中国科学: 技术科学 2023 年 第 53 卷 第 7 期: 1114 ~ 1126
SCIENTIA SINICA Technologica techcn.scichina.com
《中国科学》杂志社 SCIENCE CHINA PRESS
论 文 工业大数据与工业智能专辑

考虑夹具-托盘组合优化的多资源约束柔性作业车间智能调度

刘墨林[1], 周雨露[1], 王思阳[2], 张春明[2], 杜世昌[1*], 奚立峰[1]

1. 上海交通大学机械与动力工程学院工业工程与管理系, 上海 200240;
2. 潍柴动力股份有限公司, 潍坊 261061
* E-mail: lovbin@sjtu.edu.cn

收稿日期: 2022-10-16; 接受日期: 2022-12-26; 网络版发表日期: 2023-06-09
国家自然科学基金(批准号: 52275499)和国家重点研发计划(编号: 2022YFF0605700)资助项目

摘要 针对多产品混合加工车间中有限夹具-托盘资源引发的生产力制约问题, 提出了考虑夹具-托盘组合优化的多资源约束柔性作业车间智能调度方法. 首先, 以最小化最大完工时间为目标建立了基于设备-夹具-托盘三资源约束的混合整数规划模型. 其次, 设计了基于可行性修复和自学习型变邻域搜索的改良遗传算法并进行求解, 提出了种群染色体初始化、交叉、变异过程中的可行性修复策略, 在算法迭代中后期引入三种变邻域搜索方法并构建搜索策略知识库, 自学习地求得最优解. 最后, 基于工业大数据生成算例, 并通过数值实验证明了该算法具备求解精度和求解时间上的双重优越性, 可有效解决设备-夹具-托盘约束下的柔性作业车间调度问题, 为加工车间的生产排产智能化转型提供有力支持.

关键词 生产调度, 柔性作业车间, 夹具-托盘约束, 可行性修复策略, 自学习型变邻域搜索

1 引言

工业4.0时代, 伴随客户需求的日益个性化, 为适应激烈的市场竞争, 制造型企业的生产加工模式纷纷向以工业大数据和工业智能为驱动的柔性作业加工转变[1,2], 其典型特征为: (1) 产品种类多样化, 且不同产品的工艺路线和加工时间存在差异; (2) 产品的各加工工序可能存在多台可选择的加工设备. 具备上述特征的生产排产在学界被定义为柔性作业车间调度问题(flexible job shop scheduling problem, FJSP).

从20世纪80~90年代开始, 在作业车间调度问题(job shop scheduling problem, JSP)的基础上, 学界开展对柔性作业车间调度问题的研究[3], 并将其拆分为设备选择和工序排序两个子问题[4,5]. 而实际生产过程中, 人员、刀具、托盘、夹具等生产资源是限制柔性作业车间生产能力进一步提升的重要因素. 一些学者在广义的柔性作业车间调度问题的基础上, 研究了除加工设备外的其他生产资源对车间调度的影响. Lei和Guo[6]研究了人员-设备双资源约束情景下的柔性作业车间调度问题, 并利用变邻域搜索算法进行求解. 刘璐

引用格式: 刘墨林, 周雨露, 王思阳, 等. 考虑夹具-托盘组合优化的多资源约束柔性作业车间智能调度. 中国科学: 技术科学, 2023, 53: 1114–1126
Liu M L, Zhou Y L, Wang S Y, et al. Multiresource constrained flexible job shop intelligent scheduling considering fixture-pallet combinatorial optimization (in Chinese). Sci Sin Tech, 2023, 53: 1114–1126, doi: 10.1360/SST-2022-0334

等人[7]充分考虑了工人学习效应，设计了改进生物迁徙算法求解双资源约束问题. 除了人员以外，刀具、夹具等资源也已在部分研究工作的考虑之中. 李海[8]针对刀具数量和寿命有限的情况，以最小化生产成本为目标，建立了考虑刀具剩余寿命约束的双资源柔性作业车间调度模型并进行求解. Wu等人[9]考虑了实际加工过程中更换夹具对装卸时间的影响，设计了改良果蝇算法对柔性作业车间调度问题进行求解.

但是，目前几乎没有综合考虑设备-夹具-托盘三资源约束的智能调度研究工作. 夹具-托盘是柔性加工车间中不可忽略的辅助型生产资源, 一方面, 产品复杂的工艺流程决定了不同工序可能需要不同的夹具，即便是同一道工序也可能存在多个可选择的夹具；另一方面，由于加工设备体积大小、厂商配置等因素的影响，不同设备配备的托盘站负荷能力也存在差异，即可放置的夹具数量有所不同. 实际生产过程中，有限的夹具-托盘资源往往成为限制产能进一步突破的瓶颈, 甚至会带来生产停滞、延期的恶劣后果, 而不合理的夹具-托盘组合方式更会加剧这种影响. 因此，如何权衡有限的设备-夹具-托盘资源，选择最优的夹具-托盘组合方式，提供最佳的智能生产调度方案，是我国工业智能化转型必须要解决的问题.

本文研究了考虑夹具-托盘组合优化的多资源约束柔性作业车间智能调度(multi-resource constrained flexible job shop scheduling problem considering fixture-pallet combination optimization, MRFJSP-FPCO)，建立了考虑设备-夹具-托盘三资源约束的混合整数规划模型，并设计了可有效解决真实工业场景难题的智能优化算法——基于可行性修复和自学习型变邻域搜索的改良遗传算法，通过数值试验证明了算法的可行性和高效性.

