控制与决策
Control and Decision

具有顺序柔性的车间调度问题的变邻域禁忌搜索算法

宁国宇, 陶汉桥, 宋国鹏, 李明浩, 杨克巍

引用本文:
宁国宇, 陶汉桥, 宋国鹏, 等. 具有顺序柔性的车间调度问题的变邻域禁忌搜索算法[J]. 控制与决策, 2026, 41(1): 31-43.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2025.0509

具有顺序柔性的车间调度问题的变邻域禁忌搜索算法

宁国宇，陶汉桥，宋国鹏，李明浩†，杨克巍

(国防科技大学 系统工程学院，长沙 410073)

摘　要: 随着工业智能化技术的快速发展, 车间制造模式正逐步向柔性化方向转型. 在柔性制造模式下, 工件的加工过程突破了固定设备和刚性工艺路线的约束, 展现出多维度的柔性特征. 然而, 现有研究主要集中于设备资源的柔性, 对加工顺序柔性的建模与优化却存在显著的不足. 为了解决这一问题, 首先设计一种能够考虑顺序柔性特征的新型邻域结构, 可以有效地调整工件中工序加工的顺序. 在此基础上, 进一步提出一种变邻域禁忌搜索算法, 该算法通过变邻域搜索与禁忌策略的协同优化, 能够高效求解具有顺序柔性的车间调度问题, 获得高质量调度方案. 实验结果表明, 所提出算法的求解能力与通用数学规划求解器相比具有明显优势, 为相关车间调度问题提供了科学的方法支撑.

关键词: 车间调度；柔性制造；顺序柔性；新型邻域结构；变邻域禁忌搜索

中图分类号: TP301　　**文献标志码:** A

DOI: 10.13195/j.kzyjc.2025.0509

引用格式: 宁国宇, 陶汉桥, 宋国鹏, 等. 具有顺序柔性的车间调度问题的变邻域禁忌搜索算法 [J]. 控制与决策, 2026, 41(1): 31-43.

Variable neighborhood tabu search algorithm for job shop scheduling problem with sequencing flexibility

NING Guo-yu，*TAO Han-qiao*，*SONG Guo-peng*，*LI Ming-hao*†，*YANG Ke-wei*

(College of Systems Engineering，National University of Defense Technology，Changsha 410073，China)

Abstract: With the rapid development of industrial intelligence technology, the workshop manufacturing mode is gradually transitioning towards flexibility. In the flexible manufacturing mode, the machining process of job breaks through the constraints of fixed equipment and rigid process routes, exhibiting multidimensional flexible characteristics. However, existing researches mainly focus on the flexibility of equipment resources, and there are significant shortcomings in modeling and optimizing the flexibility of processing sequences. To address this problem, this paper first proposes a novel neighborhood structure that takes into account sequential flexible features, which can effectively adjust the processing order of operation in job. On this basis, this paper further proposes a variable neighborhood tabu search algorithm, which can efficiently solve job scheduling problems with sequencing flexibility and obtain high-quality scheduling solutions through the collaborative optimization of variable neighborhood search and tabu strategy. The experimental results show that the solving ability of the proposed algorithm exhibits significant advantages compared to general mathematical programming solvers, providing scientific methodological support for related job shop scheduling problems.

Keywords: job shop scheduling； flexible manufacturing mode； sequencing flexibility； new neighborhood structure；variable neighborhood tabu search

0　引　言

工业 4.0 与智能制造的深化加速了柔性制造模式驱动的制造业转型[1]. 基于工业物联网与智能感知技术, 传统固定设备与刚性工艺的制造范式正向多维度柔性协同模式演进, 通过动态配置设备、工艺路径及加工顺序, 实现定制化需求与动态扰动的快速响应[2-4]. 工序加工顺序的柔性使得能够对工序执行序列进行优化调整, 从而显著提升资源利用率以及制造系统的敏捷性.

在传统作业车间调度问题 (job shop scheduling problem, JSP) 中, 一个工件 (job) 包含多个工序 (operation), 工件中各个工序的执行顺序是固定的,

收稿日期: 2025-05-15；**录用日期:** 2025-10-17.
基金项目: 国家自然科学基金重点项目 (72231011)；国家自然科学基金重大项目 (92467302).
责任编委: 唐万生.
†通信作者. E-mail: liminghao4869@nudt.edu.cn.

每个工序只能被单个机器进行加工且加工时间固定[5-6]. 而柔性作业车间调度问题 (flexible job shop scheduling problem, FJSP) 相较于 JSP 则增加了机器选择上的柔性, 即一个工序可以在多个不同的机器上完成, 且工序在不同机器上处理的时间不同[7-8]. 本文研究的具有加工顺序柔性的柔性车间调度问题 (flexible job shop scheduling problem with sequencing flexibility, FJSP-SF) 在经典 FJSP 基础上, 进一步考虑了工序的顺序柔性约束. 区别于经典 FJSP 中工序间严格的全序关系约束, 本问题的核心特征在于: 允许同一工件的各工序加工顺序遵循有向无环图 (directed acyclic graph, DAG) 的拓扑排序规则, 即工序间仅需满足部分偏序约束而非严格的全序约束. 同时, 与 FJSP 一样, FJSP-SF 中同一工件内的各工序间不能并行加工, 且不同工件的工序间不存在工艺约束关系. 因此, FJSP-SF 可看作 FJSP 的拓展[9]. 此外, 由于 FJSP-SF 仅要求工件工序约束可表示为 DAG, 其亦可被视为开放车间调度 (open shop scheduling problem, OSP)、分组车间调度 (group shop scheduling problem, GSP) 等问题的推广形式[10]. 值得注意的是, 尽管装配作业车间调度 (assembly job shop scheduling problem, AJSP[11]) 与 FJSP-SF 均采用有向无环图 (DAG) 表征工序约束, 但前者描述工件间的装配关系, 而后者用于定义同一工件内部工序间的约束关系.

FJSP-SF 在半导体制造、航空发动机制造等领域相较于 JSP 或者 FJSP 更贴近实际生产场景[12-15]. 然而, 目前对 FJSP-SF 中所存在的顺序柔性的研究工作十分有限[16-17]. 例如, Yuan 等[13] 对部分工序具有任意顺序自由度的 FJSP-SF 进行研究, 通过优化变量定义和约束条件, 提出了两种新型混合整数规划模型, 在小规模问题上相较于 Gong 等[15] 提出的混合整数规划模型明显提升了求解速度和解的质量, 同时, 针对大规模算例提出了一种结合局部搜索策略的混合进化算法, 在该类问题的 110 个标准算例上取得了最佳的求解效果. Birgin 等[18-19] 声称构建了 FJSP-SF 的整数规划模型并提出了相应的求解方法. 然而, Dauzère-Pérès 等[17] 指出, Birgin 等研究 FJSP-SF 时认为单个工件的不同工序之间是可以并行加工的, 忽略了单个工件在每个时刻只能加工一个工序的要求, 导致其研究结果不能直接应用于求解真正的 FJSP-SF.

尽管 FJSP-SF 在工业制造领域有着十分重要的研究意义, 但相较于 JSP 和 FJSP, FJSP-SF 仍然缺乏高效的启发式算法, 在求解大规模的 FJSP-SF 时难以快速获得问题的一个高质量可行解. 在现有的众多求解车间调度相关问题的算法中[20-22], 基于局部搜索的算法对于求解 JSP[5] 及 FJSP[23-25] 往往有很好的求解效果. 而基于局部搜索的算法其效果好坏取决于邻域结构的设计是否合理, 尽管当前对于 JSP 或者 FJSP 都存在较好的邻域结构[26-28], 但这些邻域结构都是以工序加工顺序是严格的全序关系为前提的, 无法灵活地改变工件中工序的加工顺序, 因此并不能对 FJSP-SF 的解空间进行完整地搜索. 针对这一问题, 本文建立 FJSP-SF 的数学模型, 提出一类新型邻域结构, 以工件中工序加工顺序的改变作为邻域动作, 从而能够灵活地处理 FJSP-SF 所存在的顺序柔性问题. 基于所提出的新型邻域结构和 JSP 以及 FJSP 中经典的邻域结构, 本文设计一种结合禁忌策略的变邻域搜索算法 (variable neighborhood tabu search algorithm, VNTSA) 以求解 FJSP-SF. 实验结果表明, 所提出变邻域禁忌搜索算法相较于通用求解器 CPLEX 在求解大规模的 FJSP-SF 时具有明显优势.

1 问题描述与数学模型

为了对 FJSP-SF 的数学模型以及后续相关概念进行说明, 首先定义符号如表 1 所示.

