889	0.560
DFM16	786	**786**	801.6	0	1054	0.24	937	0.100	851	0.080	1102	0.290
DFM17	734	**734**	753.4	0	1049	0.43	807	0.099	734	0	964	0.310
DFM18	853	**853**	861.7	0	967	0.31	853	0	853	0	1034	0.400
DFM19	789	**789**	803.6	0	1001	0.26	890	0.130	796	0.009	989	0.240
DFM20	862	**862**	876.6	0	930	0.079	906	0.051	862	0	1052	0.220
DFM21	950	**950**	956.2	0	2128	1.24	1317	0.390	1102	0.160	1321	0.390
DFM22	858	**858**	865.3	0	1768	1.06	1242	0.450	1023	0.190	1159	0.350
DFM23	903	**903**	916.3	0	2318	1.57	1356	0.500	1127	0.250	1298	0.440
DFM24	858	**858**	873.9	0	2061	1.40	1232	0.440	1089	0.270	1201	0.400
DFM25	850	**850**	860.9	0	2020	1.38	1198	0.410	1054	0.240	1198	0.410
DFM26	910	**910**	929.0	0	2062	1.27	1253	0.380	1101	0.210	1311	0.440
DFM27	951	**951**	966.0	0	2596	1.73	1521	0.599	1399	0.470	1476	0.550
DFM28	940	**940**	957.7	0	2281	1.43	1374	0.460	1147	0.220	1412	0.500
DFM29	890	**890**	907.9	0	3186	2.58	1289	0.450	1123	0.260	1342	0.510
DFM30	974	**974**	985.4	0	3043	2.12	1561	0.600	1412	0.450	2013	1.070
DFM31	976	**976**	983.5	0	3257	2.34	1711	0.750	1459	0.490	1981	1.030
DFM32	962	**962**	980.6	0	3781	2.93	1612	0.680	1438	0.490	1865	0.940
DFM33	920	**920**	931.9	0	3583	2.89	1532	0.670	1367	0.490	1722	0.870
DFM34	892	**892**	908.7	0	3857	3.32	1457	0.630	1297	0.450	1654	0.850
DFM35	909	**909**	924.4	0	4041	3.45	1635	0.799	1323	0.460	1692	0.860
DFM36	1326	**1326**	1346.7	0	2412	0.819	1952	0.470	1771	0.340	2315	0.750
DFM37	1354	**1354**	1364.2	0	2706	0.999	1734	0.280	1505	0.110	2389	0.760
DFM38	1286	**1286**	1313.6	0	3176	1.47	1781	0.380	1598	0.240	2256	0.750
DFM39	1295	**1295**	1317.1	0	3698	1.86	1739	0.340	1571	0.210	2309	0.780
DFM40	1317	**1317**	1339.2	0	2392	0.82	1818	0.380	1694	0.290	2346	0.780
W_{av}		0			1.18		0.41		0.27		0.51	
K_{opt}		40			1		1		3		0	

注：粗体表示IIG获得的最优解。

针对每组场景的每个算例，算法独立运行10次，记录最优值C_{max}以及相对偏差W，测试结果如表10所示。在三组不同场景下，通过对比算法的C_{max}以及W值可知，本文算法呈现出更优的性能以及稳定性，在实际生产中能更加有效地应对跨厂运输耗时带来的调度压力。