2　问题定义与建模

2.1　问题定义

MRFJSP-FPCO问题描述如下：已知产品集合I、设备集合M、托盘站集合P和夹具集合F，J_i是产品i的工序集合. 每一台加工设备m都拥有一个托盘站p，即设备集合M和托盘站集合P的元素一一对应. 夹具和托盘在生产开始前完成绑定，然后放置在承载能力已知的托盘站内并且在生产过程中不再变动. 因此，夹具-托盘的绑定实际上可以看作夹具-托盘站的绑定，若无特殊说明，本文出现的夹具-托盘即为夹具和托盘站的组合. 产品的工序已知且固定, 并严格按照工艺路线进行加工. 每道工序O_{ij}都对应一个可选设备集$M(O_{ij})$和一个可选夹具集$F(O_{ij})$, 并且同一道工序在不同设备上的加工时间可能不同. 在满足各类约束的前提下, 为每道工序安排合适的加工设备、夹具和加工顺序并给出合理的夹具-托盘组合方式, 使得工件的总完工时间最小. 此外, MRFJSP-FPCO满足如下假设:

(1) 所有产品均可从零时刻开始加工;

(2) 每件产品的各道工序的加工过程连续, 加工一旦开始原则上不允许中断;

(3) 每台加工设备在同一时间至多加工一件产品的一道工序;

(4) 每一个夹具在同一时间至多加工一件产品的一道工序;

(5) 若产品的某几道连续工序可选择使用同一个夹具, 则优先使用该夹具;

(6) 夹具在生产过程中的装卸时间忽略不计;

(7) 不考虑只能选取不同设备的两道工序只能使用同一把刀具的极端情况，夹具资源有限但能够满足基本需求.

为了更直观地理解，以部分产品的部分工艺为例进行说明, 如表1所示.

表1包含了两种产品的工艺信息: 产品1和产品2均有三道工序，每道工序的可选设备集合和可选夹具集合如表中呈现. 同一道工序在不同加工设备上的加工时间可能存在差异，例如: 产品1的工序1在设备1上的加工时间为10, 在设备2上的加工时间为12. 若不考虑夹具-托盘约束，一种可行的生产计划甘特图如图1(a)所示; 而考虑夹具-托盘约束之后, 有限的夹具资源(图1(b))和多可能性的夹具-托盘组合方式(图1(c)和(d))都让生产排产的难度骤然增加.

2.2　混合整数规划模型

为了在设备-夹具-托盘三资源约束的复杂生产情景下同时给出最优的排产计划和夹具-托盘组合方案, 以最小化最大完工时间为目标, 遵循上述问题假设, 建立了MRFJSP-FPCO问题的混合整数规划模型.

模型所涉及的符号定义如表2所示.

MRFJSP-FPCO问题的混合整数规划模型如下:

表 1 产品工艺信息

Table 1 Products' process information

产品	工序	设备集合				夹具集合		
		设备1	设备2	设备3	设备4	夹具1	夹具2	夹具3
产品1	1	10	12	–	–	可用	可用	–
	2	8	8	–	–	可用	可用	–
	3	–	–	10	9	–	–	可用
产品2	1	8	10	–	–	可用	可用	–
	2	10	8	–	–	可用	可用	–
	3	–	–	7	7	–	–	可用

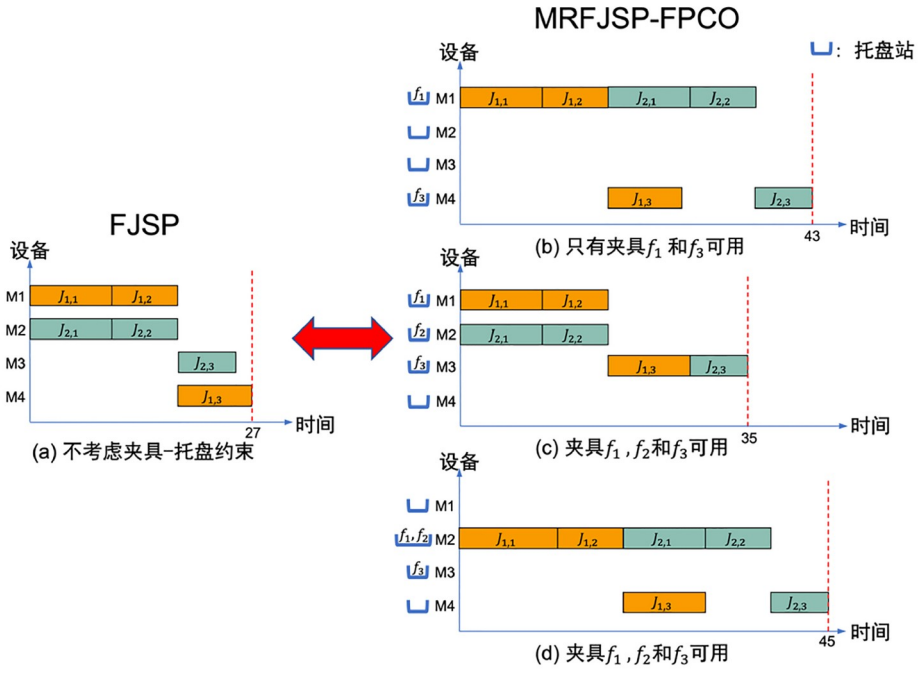

图 1 不同夹具-托盘组合情景下的生产计划甘特图

Figure 1 Gantt chart under different fixture-pallet combination scenarios.

$$\text{Minimize } C_{\max}$$

s.t.

$$\sum_{f\in F_{i,j}}\sum_{m\in M_{i,j}} X_{i,j,m,f}=1,\forall i\in I,\forall j\in J_i, \tag{1}$$

$$\sum_{f\in F} Z_{m,f}\leqslant P_m,\forall m\in M, \tag{2}$$

$$Z_{m,f}\geqslant 1-L\cdot\left(1-X_{i,j,m,f}\right),\quad \forall i\in I,\forall j\in J_i,\forall m\in M_{i,j},\forall f\in F_{i,j}, \tag{3}$$

$$\sum_{m'\in M_{i,j}\setminus m} Z_{m',f}\leqslant 0+L\cdot\left(1-X_{i,j,m,f}\right),\quad \forall i\in I,\forall j\in J_i,\forall m\in M_{i,j},\forall f\in F_{i,j}, \tag{4}$$

$$X_{i,j,m,f}\leqslant 0+L\cdot Z_{m,f},\quad \forall i\in I,\forall j\in J_i,\forall m\in M_{i,j},\forall f\in F_{i,j}, \tag{5}$$

$$X_{i,j+1,m,f}\geqslant 1-L\cdot\left(1-X_{i,j,m,f}\right),\quad \forall i\in I,\forall j\in J_i^f\cap J_i^m,\forall j+1\in J_i^f\cap J_i^m,\ \forall m\in M_{i,j}\cap M_{i,j+1},\forall f\in F_{i,j}\cap F_{i,j+1}, \tag{6}$$