表1 符号定义

符号	符号解释
$\mathbb{J}$	集合常量, 代表所有工件的集合
$\mathcal{O}$	集合常量, 代表所有工序的集合
$\mathcal{O}_{\mathcal{J}}$	代表工件$\mathcal{J}$所包含的所有工序
$\mathcal{F}$	集合常量, 代表所有可用机器的集合
$\mathcal{F}_i$	集合常量, 代表可用于加工工序i的所有机器的集合
t_{ik}	正数常量, 代表工序i在机器k上加工所需的时长
s_i	正数决策变量, 代表工序i最早开始时间
l_i	正数变量, 代表工序i最迟开始时间
a_{ij}	0-1常量, 为1时代表在工序约束的有向无环图中工序i为工序j的拓扑前序
x_{ik}	0-1决策变量, 为1时代表工序i被分配到机器k上进行加工
y_{ij}	0-1决策变量, 为1时代表工序i与工序j在同一机器上加工, 且工序i比j先加工
z_{ij}	0-1决策变量, 为1时代表工序i与工序j为同一工件的工序, 且工序i比j先加工
$C_{\max}$	正数决策变量, 代表完成所有加工任务的最大完工时间
$\mathbb{T}_i$	代表工序i在当前机器上所需的加工时长
$\mathrm{MS}(i)$	代表工序i在当前机器上后一个加工的工序
$\mathrm{MP}(i)$	代表工序i在当前机器上前一个加工的工序
$\mathrm{JS}(i)$	代表工序i在所属工件上后一个加工的工序
$\mathrm{JP}(i)$	代表工序i在所属工件上前一个加工的工序

1.1　问题描述

FJSP-SF 以最小化最大完工时间为优化目标, 具有下列 4 个前提条件: 所有工件在初始时刻可加工; 所有机器在初始时刻可用; 工序一旦开始加工便不会被中断; 每个工序可以在多个不同机器上加工, 且在不同机器上加工的时间不完全相同. 同时满足以下约束:

1) 每个工序仅需被加工一次;

2) 每个机器在同一时刻只能加工一个工序;

3) 每个工件在同一时刻只能被一个机器加工, 即同一工件的不同工序之间不能并发执行, 这是 FJSP-SF 与 Birgin 等[18-19] 所解决问题的核心差异;

4) 每个工件中工序之间的加工顺序遵守一个有向无环图的拓扑排序关系, 不需遵守严格的全序关系, 这是 FJSP-SF 与 FJSP 的关键区别.

本文使用析取/合取图 (disjunctive/conjunctive graph)[5,16] 描述 FJSP-SF. 在 JSP 和 FJSP 中, 工件的工序顺序固定, 因此在对应的析取图中, 同一工件的工序之间由有向弧连接. 而 FJSP-SF 中工序顺序是柔性的, 同一工件的工序间可以由无向弧连接. 调度方案确定后, 析取图中所有无向弧被定向, 此时将其称为合取图并表示为$G=(N,A,E)$, 能唯一代表该调度方案. 其中: $N=\mathcal{O}\bigcup\{0,\#\}$为节点集合, 一个节点对应一个工序, "0" 节点和 "#" 节点分别为开工和完工时刻的虚拟节点; A为工件中工序加工顺序的有向弧集合; E为机器上工序加工顺序的有向弧集合. 为方便后续说明, 本文以 "作业弧" 代指集合A中的弧, 以 "机器弧" 代指E中的弧, 分别用实线和虚线表示. 考虑一个包含 3 个工件 (i ∼ iii, 各含 4 道工序) 和 5 台机器的 FJSP-SF 实例, 该实例中各个工序的可用机器及在可用机器上所需的加工时间如表 2 所示. 工件中工序的工艺路线约束如图 1 所示.

表2　示例中工序的可用机器及加工时间

工件编号	工序编号	可用机器及时间				
		I	II	III	IV	V
i	1	6	—	—	7	5
	2	—	2	5	—	—
	3	4	—	—	6	7
	4	—	8	—	5	—
ii	5	1	—	—	3	2
	6	—	6	9	—	—
	7	2	2	—	—	5
	8	—	—	5	8	—
iii	9	2	—	3	—	—
	10	6	5	—	—	—
	11	—	6	4	—	5
	12	3	—	3	—	1

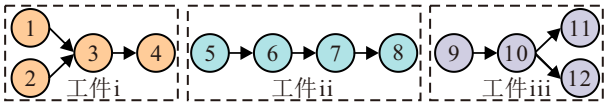


图1　示例中工序的工艺约束关系

工件 i 中工序 1 和工序 2 以及工件 iii 中工序 11 和工序 12 的加工顺序不固定, 表现出顺序柔性; 而工件 ii 中的工序严格遵循全序约束, 不具备顺序柔性. 该实例某一可行调度方案如图 2 所示. 为了能够全面考虑 FJSP-SF 中的顺序柔性与机器柔性, 在 FJSP 常用的基于机器视角的甘特图 (图 2(a)) 与基于工件视角的合取图 (图 2(b)) 的基础上, 构建了基于工件视角的甘特图 (图 2(c)) 和基于机器视角的合取图 (图 2(d)).

1.2　数学模型

本文结合表 1 定义的符号, 构建如下 FJSP-SF 的 0-1 整数规划模型, 作为后续算法设计的基础:

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

$$\text{s.t. } C_{\max}\geqslant s_i+\sum_{k\in\mathcal{F}_i}t_{ik}x_{ik},\ i\in\mathcal{O}; \tag{2}$$

$$\sum_{k\in\mathcal{F}_i}x_{ik}=1,\ i\in\mathcal{O},\ k\in\mathcal{F}_i; \tag{3}$$

$$s_i\geqslant s_j+t_{jk}-L(2-x_{ik}-x_{jk}+y_{ij}),\quad i,j\in\mathcal{O},\ i\neq j,\ k\in\mathcal{F}_i\bigcap\mathcal{F}_j; \tag{4}$$

$$s_j\geqslant s_i+t_{ik}-L(3-x_{ik}-x_{jk}-y_{ij}),\quad i,j\in\mathcal{O},\ i\neq j,\ k\in\mathcal{F}_i\bigcap\mathcal{F}_j; \tag{5}$$

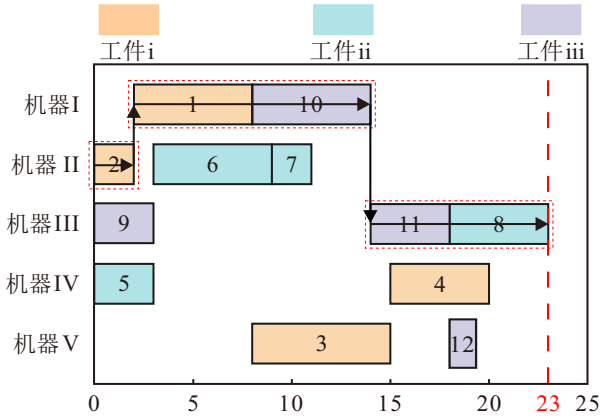
$$z_{ij}\geqslant a_{ij},\ i,j\in\mathcal{O}; \tag{6}$$

$$s_i\geqslant s_j+\sum_{k\in\mathcal{F}_j}t_{jk}x_{jk}-Lz_{ij},\quad i,j\in\mathcal{O}_{\mathcal{J}},\ i\neq j,\ \mathcal{J}\in\mathbb{J}; \tag{7}$$

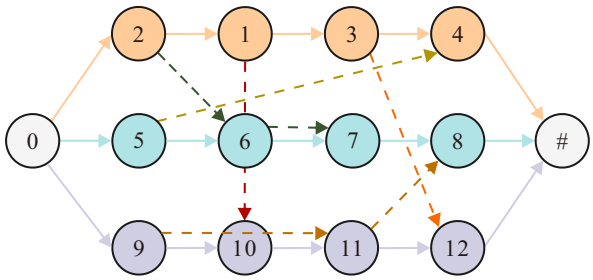
$$s_j\geqslant s_i+\sum_{k\in\mathcal{F}_i}t_{ik}x_{ik}-L(1-z_{ij}),\quad i,j\in\mathcal{O}_{\mathcal{J}},\ i\neq j,\ \mathcal{J}\in\mathbb{J}; \tag{8}$$

$$x_{ik},y_{ij},z_{ij}\in\{0,1\},\ i,j\in\mathcal{O},\ k\in\mathcal{F}_i. \tag{9}$$

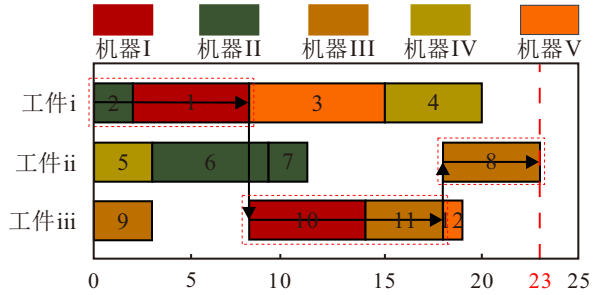
其中: L为一个足够大的正数; 式 (1) 为优化目标, 代表最小化工件完工的最大完工时间; 式 (2) 代表最大完工时间是每个工序完工时间的最大值; 式 (3) 代表每个工序有且仅在一个机器上进行加工; 式 (4) 和 (5) 保证了任意机器不会同时加工两个工序, 且式 (4) 和 (5) 分别蕴含同一机器上工序i和j的y_{ij}与y_{ji}不全部为 0 和不全部为 1 的限制; 式 (7) 和 (8) 保



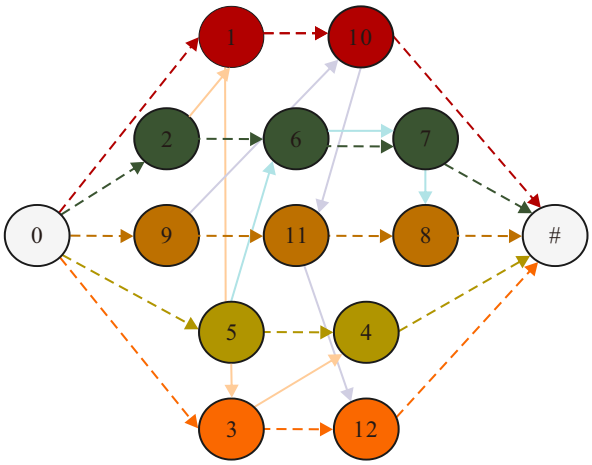
(a) 机器视角下的调度甘特图



(b) 工件视角下的合取图



(c) 工艺视角下的调度甘特图



(d) 机器视角下的合取图

图2 示例调度方案的甘特图和合取图模型

证了任意工件不会同时有两个被加工的工序, 并确保每个工件中工序之间的加工顺序满足给定的排序关系, 同时式 (7) 和 (8) 分别蕴含同一工件上工序i和j的z_{ij}与z_{ji}不全部为 0 和不全部为 1 的限制; 式 (9) 定义了决策变量x_{ik}、y_{ij}与z_{ij}的定义域.