5　结论

本文针对工件可跨工厂运输的分布式柔性作业车间调度问题(DFJSPT)提出了一种改进的迭代贪婪(IIG)算法，经过测试验证了所提算法的有效性，得到如下结论：

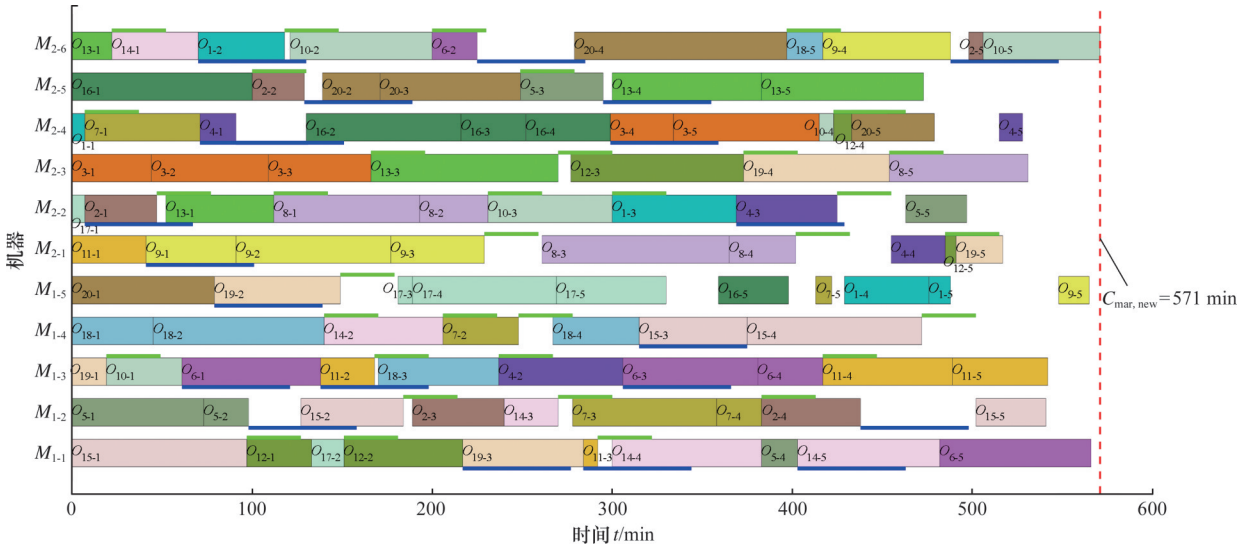


图11 DFM15优化甘特图

Fig.11 The optimization Gantt chart of DFM15

表10 不同场景下算法对比

Tab.10 Comparison of algorithms in different scenarios

场景	算例	规模	C^*/min	IIG		EMA		EDA-VNS		ILAHC	
				C_{max}/min	W	C_{max}/min	W	C_{max}/min	W	C_{max}/min	W
S_1	DFM04	10×5×2×6	476	476	0	517	0.086	503	0.057	484	0.017
	DFM06	15×5×2×6	487	487	0	741	0.522	710	0.458	685	0.407
	DFM11	20×5×2×6	607	607	0	1208	0.990	892	0.470	813	0.339
	DFM18	10×10×4×8	878	878	0	983	0.120	881	0.003	879	0.001
	DFM21	15×10×4×8	961	961	0	2231	1.322	1506	0.567	1287	0.339
	DFM29	20×10×4×8	929	929	0	3207	2.452	1296	0.395	1154	0.242
	DFM32	30×10×6×10	977	977	0	3798	2.887	1721	0.762	1466	0.501
	DFM38	15×15×6×10	1328	1328	0	3312	1.494	1923	0.448	1687	0.270
S_2	DFM04	10×5×2×6	476	476	0	523	0.099	506	0.063	491	0.032
	DFM06	15×5×2×6	509	509	0	750	0.473	744	0.462	703	0.381
	DFM11	20×5×2×6	631	631	0	1294	1.051	927	0.469	830	0.315
	DFM18	10×10×4×8	886	886	0	978	0.104	902	0.018	915	0.033
	DFM21	15×10×4×8	983	983	0	2216	1.254	1532	0.558	1304	0.327
	DFM29	20×10×4×8	948	948	0	3327	2.509	1387	0.463	1216	0.283
	DFM32	30×10×6×10	992	992	0	3861	2.892	1903	0.918	1534	0.546
	DFM38	15×15×6×10	1380	1380	0	3396	1.461	1962	0.422	1658	0.201
S_3	DFM04	10×5×2×6	476	476	0	522	0.097	519	0.090	508	0.067
	DFM06	15×5×2×6	520	520	0	791	0.521	762	0.465	739	0.421
	DFM11	20×5×2×6	654	654	0	1289	0.971	971	0.485	864	0.321
	DFM18	10×10×4×8	894	894	0	1016	0.136	959	0.073	921	0.030
	DFM21	15×10×4×8	976	976	0	2278	1.334	1614	0.654	1381	0.415
	DFM29	20×10×4×8	934	934	0	3297	2.530	1402	0.501	1276	0.366
	DFM32	30×10×6×10	1004	1004	0	3889	2.874	2017	1.009	1589	0.583
	DFM38	15×15×6×10	1427	1427	0	3428	1.402	2069	0.450	1721	0.206