表 2　混合整数规划模型符号定义

Table 2　Symbol definition of the MILP model

符号	定义
I	产品集合, i=1,2,⋯,$\|I\|$
M	加工设备集合, $m=1,2,\ldots,\|M\|$
F	夹具集合, $f=1,2,\ldots,\|F\|$
P	托盘系统集合, $p=1,2,\ldots,\|P\|$, $\|P\|=\|M\|$
J_i	产品i的工序j的工序集合, $i\in I$
$M_{i,j}$	产品i的工序j的可选加工设备集合, $i\in I$, $j\in J_i$
$F_{i,j}$	产品i的工序j的可选夹具集合, $i\in I$, $j\in J_i$
P_m	加工设备m的托盘系统可最大承载的夹具-托盘的数量
$pt_{i,j,m}$	产品i的工序j在设备m上的加工时间, $i\in I$, $j\in J_i$, $m\in M_{i,j}$
J_i^m	产品i的可选择相同加工设备的连续工序集合, $J_i^m\subset J_i$, $\left\|J_i^m\right\|\geqslant 2$
J_i^f	产品i的可选择相同夹具的连续工序集合, $J_i^f\subset J_i$, $\left\|J_i^f\right\|\geqslant 2$
L	一个大数, L>0
$S_{i,j}$	产品i的工序j的作业开始时间, $i\in I$, $j\in J_i$
$C_{i,j}$	产品i的工序j的作业结束时间, $i\in I$, $j\in J_i$
$C_{\max}$	所有产品的最大完成时间
$X_{i,j,m,f}$	0-1决策变量: 当产品i的工序j在加工设备m上使用夹具f进行加工时, 值为1; 其他情况, 值为0. $i\in I$, $j\in J_i$, $m\in M_{i,j}$, $f\in F_{i,j}$
$Y_{i,j,i',j',m}$	0-1决策变量: 当产品i的工序j和产品 i' 的工序 j' 均在加工设备m上进行加工且产品i的工序j在产品i' 的工序 j' 之前加工时, 值为1; 其他情况, 值为0. $i\in I$, $j\in J_i$, $i'\in I\setminus i$, $j'\in J_{i'}$, $m\in M_{ij}\cap M_{i'j'}$
$Y_{i,j,i',j',f}$	0-1决策变量: 当产品i的工序j和产品 i' 的工序 j' 均使用夹具f进行加工且产品i的工序j在产品 i' 的工序 j' 之前加工时, 值为1; 其他情况, 值为0. $i\in I$, $j\in J_i$, $i'\in I\setminus i$, $j'\in J_{i'}$, $f\in F_{ij}\cap F_{i'j'}$
$Z_{m,f}$	0-1决策变量: 当夹具f绑定加工设备m的托盘系统时, 值为1; 其他情况, 值为0. $m\in M$, $f\in F$

$$S_{i,j}+\sum_{m\in M_{i,j}}\left(p_{i,j,m}\cdot\sum_{f\in F_{i,j}}X_{i,j,m,f}\right)\leqslant C_{i,j},\quad \forall i\in I,\forall j\in J_i, \tag{7}$$

$$S_{i,j+1}\geqslant C_{i,j},\forall i\in I,\forall j\in J_i, \tag{8}$$

$$S_{i,j}\geqslant C_{i'j'}-L\cdot\left(2-\sum_{f\in F_{i,j}}X_{i,j,m,f}-\sum_{f\in F_{i',j'}}X_{i',j',m,f}\right)-L\cdot Y_{i,j,i',j',m},\quad \forall i\in I,\forall j\in J_i,\forall i'\in I\setminus i,\forall j'\in J_{i'},\forall m\in M_{i,j}\cap M_{i',j'}, \tag{9}$$

$$S_{i',j'}\geqslant C_{i,j}-L\cdot\left(2-\sum_{f\in F_{i,j}}X_{i,j,m,f}-\sum_{f\in F_{i',j'}}X_{i',j',m,f}\right)-L\cdot\left(1-Y_{i,j,i',j',m}\right),\quad \forall i\in I,\forall j\in J_i,\forall i'\in I\setminus i,\forall j'\in J_{i'},\forall m\in M_{i,j}\cap M_{i',j'}, \tag{10}$$

$$S_{i,j}\geqslant C_{i',j'}-L\cdot\left(2-\sum_{f\in F_{i,j}}X_{i,j,m,f}-\sum_{f\in F_{i',j'}}X_{i',j',m,f}\right)-L\cdot Y_{i,j,i',j',f},\quad \forall i\in I,\forall j\in J_i,\forall i'\in I\setminus i,\forall j'\in J_{i'},\forall f\in F_{i,j}\cap F_{i',j'}, \tag{11}$$

$$S_{i',j'} \geqslant C_{i,j} - L \cdot \left(2 - \sum_{f \in F_{i,j}} X_{i,j,m,f} - \sum_{f \in F_{i',j'}} X_{i',j',m,f}\right) - L \cdot \left(1 - Y_{i,j,i',j',f}\right), \quad \forall i \in I, \forall j \in J_i, \forall i' \in I \setminus i, \forall j' \in J_{i'}, \forall f \in F_{i,j} \cap F_{i',j'}, \tag{12}$$

$$Y_{i,j,i',j',m} \leqslant \sum_{f \in F_{i,j}} X_{i,j,m,f}, \forall i \in I, \forall j \in J_i, \forall m \in M_{i,j}, \tag{13}$$

$$Y_{i,j,i',j',m} \leqslant \sum_{f \in F_{i',j'}} X_{i',j',m,f}, \quad \forall i' \in I, \forall j' \in J_{i'}, \forall m \in M_{i',j'}, \tag{14}$$

$$Y_{i,j,i',j',f} \leqslant \sum_{m \in M_{i,j}} X_{i,j,m,f}, \forall i \in I, \forall j \in J_i, \forall f \in F_{i,j}, \tag{15}$$

$$Y_{i,j,i',j',f} \leqslant \sum_{m \in M_{i',j'}} X_{i',j',m,f}, \quad \forall i' \in I, \forall j' \in J_{i'}, \forall f \in F_{i',j'}, \tag{16}$$

$$C_{\max} \geqslant C_{i,j}, \forall i \in I, \forall j \in J_i, \tag{17}$$

$$X_{i,j,m,f} \in \{0,1\}, \forall i \in I, \forall j \in J_i, \forall m \in M_{i,j}, \forall f \in F_{i,j}, \tag{18}$$

$$Y_{i,j,i',j',m}, Y_{i,j,i',j',f} \in \{0,1\}, \forall i \in I, j \in J_i, \forall i' \in I \setminus i, \forall j' \in J_{i'}, \forall m \in M_{i,j} \cap M_{i',j'}, \forall f \in F_{i,j} \cap F_{i',j'}, \tag{19}$$

$$Z_{m,f} \in \{0,1\}, \forall m \in M, \forall f \in F, \tag{20}$$

$$S_{i,j}, C_{i,j}, C_{\max} \geqslant 0, \forall i \in I, \forall j \in J_i. \tag{21}$$