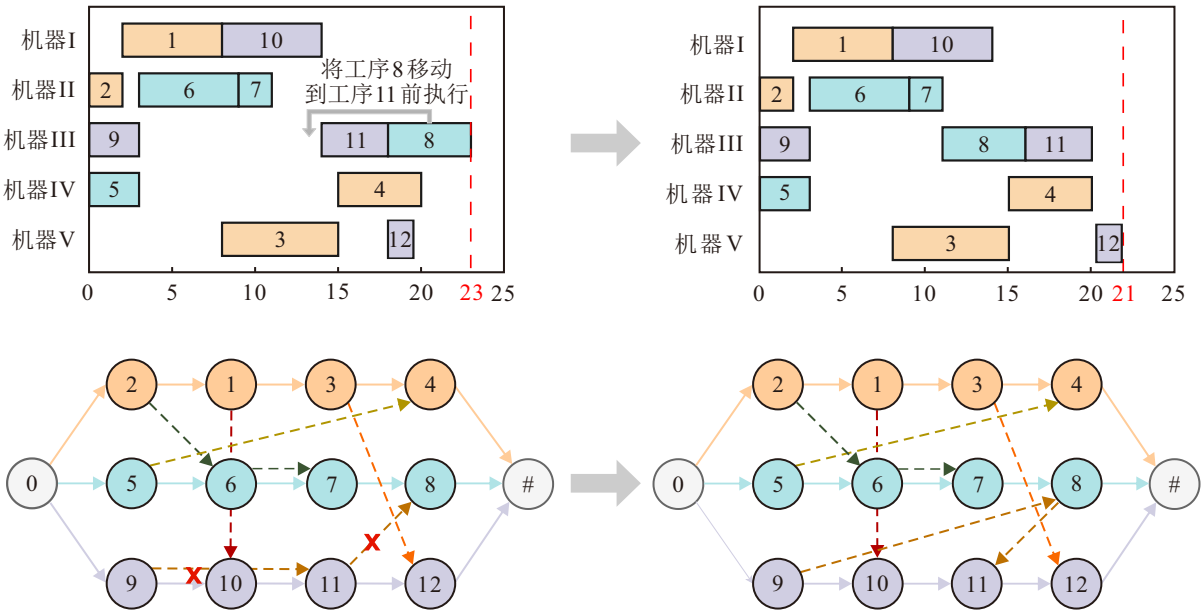
2 考虑顺序柔性的新型邻域结构

2.1 JSP 和 FJSP 中的关键工序块的定义以及邻域动作的设计

传统的 JSP 和 FJSP 每个工件中工序之间不存在顺序柔性, 即每个工件中的所有工序是顺序执行的, 仅能对工序在机器上的执行顺序进行改变. 因此, 以往研究中考虑的关键工序块都是在机器视角下定义的[15-16]: 关键工序块为关键路径中位于同一个机器上的连续工序的集合, 关键路径即合取图模型中从虚拟头节点 “0” 到虚拟尾节点 “#” 的最长路径[26] (不包括 “0” 和 “#”). 例如, 在图 2 中, 工序 2→工序 1→工序 10→工序 11→工序 8 为所示调度方案的关键路径, 该路径中存在 3 个关键工序块, 分别是 {工序 2}、{工序 1, 工序 10}, {工序 11, 工序 8}. 关键路径的计算可采用 “逆向追溯法” 实现: 从具有最大完工时间的节点出发, 沿合取图逆向回溯, 每一步回溯到完工时间更大的工件前序或机器前序节点, 直至回溯至 “0” 节点, 所经过的节点序列即为关键路径. 由于合取图具有 “有向无环” 的性质, 每个节点最多仅访问一次, 访问节点的最大次数不超过图中工序的总数$|\mathcal{O}|$.

对于 JSP 的邻域动作设计, 其邻域动作可概括为对处于关键工序块中的单个工序, 在机器上进行移动改变其加工顺序, 从而产生一个新的可行调度方案, 通过合取图可将 JSP 的邻域动作描述为在不改变合取图中作业弧 (如图 2(b) 中的实线) 的情况下, 重构合取图中与关键工序块相连的机器弧 (图 2(b) 中的虚线), 产生一个不包含回路的新合取图[26-29]. FJSP 的邻域结构也可被视为在保证合取图中不形成回路的前提下, 对合取图中的机器弧进行反转和重构. 与 JSP 的不同在于, FJSP 邻域结构中的机器弧反转重构可能涉及多个不同的机器[30-31].

尽管 JSP 与 FJSP 的邻域并不是针对 FJSP-SF 所设计的, 但其研究思路对 FJSP-SF 邻域结构的设计具有重要的参考意义. 图 3 给出了示例问题中一个仅改变机器弧的邻域动作. 图中, 机器 III 中的工序 8 移动到工序 11 前执行, 在发生移动后, 机器 III 所对应的机器弧的指向变为工序 9→工序 8→工序 11, 而作业弧不发生改变. 由于 JSP 与 FJSP 的邻域结构无法改变工序在工件中的执行顺序, 在合取图模型上不涉及对作业弧的重构, 这些邻域动作无法保证得到 FJSP-SF 的最优解.



与工序8和11相连的机器弧进行反转重构

图3　仅改变机器弧的邻域动作示例

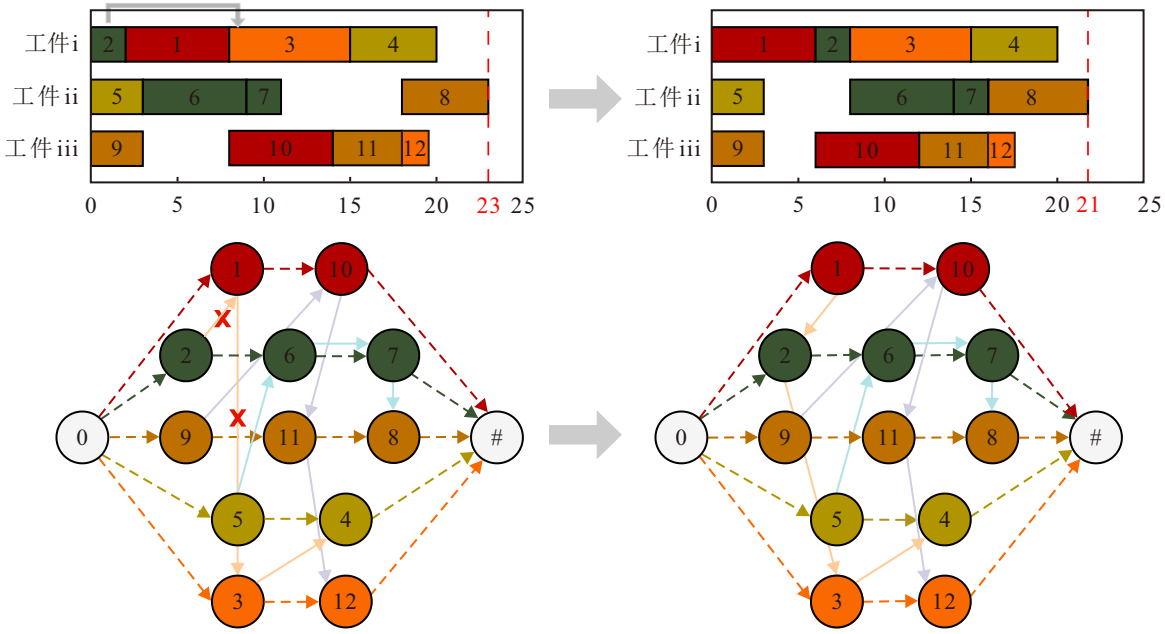
2.2　考虑顺序柔性的新型邻域结构

相较于 JSP 和 FJSP, FJSP-SF 中需要进一步考虑工序之间的顺序柔性. 因此, 调整关键路径中的作业弧也能得到 FJSP-SF 中可行调度方案的邻域解. 为了方便后续对本文所设计新型邻域结构的阐述, 首先给出作业关键工序块的定义.

作业关键工序块: 关键路径中属于同一个工件的连续工序的集合. 例如, 图 2(c) 中存在着{工序 2, 工序 1}, {工序 10, 工序 11}以及{工序 8}共 3 个作业关键工序块.

结合 JSP 与 FJSP 的邻域结构, 本文提出 FJSP-SF 的一种新型邻域结构. 即在保证工序在机器上执行顺序不变的情况下, 改变位于作业关键工序块中的工序在工件中的执行顺序.

在合取图中, 可对该类邻域结构表述如下: 在不改变合取图中机器弧 (图 2(d) 虚线) 的前提下, 仅改变与作业关键工序块相连的作业弧 (图 2(d) 实线) 产生一个不包含回路的新合取图. 图 4 给出了示例问题中一个仅改变作业弧的邻域动作. 其中, 工件i中的工序 2 被移动到工序 1 之后执行. 执行邻域动作后, 工件i所对应的作业弧指向变为工序 1→工序 2→工序 3→工序 4, 而机器弧不发生改变, 并且最大



与工序1和工序2相连的作业弧进行反转重构

图4　仅改变作业弧的邻域动作示例

完工时间由 23 减少到 21.

由图 2 ∼ 图 4 易知, 在合取图中如何仅改变机器弧而不产生回路与如何仅改变作业弧而不产生回路这两个问题在数学形式上是相似的. 然而, FJSP-SF 中作业弧的改变除了受到不产生回路的限制外, 还存在着两个额外的约束: 1) 重构作业弧时不能违背工件的工艺约束关系; 2) 重构作业弧时不能将一个工件的工序与其余工件的作业弧相连.