1)设计了一种基于机器空闲的贪婪插入策略(GIMI),基于初始方案将关键路径上的工件分配至其余空闲机器上以平衡最大完工时间,设计三属性评估指标和评估函数以筛选全局最优插入位置。

2)基于GIMI设计G1、G2双梯度邻域结构,在扩大解空间的同时降低了转运时间对完工时间的影响,提高了搜索效率。

3)采用扩展的FJSP基准算例与4种优秀的算法进行对比实验,结果表明,IIG对比其余算法表现更加优异,在DFJSPT求解中获得了所有算例的最优解。

未来的工作将从以下方面进行考虑:①算法

扩展性：在下一步研究中可以将IIG应用于紧急插单、订单延期以及考虑多目标调度优化等场景中；②实际生产需求：后续研究工作还可以围绕能耗优化、设备故障、运输路径优化等实际性问题开展，以提升算法的工程应用价值。

参考文献：

[1] 王凌，邓瑾，王圣尧. 分布式车间调度优化算法研究综述[J]. 控制与决策，2016，31(1)：1-11.
WANG Ling，DENG Jin，WANG Shengyao. Survey on Optimization Algorithms for Distributed Job Shop Scheduling[J]. Control and Decision，2016，31(1)：1-11.

[2] 祝正宇，郭具涛，吕佑龙，等. 面向柔性作业车间生产调度的深度强化学习方法[J]. 中国机械工程，2024，35(11)：2007-2014.
ZHU Zhengyu，GUO Jutao，LYU Youlong，et al. Deep Reinforcement Learning Method for Flexible Job Shop Scheduling[J]. China Mechanical Engineering，2024，35(11)：2007-2014.

[3] XIE J，LI X，GAO L，et al. A Hybrid Genetic Tabu Search Algorithm for Distributed Flexible Job Shop Scheduling Problems[J]. Journal of Manufacturing Systems，2023，71：82-94.

[4] MENG L，ZHANG C，REN Y，et al. Mixed-integer Linear Programming and Constraint Programming Formulations for Solving Distributed Flexible Job Shop Scheduling Problem[J]. Computers & Industrial Engineering，2020，142：106347.

[5] LI J，DUAN P，CAO J，et al. A Hybrid Pareto-based Tabu Search for the Distributed Flexible Job Shop Scheduling Problem with E/T Criteria[J]. IEEE Access，2018，6：58883-58897.

[6] 魏光艳，叶春明. 分布式多柔性装配作业车间调度问题研究[J]. 中国机械工程，2023，34(20)：2442-2455.
WEI Guangyan，YE Chunming. Research on Distributed and Multi-flexible Assembly Job Shop Scheduling Problems[J]. China Mechanical Engineering，2023，34(20)：2442-2455.

[7] 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.

[8] 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.

[9] 余佳林，姚锡凡，单文俊，等. 考虑工件运输时间的分布式柔性作业车间调度[J]. 组合机床与自动化加工技术，2024(6)：185-192.
YU Jialin，YAO Xifan，SHAN Wenjun，et al. The Distributed Flexible Job Shop Scheduling Considering the Transportation Time of Workpieces[J]. Modular Machine Tool & Automatic Manufacturing Technique，2024(6)：185-192.