模型的目标为最小化最大完工时间. 式(1)保证了一件产品的一道工序只能选择一台加工设备以及一个夹具进行加工; 式(2)保证了每一台加工设备的托盘系统内的夹具-托盘数量不超过其最大承载能力; 式(3)~(5)约束了夹具和托盘一旦绑定就不会在生产过程中变动; 式(6)保证了如果产品的某几道连续工序可选择使用同一个夹具, 那么便选择此夹具(假设5); 式(7)表示一件产品的每道工序的加工开始时间和加工结束时间的关系; 式(8)表示一件产品的各道工序按照确定的先后顺序进行加工; 式(9)和(10)保证了同一台加工设备同一时间只能加工一件产品的一道工序; 式(11)和(12)保证了同一个夹具同一时间只能用于一件产品的一道工序; 式(13)~(16)描述了决策变量$Y_{i,j,i',j',m}$, $Y_{i,j,i',j',f}$和 $X_{i,j,m,f}$, $X_{i',j',m,f}$的相互关系; 式(17)定义了最大完工时间; 式(18)~(21)定义了各变量类型.

3 基于可行性修复与自学习型变邻域搜索的改良遗传算法

FJSP已被证明是NP-hard问题, 在多项式时间内难以精确求解[10]. MRFJSP-FPCO中, 有限的夹具-托盘资源约束和不确定的夹具-托盘组合方式, 增加了问题的求解难度. 本文提出了一种基于可行性修复和自学习型变邻域搜索的改良遗传算法(improved genetic algorithm hybrid with feasibility correction strategy and self-learning variable neighborhood search, IGA-FCSSVNS), 算法整体框架如图2所示. 考虑到设备-夹具-托盘三资源约束的复杂情景, 设计了三段式染色体结构, 并为了确保夹具-托盘的绑定关系在种群初始化和迭代优化过程中引入可行性修复策略; 在算法迭代中后期, 引入自学习型变邻域搜索策略, 从精英解池中提取变邻域搜索策略选择记录并构建策略知识库, 引导精英解在局部搜索过程中进一步优化.

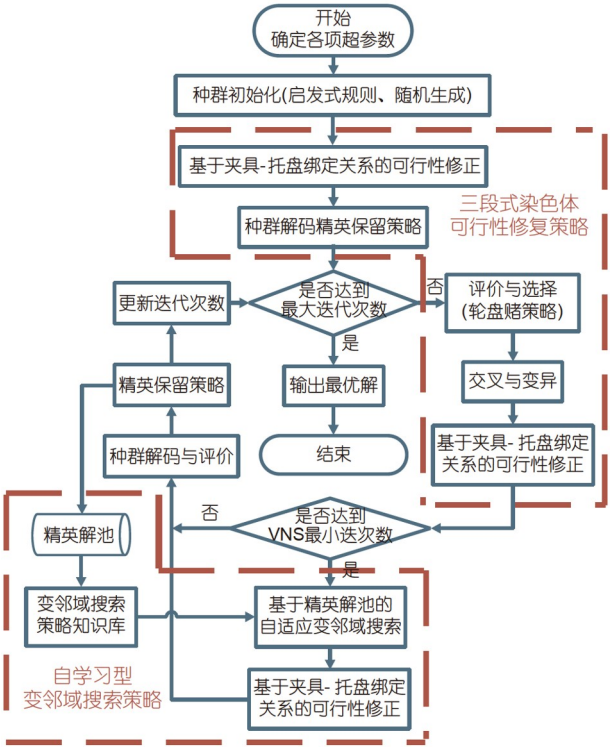

图 2 (网络版彩图) IGA-FCSSVNS算法框架

Figure 2 (Color online) Structure of IGA-FCSSVNS.

3.1　编码与解码

根据前述, TRFJSP-FPCO问题可拆解为工序排序(operation sequence, OS)、设备选择(machine selection, MA)和夹具选择(fixture selection, FI)三个子问题. 因此, 采用基于工序-设备-夹具的三段式编码方式, 如图3所示, 分别简记为OS, MA和FI. 每一段染色体的长度均等于所有产品的总工序数, OS段内的数字代表产品的编号, 出现的次序代表该产品的工序次序; MA段内的数字代表加工设备编号, 按照产品及其工序顺序排列; FI段内的数字代表夹具编号, 排序方式与MA段相同.

解码时, 由于夹具-托盘在生产启动前完成绑定并在生产过程中固定, 因此解码实际上等同于普通FJSP的解码, 在OS段从左到右遍历找到相应工序$O_{i,j}$ (其中i代表产品序号, j代表工序序号), 并在MA段获得已分配的加工设备M_k, 基于$O_{i,j}$在M_k上的实际加工时间进行排产, 直到所有工序都已调度.

3.2　可行化修复策略

在三段式染色体结构中, MA段和FI段的初始化和迭代等操作均是相互独立的, 为了满足夹具-托盘一旦完成组合就不再改动绑定关系的约束, 有必要设计面向MA段和FI段染色体的可行化修复策略.