为了确保算法能够执行可行的邻域动作, 本文结合 Xie 等[5] 提出的 N8 邻域结构, 对所提出新型邻域结构的可行性进行讨论. 对于某个工件J中的两道工序u和v, 在某个可行解中u在v之前执行; 将该解中属于工件J且位于工序u与v之间执行的工序集合记为$L=\{l_1,l_2,\ldots,l_n\}$; 属于工件J且在工序u前执行的工序集合记为$L^-=\{l_1^-,l_2^-,\ldots,l_{n^-}^-\}$; 属于工件$J$且在工序$v$后执行的工序集合记为$L^+=\{l_1^+,l_2^+,\ldots,l_{n^+}^+\}$. 此时有如下两个命题.

命题 1 如果工序u和v同时满足式 (10) ∼ (12), 则将工序u在工件J中移动到工序v之后执行, 不会在合取图中产生回路, 也不会违背工件J的工艺约束关系 (式 (6)). 有

$$S_{\mathrm{MS}(u)}+\mathbb{T}_{\mathrm{MS}(u)}>S_v, \tag{10}$$

$$y_{uv}\neq 1, \tag{11}$$

$$\sum_{i\in L\bigcup\{v\}}a_{ui}=0. \tag{12}$$

证明 首先运用反证法证明该移动不会在合取图中产生回路. 假设将工序u移动到工序v之后在合取图中产生了一个回路C, 则C一定会包含工序u, 产生的回路C共有 4 种可能的情况:

1) 当C为$u\to \mathrm{JS}(v)\to\ldots\to v\to u$的情况时, 原合取图中必定包含$v\to \mathrm{JS}(v)\to\ldots\to v$形式的回路, 与原合取图中不包含回路的条件冲突, 因此不成立.

2) 当C为$u\to \mathrm{JS}(v)\to\ldots\to \mathrm{MP}(u)\to u$的情况时, 原合取图中必定包含$\mathrm{JS}(v)\to \mathrm{MP}(u)\to u\to\ldots\to v\to \mathrm{JS}(v)$形式的回路, 与原合取图中不包含回路的条件冲突, 因此不成立.

3) 当C为$u\to \mathrm{MS}(u)\to\ldots\to v\to u$且$\mathrm{MS}(u)\neq v$时, 原合取图中存在$\mathrm{MS}(u)\to\ldots\to v$的有向链路, 此时必有$S_{\mathrm{MS}(u)}+\mathbb{T}_{\mathrm{MS}(u)}\leqslant S_v$, 与式 (10) 冲突, 因此不成立.

4) 当C为$u\to v\to u$时, 在原合取图中存在机器弧$u\to v$与式 (11) 冲突, 因此不成立.

下面证明该移动不会违背工件J的约束关系 (式 (6)). 设i与j均为工件J上的两个不同工序, 下面进行分类讨论:

1) 当$i\in L^-\bigcup L^+$时, 对于$\forall j\in J$, z_{ij}的值不变, 工序i一定满足式 (6).

2) 当$i\in L\bigcup\{v\}$时, 对于$\forall j\in L^-\bigcup L\bigcup\{v\}\bigcup L^+$, z_{ij}的值不变, 工序i一定满足式 (6). 对于$j=u$的情况, z_{ij}由 0 变为 1, 由式 (12) 易得$a_{ij}=0,\forall i\in L\bigcup\{v\}$, 也一定满足式 (6).

3) 当$i=u$时, 对于$\forall j\in L^-\bigcup L^+$, z_{ij}的值不变, 工序i一定满足式 (6). 对于$j\in L\bigcup\{u\}$的情况, z_{ij}由 1 变为 0, 由式 (12) 易得$a_{ij}=0,\forall i\in L\bigcup\{v\}$, 也一定满足式 (6).

综上可以证明, 当满足式 (10) ∼ (12) 时, 将工序u在工件J上移动到工序v之后不会在合取图中产生回路, 也不会违背工件J的工艺约束关系. □

命题 2 如果工序u和v同时满足式 (13) ∼ (15), 则将工序v在工件J中移动到工序u之前执行, 不会在合取图中产生回路, 也不会违背工件J的工艺约束关系 (式 (6)). 有

$$S_u+\mathbb{T}_u>S_{\mathrm{MP}(v)}, \tag{13}$$

$$y_{uv}\neq 1, \tag{14}$$

$$\sum_{i\in L\bigcup\{u\}}a_{iv}=0. \tag{15}$$

命题 2 证明同命题 1, 此略.

根据命题 1 和命题 2 可对位于作业关键工序块中的工序在工件上进行移动, 得到可行的邻域解. 据此, 本文设计一种考虑顺序柔性的新型邻域结构, 这里以$\mathrm{N_{SF1}}$代指本文所提出的邻域结构.

3 基于变邻域禁忌搜索的调度算法

综合所提出的新型邻域结构与现有对 JSP 和 FJSP 所设计的邻域结构, 本文设计一种变邻域禁忌搜索算法用以高效求解 FJSP-SF, 算法流程如图 5 所示.

3.1 解的初始化

在求解车间调度类问题时, 为了加快算法搜索效率以及求解的稳定性. 本文使用基于嵌套规则的算法生成初始解[19]. 调度规则如下:

1) 最早开工时间规则: 对工序进行调度时, 始终选择具有最早开始时间的工序进行优先调度.

2) 最早完成时间规则: 若多个工序的最早开工时间相同时, 则选择具有最小完成时间的工序进行优先调度.

3) 最大剩余时间最小调度规则: 在多个工序的

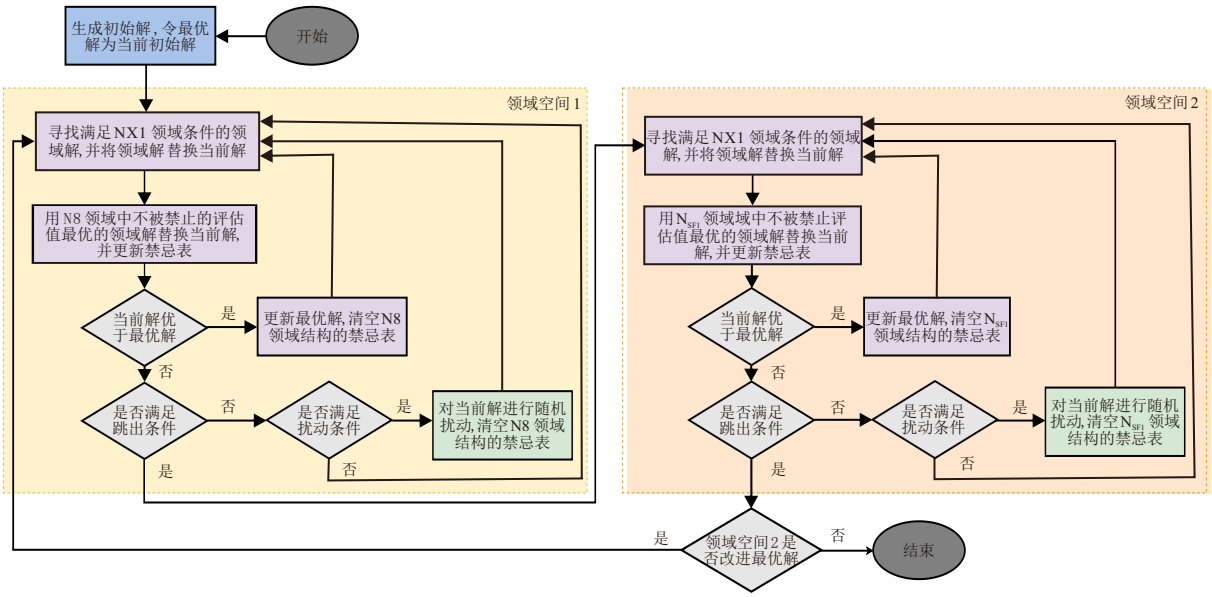


图5　变邻域禁忌搜索算法流程

最早开工时间以及完成时间相同时, 选择其所属工件未调度工序完成时间的最大值之和最小的工序进行优先调度.

4) 在以上 3 个调度规则仍然无法决定优先调度的工序时, 按照工序原有编号升序进行优先调度.

3.2　邻域结构的使用

本文使用N_{SF1}邻域结构、黄学文等[30]提出的 NX1 邻域结构和 Xie 等[5]提出的 N8 邻域结构, 分别应用于工序在工件、跨机器和单机器上的移动. 由于 NX1 邻域结构能减少关键路径或最大完工时间, 且其邻域空间较小, 将 NX1 与 N8 结合成邻域空间 1, NX1 与N_{SF1}结合成邻域空间 2, 采用变邻域禁忌搜索算法在这两个邻域空间中迭代改进解.

3.3　邻域评估和邻域裁剪

为加快算法的求解效率, 针对 N8 邻域结构, 使用 Balas 等[27]提出的邻域评估方法快速地对邻域移动进行评估. 同时, 使用 Xie 等[5]提出的邻域删减规则减少邻域搜索的空间. 通过将“作业弧”与“机器弧”互换, 并用“作业关键工序块”替代“关键工序块”, 可将 N8 邻域结构的评估和裁剪规则自然地应用到N_{SF1}邻域上. 参考文献 [5, 26-27] 可对本文所用邻域评估和邻域裁剪方法进行进一步地了解.