[10] STRONG D，KAY M，CONNER B，et al. Hybrid Manufacturing-integrating Traditional Manufacturers with Additive Manufacturing(AM) Supply Chain[J]. Additive Manufacturing，2018，21：159-173.

[11] AQEL A G，LI X，GAO L. A Modified Iterated Greedy Algorithm for Flexible Job Shop Scheduling Problem[J]. Chinese Journal of Mechanical Engineering，2019，32(1)：21.

[12] PAN Q，RUIZ R. An Effective Iterated Greedy Algorithm for the Mixed No-idle Permutation Flowshop Scheduling Problem[J]. Omega，2014，44：41-50.

[13] SHAO W，PI D，SHAO Z. Optimization of Makespan for the Distributed No-wait Flow Shop Scheduling Problem with Iterated Greedy Algorithms[J]. Knowledge-Based Systems，2017，137：163-181.

[14] HUANG L，TANG D，ZHANG Z，et al. An Iterated Greedy Algorithm Integrating Job Insertion Strategy for Distributed Job Shop Scheduling Problems[J]. Journal of Manufacturing Systems. 2024，77：746-763.

[15] YU Y，ZHONG Q，SUN L，et al. A Self-adaptive Two Stage Iterative Greedy Algorithm Based Job Scales for Energy-efficient Distributed Permutation Flowshop Scheduling Problem[J]. Swarm and Evolutionary Computation，2025，92：101777.

[16] ZHAO F，DU Y，ZHUANG C，et al. An Iterative Greedy Algorithm for Solving a Multiobjective Distributed Assembly Flexible Job Shop Scheduling Problem with Fuzzy Processing Time[J]. IEEE Transactions on Cybernetics，2025，55(5)：2302-2315.

[17] DENG G，SU Q，ZHANG Z，et al. A Population-based Iterated Greedy Algorithm for No-wait Job Shop Scheduling with Total Flow Time Criterion[J]. Engineering Applications of Artificial Intelligence，2020，88：103369.

[18] LI H，LI X，GAO L. An Iterated Greedy Algorithm with Acceleration of Job Allocation Probability for Distributed Heterogeneous Permutation Flowshop Scheduling Problem[J]. Swarm and Evolutionary Computation，2024，88：101580.

[19] 姜一啸，吉卫喜，何鑫，等. 基于改进非支配排序遗传算法的多目标柔性作业车间低碳调度[J]. 中国机械工程，2022，33(21)：2564-2577.
JIANG Yixiao，JI Weixi，HE Xin，et al. Low-carbon Scheduling of Multi-objective Flexible Job-shop Based on Improved NSGA-Ⅱ[J]. China Mechanical Engineering，2022，33(21)：2564-2577.

[20] 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.

[21] GIOVANNI D L，PEZZELLA F. An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem[J]. European Journal of Operational Research，2009，200(2)：395-408.

（编辑　胡佳慧）

作者简介：唐敦兵，男，1972年生，教授、博士研究生导师。研究方向为制造系统与自动化，智能制造系统、数字化设计与制造。E-mail：d. tang@nuaa. edu. cn。**黄　林***（通信作者），男，1995年生，博士研究生。研究方向为智能制造系统、生产调度。E-mail：hl618850@nuaa. edu. cn。

本文引用格式：

唐敦兵，崔晓哲，黄林，等. 一种改进迭代贪婪算法求解跨工厂运输的分布式柔性作业车间调度问题[J]. 中国机械工程，2026，37(11)：2851-2863.

TANG Dunbing，CUI Xiaozhe，HUANG Lin，et al. An Improved Iterative Greedy Algorithm for Solving Distributed Flexible Job-shop Scheduling Problem with Cross-factory Transportation[J]. China Mechanical Engineering，2026，37(11)：2851-2863.