设FI段中使用夹具f的工序集合为$\mathcal{O}_f$, $\mathcal{O}_f=\{\cdots,O_{i,j},\cdots\}$, 例如, 若产品1的第一道工序和产品2的第一道工序都使用了夹具f, 那么$\mathcal{O}_f=\{O_{1,1},O_{2,1}\}$. 从MA段可获取$\mathcal{O}_f$中各工序对应的加工设备集合为$M_{\mathcal{O}_f}$, $M_{\mathcal{O}_f}=\{M_1,M_2,\cdots,M_{n_f}\}$. 在$\mathcal{O}_f$中, 将可选择相同加工设备(common optional machine, COM)的工序集合记作$\mathcal{O}_{\mathrm{COM}}$, $\mathcal{O}_{\mathrm{COM}}=\{\cdots,O_{k,l},\cdots\}$, 例如, 若产品1的第二、三、四道工序都可以选用设备m, 那么$\mathcal{O}_{\mathrm{COM}}=\{O_{1,2},O_{1,3},O_{1,4}\}$. 将可用夹具仅有夹具$f$的工序集合记作$\mathcal{O}_{f\mathrm{Only}}$, $\mathcal{O}_{f\mathrm{Only}}=\{\cdots,O_{m,n},\cdots\}$, 例如, 若产品1的第二道工序和产品2的第一道工序除了夹具f外没有其他可选夹具, 那么$\mathcal{O}_{f\mathrm{Only}}=\{O_{1,2},O_{2,1}\}$. $\mathcal{O}_{\mathrm{COM}}$和$\mathcal{O}_{f\mathrm{Only}}$的严格定义如下: $\forall O_{i,j},\ O_{k,l}\in\mathcal{O}_{\mathrm{COM}}$, $O_{i,j}\neq O_{k,l}$, $M_{i,j}\cap M_{k,l}\neq\varnothing$; $\forall O_{i,j}\in\mathcal{O}_{f\mathrm{Only}}, F_{i,j}=\{f\}$. $\mathcal{O}_f$, $\mathcal{O}_{\mathrm{COM}}$, $\mathcal{O}_{f\mathrm{Only}}$和$M_{\mathcal{O}_f}$具备以下性质:

性质1　若$\left|M_{\mathcal{O}_f}\right|=1$, MA段和FI段染色体无需可行化处理;

性质2 $\mathcal{O}_{\mathrm{COM}}\subseteq\mathcal{O}_f$, $\mathcal{O}_{f\mathrm{Only}}\subseteq\mathcal{O}_f$;

性质3 若 $\left|\mathcal{O}_{f\mathrm{Only}}\right|>1$, $\mathcal{O}_{f\mathrm{Only}}\subseteq\mathcal{O}_{\mathrm{COM}}$;

性质4 若 $\mathcal{O}_{\mathrm{COM}}=\varnothing$, $\left|\mathcal{O}_{f\mathrm{Only}}\right|\leqslant 1$.

基于上述性质, 可行化修复策略如下.

步骤一: 对FI段中任意出现的夹具f, 判断是否有$\left|M_{\mathcal{O}_f}\right|=1$. 若是, 流程结束; 若不是, 转至步骤二.

步骤二: 判断是否有$\mathcal{O}_{COM}=\varnothing$. 若是, 转至步骤三; 若不是, 计算$\left|\mathcal{O}_{f\mathrm{Only}}\right|$: 若$\left|\mathcal{O}_{f\mathrm{Only}}\right|\geqslant 1$, 转至步骤四; 若$\left|\mathcal{O}_{f\mathrm{Only}}\right|=0$, 转至步骤五.

步骤三: 不改变$\mathcal{O}_f$中可选夹具最少的一道工序的MA和FI基因, 其他工序保留MA基因, 基于其他可选夹具更改FI基因.

步骤四: $\mathcal{O}_{f\mathrm{Only}}$内的工序保留原FI基因, MA基因选用共同可选加工设备$m_{f\mathrm{Only}}$; 若其他工序的可选加工设备集不含$m_{f\mathrm{Only}}$, 保留原MA基因, 基于其他可选夹具更换FI基因, 反之改为$m_{f\mathrm{Only}}$.

步骤五: $\mathcal{O}_{\mathrm{COM}}$内的工序保留原FI基因, MA基因选用共同可选加工设备m_{COM}; 若其他工序的可选加工设备集合不含m_{COM}, 保留原MA基因, 基于其他可选夹具更换FI基因, 反之改为m_{COM}.

基于上述可行化策略的三种不同情形的MA和FI染色体修复过程如图4所示.

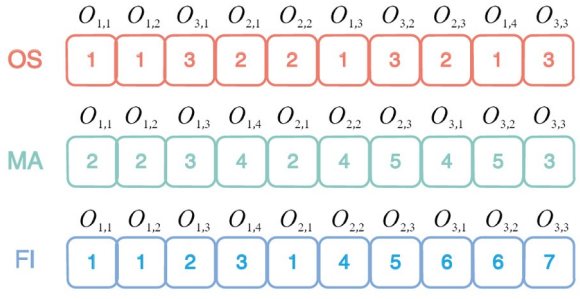

图 3　基于OS, MA和FI的三段式编码方案

Figure 3　Three-string chromosome strategy based on OS, MA, and FI.

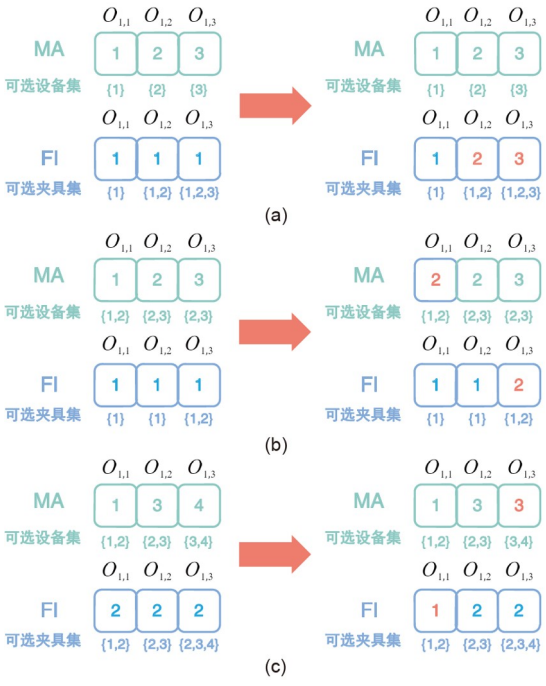

图 4　可行化修复策略示意图. (a)~(c)分别对应步骤三、四、五中的可行性修复方法

Figure 4　Schematic diagram of the feasibility correction strategy. (a)–(c) correspond to the feasibility correction methods in Steps 3, 4, and 5, respectively.

3.3　种群初始化

种群初始解的质量在很大程度上影响IGA-FCSSVNS的收敛速度和求解精度. 理想的初始解集既要保证大范围覆盖解空间, 尽可能保留向各个方向探寻优化的可能性; 又要包含部分质量良好、近似优解的初始解, 有潜质加快收敛速度.