3.4　搜索准则

本文所提出变邻域禁忌搜索算法使用以下两个搜索准则以加强求解能力:

1) 为防止算法陷入循环迭代, 采用禁忌准则. 针对 N8 和N_{SF1}邻域结构, 分别建立禁忌表, 禁止最近几次的邻域移动. 由于 NX1 邻域结构必定能够减少关键路径的数目或最大完工时间, 不针对 NX1 邻域结构构建禁忌表.

2) 为使算法能够跳出局部最优, 使用随机扰动准则, 即若算法在邻域空间 1 (或邻域空间 2) 中迭代 N_0 次后均无法找到更优解, 则对当前解进行随机扰动, 随机改变 M 个工序的加工机器及其在机器上的加工顺序.

4　算例实验

4.1　测试算例介绍

尽管 Birgin 等[18]构建的 YFJS 测试集 (20 个算例) 和 DAFJ 测试集 (30 个算例) 并非为 FJSP-SF 设计, 针对这两个测试集的求解结果往往也不是针对 FJSP-SF, 而是针对 Birgin 所提出的车间调度问题[9,31-33], 但其工艺路线约束与 FJSP-SF 相似. 因此, YFJS 和 DAFJ 测试集可作为 FJSP-SF 的测试集. 由于缺乏对 FJSP-SF 标准测试集的实验结果[34], 在 YFJS 和 DAFJ 两个测试集上仅将所提出算法 (VNTSA) 与通用数学规划求解器 CPLEX 进行对比, 并分析 VNTSA 各个组件的作用.

4.2　测试环境及算法参数

本文所提出 VNTSA 算法使用 C++编写, 编译器为 Visual Studio 2022, 数学优化求解器为 CPLEX 20.1, 个人计算机配置如下: AMD Ryzen 9950X 4.30 GHz, 64 GB 内存, 操作系统为 Windows 11. 根据现有针对 JSP 以及 FJSP 的研究[5,7], VNTSA 算法参数设置如下:

1) Zhang 等[28]指出 JSP 的禁忌表长度应该与问题规模成正比. 由于 JSP 与 FJSP-SF 具有相似性, 本文根据预实验的结果按照如下公式计算 VNTSA 算法中的 N8 和N_{SF1}邻域结构的禁忌表长度L_{tabu}:

$$L_{\text{tabu}} = \left\lfloor \frac{|\mathcal{O}|}{|\mathcal{F}|} \right\rfloor + 10. \tag{16}$$

其中: $|\mathcal{O}|$、$|\mathcal{F}|$分别为所有工件工序总数与机器总数, $\lfloor a \rfloor$为不超过实数a的最大整数.

2) 根据预实验结果, 每次随机扰动时移动工序的个数M计算如下:

$$M = \left\lfloor \frac{|\mathcal{O}|}{50} \right\rfloor + 2. \tag{17}$$

3) 邻域空间 1 和邻域空间 2 的扰动代数在 VNTSA 中设计为相同的值N_0. 根据预实验结果, 当$N_0 = 500$时能取得较好的求解效果.

4) 根据预实验结果, 设置当单个邻域空间中连续迭代 10 000 次没有改进最优解或者总迭代次数超过 50 000 次后, 跳出该邻域空间.

4.3 算法性能测试及结果分析

本文对每个测试算例采用 VNTSA 算法进行 20 次独立实验, 并与 CPLEX 求解器在设置 300 s 和 1 h 求解时间限制下的结果进行对比分析. 表 3 和表 4 分别展示了 YFJS 测试集和 DAFJ 测试集的求解结果. 其中: "*" 表示所得解为该算例的最优解; 加粗字体表示针对算例所找到的已知最好解; "-" 表示 CPLEX 在规定时间的限制内未找到可行解; "LB" 代表问题下界, 由 CPLEX 求解 2 h 获得. 本文使用下式对所提出算法在某一算例上的求解效果进行量化比较, 其含义为算法针对算例的平均求解结果与该算例已知最好解的偏差:

$$\text{GAP} = \frac{M_{\text{Ave}} - M_{\text{Best}}}{M_{\text{Best}}}. \tag{18}$$

其中: M_{Ave}为算法在该算例上求解结果的均值, M_{Best}为针对该算例本研究所得到的最好解.

根据表 3 和表 4, 在 YFJS 测试集上, 针对 YFJS01 ∼ YFJS13 等 CPLEX 成功找到最优解的小规模算例, VNTSA 所得解与最优解的偏差平均不超过 4%. 在 YFJS14 ∼ YFJS20 等大规模算例上, VNTSA 算法在 120 s 内的运行结果优于或等于 CPLEX 运行 1 h 所得到的解并在部分算例上求得了最优解. 在 DAFJ 测试集上, VNTSA 在任何算例上的求解结果都优于或等于 CPLEX 运行 1 h 所得到的结果.

在 CPLEX 所找到的 DAFJ 测试集中最优解的算例上, VNTSA 每次运行也都找到了对应的最优解. 实验结果表明: VNTSA 在多数情况下 (尤其大规模问题) 性能优于 CPLEX, 但部分小规模算例 (YFJS01 ∼ YFJS13) 的解质量劣于 CPLEX. 同时, VNTSA 能否获得最优解与问题规模无显著关联 (如未获得 YFJS02 最优解但获得 YFJS17 最优解), 而 CPLEX 寻优能力随规模增大明显下降.

VNTSA 与 CPLEX 在不同规模算例上的性能差异体现了元启发式方法与精确方法的本质区别[35]:

表3 VNTSA 与 CPLEX 在 YFJS 数据集上的对比

算例	($\|\mathcal{O}\|$, $\|\mathcal{F}\|$)	LB	VNTSA			CPLEX		
			Best	Ave	time/s	300 s	1 h	time/s
YFJS01	(40, 7)	832	**832***	845.7	3.78	**832***	**832***	14.21
YFJS02	(40, 7)	899	911	925.2	2.60	**899***	**899***	14.36
YFJS03	(24, 7)	366	**366***	370.9	0.37	**366***	**366***	1.55
YFJS04	(28, 7)	390	**390***	407.0	0.25	**390***	**390***	3.73
YFJS05	(32, 7)	458	465	469.7	0.36	**458***	**458***	10.26
YFJS06	(36, 7)	450	456	462.6	0.45	**450***	**450***	93.25
YFJS07	(36, 7)	462	465	470.0	0.39	**462***	**462***	28.02
YFJS08	(36, 12)	360	368	373.4	0.17	**360***	**360***	1.21
YFJS09	(36, 12)	254	**254***	261.8	0.74	**254***	**254***	15.40
YFJS10	(40, 12)	440	**440***	**440.0***	0.24	**440***	**440***	0.87
YFJS11	(50, 10)	568	**568***	569.8	0.73	**568***	**568***	4.18
YFJS12	(50, 10)	555	**555***	557.8	0.83	**555***	**555***	7.01
YFJS13	(50, 10)	420	424	432.3	0.85	**420***	**420***	31.00
YFJS14	(221, 26)	1 324	**1 421**	**1 421.0**	17.24	—	**1 421**	3 600
YFJS15	(221, 26)	1 246	**1 342**	1 342.9	39.33	—	**1 342**	3 600
YFJS16	(221, 26)	1 331	**1 331***	**1 331.0***	31.15	—	**1 331***	3 600
YFJS17	(289, 26)	1 133	**1 133***	1 145.9	100.42	—	—	3 600
YFJS18	(289, 26)	1 220	**1 338**	1 338.6	61.12	—	—	3 600
YFJS19	(289, 26)	949	**1 041**	1 065.7	139.01	—	—	3 600
YFJS20	(289, 26)	968	**1 055**	1 072.1	100.16	—	—	3 600

注: CPLEX 在 1 h 内求得 YFJS16 的最优解, 但未证明最优性;
CPLEX 求解 2 h 给出 YFJS17 的下界与 VNTSA 所得最优解相等, 从而表明了该解的最优性.

表4　VNTSA 与 CPLEX 在 DAFJ 数据集上的对比

算例	(|$\mathcal{O}$|, |$\mathcal{F}$|)	LB	VNTSA			CPLEX		
			Best	Ave	time/s	300 s	1 h	time/s
DAFJ01	(26, 5)	402	**402***	**402.0***	0.29	**402***	**402***	1.76
DAFJ02	(25, 5)	502	**502***	**502.0***	0.14	**502***	**502***	2.42
DAFJ03	(55, 10)	724	**724***	**724.0***	8.55	**724***	**724***	42.86
DAFJ04	(43, 10)	717	**717***	**717.0***	0.29	**717***	**717***	5.12
DAFJ05	(39, 5)	626	**626***	**626.0***	2.24	**626***	**626***	84.87
DAFJ06	(44, 5)	574	**574***	**574.0***	3.85	**574***	**574***	256.26
DAFJ07	(85, 10)	913	**992**	**992.0**	17.19	—	**992**	3 600
DAFJ08	(85, 10)	880	**1 172**	**1 172.0**	27.22	**1 172**	**1 172**	300
DAFJ09	(45, 5)	469	**485**	493.4	1.62	509	495	3 600
DAFJ10	(58, 5)	493	**544**	560.4	9.00	623	553	3 600
DAFJ11	(113, 10)	836	**1 077**	**1 077.0**	42.69	—	**1 077**	3 600
DAFJ12	(117, 10)	855	**1 156**	**1 156.0**	90.70	—	**1 156**	3 600
DAFJ13	(62, 5)	543	**654**	666.4	7.08	711	696	3 600
DAFJ14	(69, 5)	551	735	744.8	8.21	882	762	3 600
DAFJ15	(120, 10)	770	**1 098**	**1 098.0**	28.07	—	**1 098**	3 600
DAFJ16	(120, 10)	1 070	**1 285**	**1 285.0**	96.66	—	**1 285**	3 600
DAFJ17	(82, 5)	562	**796**	813.9	20.33	988	854	3 600
DAFJ18	(74, 5)	524	**788**	806.4	7.18	883	838	3 600
DAFJ19	(70, 7)	820	**820***	**820.0***	3.82	**820***	**820***	162.23
DAFJ20	(92, 7)	591	765	776.2	35.19	972	766	3 600
DAFJ21	(107, 7)	640	**808**	817.1	47.90	1 123	849	3 600
DAFJ22	(116, 7)	615	**698**	712.4	271.51	—	770	3 600
DAFJ23	(76, 9)	767	**767***	**767.0***	5.30	**767***	**767***	300
DAFJ24	(92, 9)	798	**1 136**	**1 136.0**	47.82	1 145	**1 136**	3 600
DAFJ25	(123, 9)	842	**1 157**	**1 157.0**	71.78	—	**1 157**	3 600
DAFJ26	(119, 9)	707	**926**	936.6	83.30	—	934	3 600
DAFJ27	(127, 9)	759	**1 087**	**1 087.0**	74.61	—	**1 087**	3 600
DAFJ28	(91, 10)	816	**816***	**816.0***	6.03	844	**816***	1 485.25
DAFJ29	(95, 10)	1 013	**1 395**	**1 395.0**	20.53	1 440	**1 395**	3 600
DAFJ30	(98, 10)	731	**947**	**947.0**	10.11	**947**	**947**	300