因此, IGA-FCSSVNS采用随机生成和多种启发式规则相结合的初始化策略, 其中随机生成的解占初始种群的75%, 用于增强种群的多样性; 全局搜索规则[11]、Shortest Processing Time (SPT)规则[12]和Most Total Work Remaining (MTWR)规则[13]分别占初始种群的23%, 1%和1%, 为种群提供初始进化方向. 初始化的种群, 需进行MA段和FI段的可行性修复, 以确保夹具-托盘的绑定关系固定且不存在冲突.

3.4　迭代优化

IGA-FCSSVNS和多数元启发式算法的种群迭代优化思路相似, 主要包含了选择、交叉和变异等操作.

每一次迭代对当前种群进行适应度计算与评价, 适应度值即种群解码后的最大完工时间的倒数; 引入精英保留策略, 将当前种群的优秀个体存储到精英解池; 然后将当前种群作为父代, 采用轮盘赌策略[14]进行选择, 选择后的群体按照一定概率进行交叉、变异操作生成子代; 子代成为新的父代, 重复上述过程直至达到最大迭代次数.

OS段使用优先保持交叉算子(precedence preservative crossover)和多次交换变异算子(multi-times swap mutation), MA段使用多点交叉算子(multi-point crossover)和随机变异算子(random mutation), FI段使用顺序交叉算子(order crossover)和随机变异算子(random mutation)[15].

此外, 交叉概率P_c和变异概率P_m是影响算法寻求全局搜索和局部搜索平衡的关键因素. 理想的迭代过程在前期侧重全局搜索, P_c和P_m较大为宜; 后期侧重局部搜索, P_c和P_m较小为宜. 固定的交叉、变异概率很难兼顾迭代各时期的需求, 因此, 设置如下可伴随迭代过程动态调整的交叉、变异概率:

$$P_c = P_{c\,\mathrm{MAX}} - \frac{\mathrm{ITER}_C}{\mathrm{ITER}_T} \times \left(P_{c_{\mathrm{MAX}}} - P_{c_{\mathrm{MIN}}}\right), \tag{22}$$

$$P_m = P_{m_{\mathrm{MAX}}} - \frac{\mathrm{ITER}_C}{\mathrm{ITER}_T} \times \left(P_{m_{\mathrm{MAX}}} - P_{m_{\mathrm{MIN}}}\right), \tag{23}$$

其中, $P_{c_{\mathrm{MAX}}}$, $P_{m_{\mathrm{MAX}}}$代表迭代开始时的交叉、变异概率, $P_{c_{\mathrm{MIN}}}$, $P_{m_{\mathrm{MIN}}}$代表迭代结束时的交叉、变异概率, ITER_C代表当前迭代次数, ITER_T代表总迭代次数.

3.5　自学习型变邻域搜索

在种群迭代的中后期, 为进一步提升解的质量, 增强算法寻优能力, 引入了自学习型变邻域搜索(self-learning variable neighborhood search, SVNS)策略. 变邻域搜索(variable neighborhood search, VNS)作为一种有效的局部搜索方法[16], 可协助种群在趋于收敛的中后期实现解的再优化. SVNS在此基础上, 从精英解池中构建三种邻域结构的知识库, 通过不断地学习、反馈和更新, 伴随迭代进程自主地实现变邻域搜索策略优选, 从而实现对精英个体的优化提升.

3.5.1　邻域结构设计

在文献[17,18]的基础上, 设计了以下三种不同的邻域结构.

(1) VNS1: 针对MA和FI段的邻域结构

从MA段随机选取$k \in [1, |MA|]$个基因点位，在相应基因点位对应的加工工序的可选设备集中选择加工时间最短的设备进行替换; 之后, 进行MA段和FI段的可行性修复.

(2) VNS2: 针对OS段的邻域结构

从OS段随机选取$k \in [1, |OS|]$个基因点位, 将这k个基因点位对应的工序进行倒序排列.

(3) VNS3: 针对OS段的邻域结构

随机选择两件加工产品i和j, 将工序数少的产品的工序前置.

VNS1, VNS2, VNS3的示意如图5所示.

3.5.2　搜索策略知识库构建

在精英解池中提取过往VNS迭代过程中各精英解关于三种邻域结构(VNS1，VNS2和VNS3)的选择情况和优化效果，将其记录在成功知识矩阵和失败知识矩阵中，从而构建起搜索策略知识库，如图6所示. 其中，成功知识矩阵元素的一般形式为 $n_{ij}^{s}, i \in \{1,2,\cdots,m\}$, $j \in \{1,2,3\}$，表示精英解i在变邻域搜索迭代过程中选择邻域结构VNSj并成功优化当前最优解的次数; 失败知识矩阵元素的一般形式为$n_{ij}^{f}, i \in \{1,2,\cdots,m\}$, $j \in \{1,2,3\}$，表示精英解i在变邻域搜索迭代过程中选择邻域结构VNSj并未能优化当前最优解的次数. 如式(24)所示, 有

$$\forall i \in \{1,2,\cdots,m\}, \sum_{j=1}^{3} n_{ij}^{s} + \sum_{j=1}^{3} n_{ij}^{f} = \mathrm{ITER}_{\mathrm{VNS}}, \tag{24}$$

其中，$\mathrm{ITER}_{\mathrm{VNS}}$表示一次迭代过程进行变邻域搜索的次数.