注: CPLEX在300 s内求得DAFJ23的最优解, 但1 h内无法验证其最优性.

对于小规模问题, 解空间有限, CPLEX 凭借分支定界等策略可实现高效的系统性全局搜索, 从而保证最优解; 但随着问题规模扩大, 复杂度呈指数级增长, 此类精确方法往往难以在可接受的时间内完成求解. 反之, VNTSA 作为基于邻域搜索的元启发式算法, 虽不能保证最优性, 但能在大规模复杂实例中维持计算可行性; 而在小规模场景下, 由于搜索空间受限, 其更易陷入局部最优, 难以稳定收敛至全局最优解.

4.4　算法消融实验及结果分析

同时, 为了进一步验证所提出算法各个组件的作用, 分别移除 VNTSA 算法中的邻域空间 1、邻域空间 2、扰动准则、禁忌准则 (依次记作VNTSA^{1-}、VNTSA^{2-}、VNTSA^{3-}、VNTSA^{4-}), 将其与原算法进行对比, 每个算例独立测试 20 次. YFJS 测试集和 DAFJ 测试集上的求解结果分别列于表 5 和表 6.

从表 5 和表 6 可以发现, 在 VNTSA 中各个组件都对算法求解效果具有重要作用, 任意组件的移除都会导致算法效果的下降. 具体而言, 在 YFJS 测试集上, VNTSA、VNTSA^{1-}、VNTSA^{2-}、VNTSA^{3-}以及VNTSA^{4-}对各个算例的求解结果与已知最优解的偏差均值分别为 1.70%、2.45%、2.56%、3.48%、4.17%, 最大值分别为 4.36%、8.02%、8.92%、13.54%、13.27%. 在 DAFJ 测试集上, VNTSA、VNTSA^{1-}、VNTSA^{2-}、VNTSA^{3-}以及VNTSA^{4-}对各个算例求解结果与已知最优解的偏差均值分别为 0.64%、3.28%、0.71%、0.75%、2.33%, 最大值分别为 3.01%、13.34%、3.29%、3.42%、9.03%. 可以发现, 在不移除任意组件的情况下, VNTSA 算法具有最好的求解结果, 表现最佳, 求解结果稳定, 偏差均在 5% 以内.

同时, 原始 VNTSA 算法相较于移除组件的 VNTSA 算法, 在 YFJS 测试集和 DAFJ 测试集上分别找到了最多的已知最好解: YFJS 上 14/20 个算例, DAFJ 上 28/30 个算例.

以上实验结果表明了本文所设计 VNTSA 中各

表5 移除不同组件的 VNTSA 在 YFJS 数据集上的对比

算例	VNTSA			$VNTSA^{1-}$			$VNTSA^{2-}$			$VNTSA^{3-}$			$VNTSA^{4-}$		
	Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s
YFJS01	**832***	845.7	3.78	836	843.8	0.32	906	906.25	3.00	846	895.3	2.05	**832***	850.7	11.57
YFJS02	911	925.2	2.60	911	921.8	0.26	944	944	1.86	911	935.0	1.89	913	932.4	6.41
YFJS03	**366***	370.9	0.37	**366***	373.2	0.14	375	375.8	0.23	379	379.0	0.10	**366***	375.7	1.21
YFJS04	**390***	407.0	0.25	392	407.8	0.10	406	407.5	0.35	406	410.9	0.09	393	407.3	0.98
YFJS05	465	469.7	0.36	472	477.8	0.13	465	468.9	0.67	520	520.0	0.07	471	475.6	1.13
YFJS06	456	462.6	0.45	456	486.1	0.16	459	465.1	0.86	468	477.5	0.11	463	479.1	1.35
YFJS07	465	470.0	0.39	472	487.6	0.16	467	468.3	0.67	492	492.0	0.09	467	490.7	1.57
YFJS08	368	373.4	0.17	**360***	369.7	0.15	368	373.6	0.30	377	377.0	0.09	360*	372.2	1.42
YFJS09	**254***	261.8	0.74	255	263.3	0.15	258	266.7	0.89	258	267.8	0.43	255	263.9	2.36
YFJS10	**440***	**440***	0.24	**440***	**440.0***	0.08	**440***	**440***	0.36	**440***	**440.0***	0.12	**440***	**440***	1.06
YFJS11	**568***	569.8	0.73	**568***	**568.0***	0.11	**568***	568.8	1.01	**568***	569.4	0.24	**568***	576.2	2.25
YFJS12	**555***	557.8	0.83	571	578.2	0.22	**555***	557.3	1.57	**555***	558.8	0.73	560	572.0	1.95
YFJS13	424	432.3	0.85	**420***	441.7	0.22	423	431.3	8.63	423	431.9	0.47	432	444.5	2.07
YFJS14	**1421**	**1421.0**	17.24	**1421**	**1421.0**	9.24	**1421**	**1421.0**	27.06	**1421**	1421.0	14.30	**1421**	1424.5	92.89
YFJS15	**1342**	1342.9	39.33	**1342**	1342.1	14.36	**1342**	1346.7	38.47	**1342**	1349.1	26.42	**1342**	1371.5	168.91
YFJS16	**1331**	**1331.0**	31.15	**1331**	**1331.0**	11.29	1336	1342.4	29.74	**1331**	1336.3	21.41	**1331**	1363.3	84.75
YFJS17	**1133**	1145.9	100.42	1141	1147.0	59.20	1141	1146.3	134.25	1137	1144.7	90.92	1169	1203.2	307.43
YFJS18	**1338**	1338.6	61.12	**1338**	**1338.0**	26.76	**1338**	1339.3	66.78	**1338**	1342.6	34.31	1342	1368.9	249.16
YFJS19	**1041**	1065.7	139.01	1044	1059.9	85.50	1080	1100.0	84.11	1054	1087.1	66.04	1114	1179.1	242.78
YFJS20	**1055**	1072.1	100.16	1058	1065.5	82.35	1066	1075.7	86.53	1058	1069.6	73.08	1107	1146.2	164.10

表6 移除不同组件的 VNTSA 在 DAFJ 数据集上的对比

算例	VNTSA			$VNTSA^{1-}$			$VNTSA^{2-}$			$VNTSA^{3-}$			$VNTSA^{4-}$		
	Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s
DAFJ01	**402***	**402.0***	0.29	**402***	402.6	0.19	**402***	402.9	0.33	410	410.0	0.22	**402***	402.7	1.58
DAFJ02	**502***	**502.0***	0.14	**502***	**502.0***	0.11	**502***	**502.0***	0.20	**502***	**502.0***	0.13	**502***	**502.0***	0.85
DAFJ03	**724***	**724.0***	8.55	**724***	724.2	0.17	**724***	724.6	12.36	**724***	724.6	6.58	**724***	724.7	9.58
DAFJ04	**717***	**717.0***	0.29	**717***	717.7	0.15	**717***	**717.0***	0.34	**717***	**717.0***	0.20	717*	717.5	2.15
DAFJ05	**626***	**626.0***	2.24	**626***	**626.0***	2.05	**626***	**626.0***	2.74	**626***	**626.0***	2.14	**626***	626.7	0.43
DAFJ06	**574***	**574.0***	3.85	**574***	**574.0***	3.75	**574***	**574.0***	3.68	**574***	**574.0***	2.67	**574***	574.2	4.36
DAFJ07	**992**	**992.0**	17.19	**992**	**992.0**	11.61	**992**	**992.0**	19.56	**992**	**992.0**	20.23	**992**	992.3	46.32
DAFJ08	**1172**	**1172.0**	27.22	**1172**	**1172.0**	15.09	**1172**	**1172.0**	31.94	**1172**	**1172.0**	26.86	**1172**	**1172.0**	53.10
DAFJ09	**485**	493.4	1.62	505	521.1	1.01	**485**	491.3	2.61	491	497.4	0.94	499	511.4	2.51
DAFJ10	**544**	560.4	9.00	593	605.6	1.61	546	557.1	11.13	548	562.6	7.42	571	589.9	102.09
DAFJ11	**1077**	**1077.0**	42.69	**1077**	**1077.0**	27.31	**1077**	**1077.0**	58.51	**1077**	**1077.0**	37.11	**1077**	**1077.0**	152.93
DAFJ12	**1156**	**1156.0**	90.70	**1156**	**1156.0**	32.51	**1156**	**1156.0**	136.16	**1156**	**1156.0**	97.06	**1156**	**1156.0**	269.40
DAFJ13	**654**	666.4	7.08	723	735.4	13.81	656	664.4	6.78	658	667.6	6.14	688	704.6	5.54
DAFJ14	735	744.8	8.21	809	828.5	10.59	742	751.7	8.07	**731**	745.8	7.37	778	797.0	7.04
DAFJ15	**1098**	**1098.0**	28.07	**1098**	**1098.0**	19.84	**1098**	**1098.0**	35.50	**1098**	**1098.0**	25.28	**1098**	1098.8	63.52
DAFJ16	**1285**	**1285.0**	96.66	**1285**	**1285.0**	52.85	**1285**	**1285.0**	131.47	**1285**	**1285.0**	83.79	**1285**	1285.1	208.84
DAFJ17	**796**	813.9	20.33	863	891.5	41.14	808	822.2	17.81	805	815.7	20.39	828	851.3	13.65
DAFJ18	**788**	806.4	7.18	856	885.4	12.12	794	808.0	7.17	796	803.4	7.49	827	847.5	7.87
DAFJ19	**820***	**820.0***	3.82	**820***	**820.0***	1.66	**820***	**820.0***	6.15	**820***	**820.0***	2.15	**820***	820.6	8.21
DAFJ20	765	776.2	35.19	789	808.9	56.29	765	771.5	39.64	**763**	774.6	26.30	774	796.3	37.73
DAFJ21	**808**	817.1	47.90	878	901.0	92.87	810	818.5	49.86	813	820.6	45.80	841	855.6	45.61
DAFJ22	**698**	712.4	271.51	755	772.7	487.32	704	716.8	232.93	702	710.8	254.26	733	759.8	187.06
DAFJ23	**767**	**767.0**	5.30	**767**	**767.0**	3.17	**767**	**767.0**	6.99	**767**	**767.0**	4.82	**767**	**767.0**	7.93
DAFJ24	**1136**	**1136.0**	47.82	**1136**	**1136.0**	35.99	**1136**	**1136.0**	47.75	**1136**	**1136.0**	48.64	**1136**	**1136.0**	75.02
DAFJ25	**1157**	**1157.0**	71.78	**1157**	**1157.0**	40.56	**1157**	**1157.0**	82.12	**1157**	**1157.0**	53.23	**1157**	1158.3	109.07
DAFJ26	**926**	936.6	85.30	927	935.7	68.61	932	943.2	57.76	**926**	935.1	79.65	948	966.6	63.05
DAFJ27	**1087**	**1087.0**	74.61	**1087**	**1087.0**	67.58	**1087**	**1087.0**	86.18	**1087**	**1087.0**	69.83	**1087**	1090.3	108.52
DAFJ28	**816***	**816.0***	6.03	**816***	**816.0***	2.50	**816***	**816.0***	9.12	**816***	**816.0***	4.44	**816***	816.1	6.53
DAFJ29	**1395**	**1395.0**	20.53	**1395**	**1395.0**	8.97	**1395**	**1395.0**	29.02	**1395**	**1395.0**	16.54	**1395**	**1395.0**	25.38
DAFJ30	**947**	**947.0**	10.11	**947**	**947.0**	8.39	**947**	**947.0**	13.82	**947**	**947.0**	10.11	**947**	947.3	24.49

个组件的有效性, 能有效提升算法对于 FJSP-SF 的全局寻优能力.

4.5　拓展对比实验及结果分析

由于 FJSP 是 FJSP-SF 的一种特殊形式, VNTSA 也适用于求解 FJSP. 为全面评估 VNTSA 的性能, 本文使用 Brandimarte 建立的 FJSP 标准测试集 BRdata (含 10 个算例, VNTSA 针对每个算例独立测试 20 次)[36], 将 VNTSA 与多种专门求解 FJSP 的算法进行对比, 对比算法包括: Ding 等[37] 的师徒进化算法 (master-apprentice evolutionary algorithm, MAE)、Sun 等[38] 的带有变邻域搜索的混合遗传算法 (hybrid genetic algorithm with variable neighborhood search, HGA-VNS)、Chen 等[39] 的基于强化学习的自学习遗传算法 (self-learning genetic algorithm based on reinforcement, SLGA), 以及 Ding等[40] 的混合人类学习优化的粒子群算法 (Hybrid of human learning optimization algorithm and particle swarm optimization algorithm, HLO-PSO). 对比结果见表 7.

表7　VNTSA 与不同算法在 BRdata 数据集上的对比

算例	LB[37]	MAE[37]			HGA-VNS[38]			SLGA[39]			HLO-PSO[40]			VNTSA		
		Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s	Best	Ave	time/s
MK01	40	**40***	**40.0***	0.20	**40***	**40.0***	—	**40***	42.0	27.63	40*	41.3	27.74	**40***	**40.0***	0.80
MK02	26	**26***	**26.0***	0.55	**26***	**26.0***	—	27	29.7	29.11	28	28.8	48.10	**26***	**26.0***	1.26
MK03	204	**204***	**204.0***	0.16	**204***	**204.0***	—	**204***	210.2	112.60	204*	204	447.96	**204***	**204.0***	0.68
MK04	60	**60***	**60.0***	0.47	60*	61.4	—	**60***	66.7	63.21	63	66.1	69.14	**60***	**60.0***	1.40
MK05	172	**172***	**172.0***	1.46	173	173.0	—	**172***	183.5	60.35	175	177.3	134.10	**172***	172.9	2.69
MK06	57	**57***	57.25	268.54	61	63.1	—	69	76.9	72.80	71	73.6	599.91	58	61.3	8.03
MK07	139	**139***	**139.0**	481.27	140	140.6	—	144	151.3	57.77	144	145.1	164.67	**139***	144.9	6.91
MK08	523	**523***	**523.0***	0.36	523*	523.0	—	**523***	526.6	521.69	**523***	**523.0***	554.21	**523***	**523.0***	12.13
MK09	307	**307***	**307.0***	1.13	307*	309.3	—	320	340.4	552.50	326	331.3	1 063.62	**307***	**307.0***	3.97
MK10	189	**193**	194.6	827.34	214	217.2	—	254	269.9	1 335.18	238	246.7	1 592.45	199	201.5	15.20

注: LB 来自参考文献 [36] 所列出的相关数据. "—" 代表原参考文献中未给出相关算法所用求解时间;
MAE 由 C++实现, 计算机配置为 Intel Xeon E5-2 697 2.60 GHz, 2 GB 内存, HGA-VNS 所采用的编程语言与硬件运行环境在原文中未予说明;
SLGA 由 Matlab 实现, 配置为 Intel(R) Core i5-4 590 3.30 GHz, 8 GB 内存, HLO-PSO 由 C++实现, 配置为 Intel(R) Core i7 3 GHz, 8 GB 内存.

由表 7 可见, 在 BRdata 数据集上, MAE、HGA-VNS、SLGA、HLO-PSO 和 VNTSA 所求得的平均结果相对于问题下界的平均偏差分别为 0.34%、3.27%、13.83%、9.92%、1.89%. 在求解效果方面, VNTSA 略低于 MAE 算法, 但稍优于 HGA-VNS 算法, 并显著优于 SLGA 和 HLO-PSO 算法. 同时, 在计算效率上, VNTSA 在求解 MK01–MK10 算例时所耗费的时间远少于 SLGA 和 HLO-PSO 算法, 与 MAE 算法的运行时间大致相当. 上述对比结果表明, VNTSA 在求解 FJSP 问题时仍具有较高的求解效率. 尽管其性能略逊于 MAE 这类高效启发式算法, 但 VNTSA 具有更广泛的通用性, 能够有效求解具有任意顺序柔性的车间调度问题.

5　结　论

在半导体制造、火箭发动机制造等高精尖的工业生产领域, 工件的加工工序通常并非严格遵循全序关系, 而是具有一定的顺序柔性. 为了解决这一生产制造中的实际问题, 本文研究了一类具有顺序柔性的车间调度问题, 构建了 FJSP-SF 的 0-1 整数规划模型, 以准确地描述问题和求解工序顺序柔性对生产调度的影响. 针对 FJSP-SF 中加工顺序柔性带来的挑战, 本文设计了一种新型的邻域结构N_{SF1}, 该结构能够充分考虑顺序柔性因素. 在此基础上, 进一步提出了一种变邻域禁忌搜索算法 VNTSA 以高效求解 FJSP-SF, 并通过计算实验验证了该算法的高效性. 实验结果表明, 该算法能够为实际工业生产制造提供科学的决策支持.

本文提出的变邻域禁忌搜索算法 VNTSA 有效解决了顺序柔性车间调度问题 FJSP-SF, 并展现出良好的应用前景. 未来的研究可从以下几个方面进行拓展: 1) 引入更多生产约束, 如机器维护、工件优先级及多目标优化, 以提升调度问题的现实性和适应性; 2) 结合人工智能和机器学习技术, 构建具有自适应学习能力的智能调度算法; 3) 研究不同柔性类型的生产调度问题, 开发更加高效和通用的算法框架; 4) 结合实时生产数据进行动态调度, 构建智能化实时调度平台. 通过这些创新, 车间调度算法的应用范围和效果将进一步拓展, 为智能制造领域提供更广泛的决策支持.

参考文献 (References)

[1] Zhou L P, Jiang Z B, Geng N, et al. Production and operations management for intelligent manufacturing: A systematic literature review[J]. International Journal of Production Research, 2022, 60(2): 808-846.