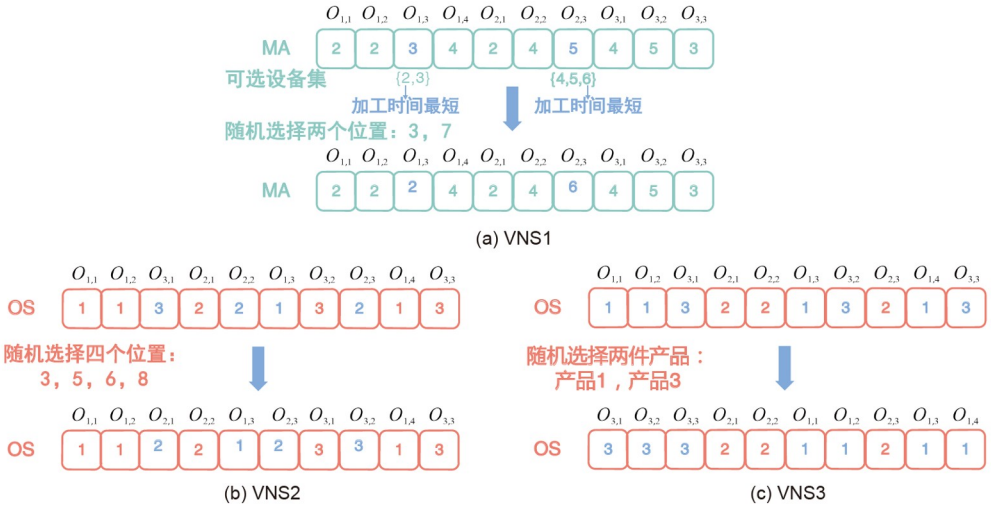

图 5　三种邻域结构示意图

Figure 5　Schematic diagrams of three VNS.

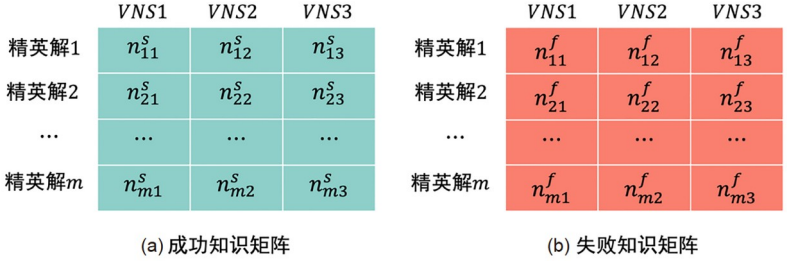

图 6　搜索策略知识库示意图

Figure 6　Schematic diagram of search strategy repository.

3.5.3 基于自学习型VNS策略优选的种群优化

在种群引入变邻域搜索策略的前期，采用随机选择邻域结构的方法进行变邻域迭代，并根据迭代效果填充搜索策略知识库. 到达一定次代后，基于从知识库提取出的统计概率知识，选择最有潜力优化种群的邻域结构进行变邻域搜索，同样在搜索过程中将搜索结果及时反馈给知识库，以保证动态更新的知识库始终提供最可靠的策略建议. 自学习型变邻域搜索的流程描述如下.

步骤一: 初始化参数. 初始解为当前次代i的最优解S_c，随机选择邻域结构的次代上限为I_{Random}，单次代的变邻域搜索迭代次数ITER_{VNS}，确定邻域结构$\text{VNS}j$，$j \in \{1,2,3\}$，成功知识矩阵和失败知识矩阵第i行的元素初始化为0.

步骤二: 判断是否满足当前次代i变邻域搜索的终止条件($i_{\text{VNS}} > \text{ITER}_{\text{VNS}}$)，若满足，输出更新后的最优解$S_c^*$并结束，若不满足转步骤三.

步骤三: 判断是否满足$i \leqslant I_{\text{Random}}$，若满足，当前次代$i$将随机选择邻域结构$\text{VNS}j$进行邻域搜索，否则将按式(25)~(27)所示基于知识库自学习地选择邻域结构.

步骤四: 选择邻域结构$\text{VNS}j$进行邻域搜索，得到新解S'.

步骤五: 若$\text{fitness}(S') < \text{fitness}(S_c^*)$，则$S_c^* = S'$，$n_{ij}^s = n_{ij}^s + 1$; 否则，$n_{ij}^f = n_{ij}^f + 1$.

步骤六: 基于更新后的知识库更新$\mathcal{P}_j, j \in \{1,2,3\}$，$i_{\text{VNS}} = i_{\text{VNS}} + 1$，转步骤二.

$$\text{VNS} = \text{Random}(VNS1, VNS2, VNS3 | \mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3), \tag{25}$$

$$P_j = \frac{\sum_{k=1}^{i} n_{kj}^s}{\sum_{k=1}^{i} n_{kj}^s + \sum_{k=1}^{i} n_{kj}^f}, j \in \{1,2,3\}, \tag{26}$$

$$\mathcal{P}_j = \frac{P_j}{\sum_{n=1}^{3} P_n}, j \in \{1,2,3\}. \tag{27}$$

基于自学习型变邻域策略优选的种群优化过程如图7所示. 随着搜索策略知识库的不断扩充和更新，种群选择最佳变邻域搜索策略的可能性也随之提高，从而提升算法整体的优化效果.

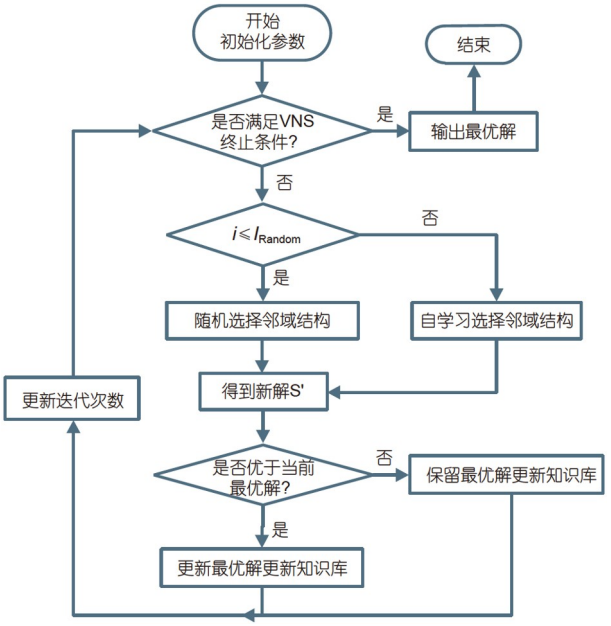

图 7 (网络版彩图)自学习型变邻域搜索算法框架

Figure 7 (Color online) Structure of self-learning VNS algorithm.

4 实验设计与结果分析

4.1 测试算例

为验证算法求解的可靠性和高效性，以国内某大型发动机制造企业新产品试制中心的真实生产数据为基础，生成了测试算例. 算例涵盖了新产品试制中心的15种产品的工艺数据: 产品的工序数量在5~10道之间不等，总工序数量为107道; 产品需要的加工设备共计25台，夹具共计61件. 依据涵盖的产品种类、产品数量、设备数量和夹具数量的不同，可将算例划分为小规模、中规模和大规模算例. 其中，小规模算例包含的产品种类不超过5种，产品数量不超过10件; 中规模算例包含的产品种类在5~10种之间，产品数量在30~40件之间; 大规模算例包含的产品种类在10~15种之间，产品数量在50~60件之间.