[2] 伊思嘉, 罗继亮, 李旭航, 等. 基于 Petri 网和人工势场的柔性制造启发式优化方法[J]. 控制与决策, 2024, 39(6): 1977-1985.
(Yi S J, Luo J L, L X H, et al. Heuristic optimal method of flexible manufacturing based on Petri nets and artificial potential field[J]. Control and Decision, 2024, 39(6): 1977-1985.)

[3] 王艳红, 付威通, 张俊, 等. 基于改进近端策略优化算法的柔性作业车间调度[J]. 控制与决策, 2025, 40(6): 1883-1891.
(Wang Y H, Fu W T, Zhang J, et al. Flexible job-shop scheduling based on improved proximal policy optimization algorithm[J]. Control and Decision, 2025, 40(6): 1883-1891.)

[4] 王晓明, 邱瑶, 沈焱, 等. 装备制造企业如何选择柔性制造策略 —— 基于制造合格率视角[J]. 控制与决策, 2022, 37(1): 213-218.
(Wang X M, Qiu Y, Shen Y, et al. How to choose flexible manufacturing strategy for equipment manufacturing enterprises — Based on perspective of manufacturing qualification rate[J]. Control and Decision, 2022, 37(1): 213-218.)

[5] Xie J, Li X Y, Gao L, et al. A new neighbourhood structure for job shop scheduling problems[J]. International Journal of Production Research, 2023, 61(7): 2147-2161.

[6] 朱家政, 张宏立, 王聪, 等. 基于深度强化学习的模糊作业车间调度问题[J]. 控制与决策, 2024, 39(2): 595-603.
(Zhu J Z, Zhang H L, Wang C, et al. Fuzzy job shop scheduling problem based on deep reinforcement learning[J]. Control and Decision, 2024, 39(2): 595-603.)

[7] Zhang J J, Lü Z P, Ding J W, et al. An effective local search algorithm for flexible job shop scheduling in intelligent manufacturing systems[J]. Engineering, 2025, 50: 117-127.

[8] Zheng Q L, Dai W, Peng C X, et al. A novel neighborhood structure for flexible job shop scheduling problem considering quality-efficiency coupling effect[J]. Computers & Industrial Engineering, 2025, 199: 110735.

[9] Cao Z C, Zhou L J, Hu B, et al. An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem[J]. Business & Information Systems Engineering, 2019, 61(3): 299-309.

[10] 桂林, 张春江, 李新宇. 具有工序顺序柔性的车间调度问题研究综述[J]. 工业工程, 2020, 23(2): 116-124.
(Gui L, Zhang C J, Li X Y. A review of research on workshop scheduling problems with flexible process sequence[J]. Industrial Engineering Journal, 2020, 23(2): 116-124.)

[11] Zhang S C, Li X, Zhang B W, et al. Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system[J]. European Journal of Operational Research, 2020, 283(2): 441-460.

[12] Ghaedy-Heidary E, Nejati E, Ghasemi A, et al. A simulation optimization framework to solve stochastic flexible job-shop scheduling problems — Case: Semiconductor manufacturing[J]. Computers & Operations Research, 2024, 163: 106508.

[13] Yuan S, Zhu X M, Cai W, et al. Mathematical modeling and hybrid evolutionary algorithm to schedule flexible job shop with discrete operation sequence flexibility[J]. Computers & Operations Research, 2025, 176: 106952.

[14] 付梅, 谢怀, 王苏馨, 等. 复杂壳体零件制造车间调度方法研究[J]. 航空制造技术, 2021, 64(17): 85-93.
(Fu M, Xie H, Wang S X, et al. Research on scheduling method of complex shell part manufacturing shop floor[J]. Aeronautical Manufacturing Technology, 2021, 64(17): 85-93.)

[15] Gong G L, Tang J Q, Huang D, et al. Energy-efficient flexible job shop scheduling problem considering discrete operation sequence flexibility[J]. Swarm and Evolutionary Computation, 2024, 84: 101421.

[16] Xie J, Gao L, Peng K K, et al. Review on flexible job shop scheduling[J]. IET Collaborative Intelligent Manufacturing, 2019, 1(3): 67-77.

[17] Dauzère-Pérès S, Ding J W, Shen L J, et al. The flexible job shop scheduling problem: A review[J]. European Journal of Operational Research, 2024, 314(2): 409-432.

[18] Birgin E G, Feofiloff P, Fernandes C G, et al. A MILP model for an extended version of the flexible job shop problem[J]. Optimization Letters, 2014, 8(4): 1417-1431.

[19] Birgin E G, Ferreira J E, Ronconi D P. List scheduling and beam search methods for the flexible job shop scheduling problem with sequencing flexibility[J]. European Journal of Operational Research, 2015, 247(2): 421-440.

[20] 王雷, 蔡劲草, 唐敦兵, 等. 基于改进遗传算法的柔性作业车间调度[J]. 南京航空航天大学学报, 2017, 49(6): 779-785.
(Wang L, Cai J C, Tang D B, et al. Flexible job shop scheduling based on improved genetic algorithm[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2017, 49(6): 779-785.)

[21] Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem[J]. Computers & Operations Research, 2008, 35(10): 3202-3212.

[22] Meng L, Cheng W, Zhang B, et al. An improved genetic algorithm for solving the multi-AGV flexible job shop scheduling problem[J]. Sensors: Basel, 2023, 23(8): 3815.

[23] Hajibabaei M, Behnamian J. Flexible job-shop

scheduling problem with unrelated parallel machines and resources-dependent processing times: A tabu search algorithm[J]. International Journal of Management Science and Engineering Management, 2021, 16(4): 242-253.

[24] Vilcot G, Billaut J C. A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem[J]. International Journal of Production Research, 2011, 49(23): 6963-6980.

[25] Ding J W, Lü Z P, Li C M, et al. A two-individual based evolutionary algorithm for the flexible job shop scheduling problem[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, 33(1): 2262-2271.

[26] Nowicki E, Smutnicki C. A fast taboo search algorithm for the job shop problem[J]. Management Science, 1996, 42(6): 797-813.

[27] Balas E, Vazacopoulos A. Guided local search with shifting bottleneck for job shop scheduling[J]. Management Science, 1998, 44(2): 262-275.

[28] Zhang C Y, Li P G, Guan Z L, et al. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem[J]. Computers & Operations Research, 2007, 34(11): 3229-3242.

[29] 于丰顺, 赵诗奎, 仵政源, 等. 基于邻接矩阵的作业车间调度可行解判定方法[J]. 控制与决策, 2025, 40(10): 3085-3095.
(Yu F S, Zhao S K, Wu Z Y, et al. A feasible solution determination method for job shop scheduling based on adjacency matrix[J]. Control and Decision, 2025, 40(10): 3085-3095.)

[30] 黄学文, 陈绍芬, 周阗玉, 等. 求解柔性作业车间调度问题的一种新邻域结构[J]. 系统工程理论与实践, 2021, 41(9): 2367-2378.
(Huang X W, Chen S F, Zhou T Y, et al. A new neighborhood structures cheduling problem[J]. Systems Engineering — Theory and Practice, 2021, 41(9): 2367-2378.)

[31] Cao Z C, Lin C R, Zhou M C, et al. Scheduling semiconductor testing facility by using cuckoo search algorithm with reinforcement learning and surrogate modeling[J]. IEEE Transactions on Automation Science and Engineering, 2019, 16(2): 825-837.

[32] Cao Z C, Lin C R, Zhou M C. A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility[J]. IEEE Transactions on Automation Science and Engineering, 2021, 18(1): 56-69.

[33] Lin C R, Cao Z C, Zhou M C. Learning-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility[J]. IEEE Transactions on Cybernetics, 2023, 53(10): 6663-6675.

[34] Vital-Soto A, Azab A, Baki M F. Mathematical modeling and a hybridized bacterial foraging optimization algorithm for the flexible job-shop scheduling problem with sequencing flexibility[J]. Journal of Manufacturing Systems, 2020, 54: 74-93.

[35] Gendreau M, Potvin J Y. Handbook of metaheuristics[M]. New York: Springer, 2010.

[36] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41(3): 157-183.

[37] Ding J, L Lü Z, Li C M, et al. A two-individual based evolutionary algorithm for the flexible job shop scheduling problem[C]. Proceedings of The AAAI Conference on Artificial Intelligence. New York: AAAI, 2019, 33: 2262-2271.

[38] Sun K, Zheng D, Song H, et al. Hybrid genetic algorithm with variable neighborhood search for flexible job shop scheduling problem in a machining system[J]. Expert Systems with Applications, 2023, 215: 119359.

[39] Chen R, Yang B, Li S, et al. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem[J]. Computers & Industrial Engineering, 2020, 149: 106778.

[40] Ding H, Gu X. Hybrid of human learning optimization algorithm and particle swarm optimization algorithm with scheduling strategies for the flexible job-shop scheduling problem[J]. Neurocomputing, 2020, 414: 313-332.

作者简介

宁国宇 (1999−), 男, 硕士生, 主要研究方向为组合优化、智能优化算法、体系韧性优化与分析, E-mail: ningguoyu19@nudt.edu.cn;

陶汉桥 (2001−), 男, 硕士生, 主要研究方向为组合优化、数学规划理论、智能计算与优化决策, E-mail: taohanqiao19@nudt.edu.cn;

宋国鹏 (1991−), 男, 副教授, 博士, 主要研究方向为规划与调度、数学优化理论方法, E-mail: guopeng.song@nudt.edu.cn;

李明浩 (1990−), 男, 副研究员, 博士, 主要研究方向为体系设计与分析、智能优化算法, E-mail: liminghao4869@nudt.edu.cn;

杨克巍 (1977−), 男, 教授, 博士, 主要研究方向为装备体系规划与分析、智能决策与优化, E-mail: kayyang27@nudt.edu.cn.