算例的命名格式为P-M-F，P表示总产品数量，M表示总设备数量，F表示总夹具数量. 同一规模的算例在总产品数量相等的前提下可能包含不同的产品组合方式，例如: 算例5-16-25表示产品总数为5件，且涵盖16台加工设备和25件夹具，在算例设计时既考虑了2件产品1、2件产品2和1件产品3的组合，也考虑了产品1~5各1件的组合.

4.2　参数设置

CPLEX求解器是学界和业界常用的高性能数学规划求解器, 因此选用CPLEX求解器求解本文建立的混合整数规划模型, 并将小规模、中规模和大规模算例的求解时间上限分别设置为3600, 14400, 14400 s. 若CPLEX求解器无法在限定时间内求出理论全局最优解, 则用当前求出的最优解作为求解结果.

在IGA-FCSSVNS中, 种群规模为300, 总迭代次数ITER_T为150, 迭代开始前的交叉概率$P_{c_{\mathrm{MAX}}}$为0.7, 变异概率$P_{m_{\mathrm{MAX}}}$为0.35, 迭代结束时的交叉概率$P_{c_{\mathrm{MIN}}}$为0.5, 变异概率$P_{m_{\mathrm{MIN}}}$为0.2. 在迭代次数为80时开始引入自学习型变邻域搜索策略, 随机选择邻域结构的次代上限I_{Random}为110, 单次代变邻域搜索的总次数$\mathrm{ITER}_{\mathrm{VNS}}$为30.

遗传算法是FJSP研究中最具代表性的元启发式算法之一, 因此将只考虑可行性修正但未引入变邻域搜索的遗传算法(genetic algorithm hybrid with feasibility correction strategy, GA-FCS)作为IGA-FCSSVNS的对比方法, 其参数设置与后者一致.

为衡量算法求解效果, 引入两个衡量指标EFF_1和EFF_2, 其定义如式(28)和(29)所示. 其中, $\overline{\mathrm{Opt}}_1$和$\overline{\mathrm{Opt}}_2$分别为IGA-FCSSVNS和GA-FCS相同算例进行5次重复实验后的平均最优解, Opt_3为设定时间范围内求解器求得的最优解. 若EFF_1和EFF_2为负, 证明ITGA-FCSSVNS相对于数学求解器和一般近似算法优化效果更好.

$$\mathrm{EFF}_1=\frac{\overline{\mathrm{Opt}}_1-\mathrm{Opt}_3}{\mathrm{Opt}_3}\times 100\%, \tag{28}$$

$$\mathrm{EFF}_2=\frac{\overline{\mathrm{Opt}}_1-\overline{\mathrm{Opt}}_2}{\overline{\mathrm{Opt}}_2}\times 100\%. \tag{29}$$

所有实验均在配置为AMD Ryzen 7 4800HS CPU@2.90 GHz+16 GB RAM的个人计算机上完成, 实验程序由Python3.8编写.

4.3　实验对比

不同规模下的算例测试结果如表3所示. 在小规模

表 3　算例测试结果统计

Table 3　Case study results

规模	*P-M-F*	编号	CPLEX		GA-FCS			IGA-FCSSVNS			EFF_1	EFF_2
			t_{CPU}/s	Opt	t_{CPU}/s	Opt	$\overline{\mathrm{Opt}}$	t_{CPU}/s	Opt	$\overline{\mathrm{Opt}}$		
小规模	5-16-25	1	0.28	218	195.50	219	219.00	191.25	218	218.00	0.00%	−0.46%
		2	1.59	234	137.00	236	237.50	124.00	234	234.00	0.00%	−1.47%
		3	18.17	284	144.00	289	293.20	148.00	285	287.20	1.13%	−2.05%
	10-16-25	1	7200.00	277	245.20	284	287.20	251.00	277	281.00	1.44%	−2.16%
		2	7200.00	302	251.20	318	322.40	251.00	303	304.20	0.73%	−5.65%
		3	7200.00	252	439.00	258	269.00	444.00	253	254.80	1.11%	−5.28%
中规模	30-20-52	1	14400.00	568	919.75	555	560.40	962.00	532	545.20	−4.01%	−2.71%
		2	14400.00	758	1065.40	586	599.00	1038.00	566	571.80	−24.56%	−4.54%
	35-16-25	1	14400.00	986	1658.00	843	852.00	1683.80	804	815.60	−17.30%	−4.27%
		2	14400.00	798	1700.00	818	836.40	1697.00	773	792.80	−0.65%	−5.21%
	40-20-52	1	14400.00	758	1461.00	739	763.40	1528.50	708	718.60	−5.20%	−5.87%
		2	14400.00	848	1543.25	753	774.00	1617.40	729	748.60	−11.72%	−3.28%
大规模	50-25-61	1	14400.00	690	1874.66	692	705.00	2037.00	649	677.60	−1.80%	−3.89%
		2	14400.00	833	1959.75	683	730.60	2127.00	669	685.40	−17.72%	−6.19%
	55-20-52	1	14400.00	1394	2426.66	1017	1047.40	2553.00	926	954.20	−31.55%	−8.90%
		2	14400.00	1231	2285.00	1003	1023.50	2474.32	956	1001.00	−18.68%	−2.20%
	60-25-61	1	14400.00	992	2546.40	815	837.00	2615.50	791	804.40	−18.91%	−3.89%
		2	14400.00	1319	2920.00	915	943.80	3030.50	869	892.50	−32.34%	−5.44%

算例中，CPLEX和IGA-FCSSVNS都可以在短时间内求出5-16-25系列中大部分算例的全局最优解，验证了模型的正确性，而GA-FCS只能得到与全局最优解相对偏差在2.05%以内的平均最优解；而对于10-16-25系列的算例, CPLEX未能在7200 s内求得全局最优解, 而IGA-FCSSVNS可在200~450 s时间范围内求得与CPLEX求解结果相差不超过1.44%的近似最优解，同样优于GA-FCS的求解效果. 而当算例规模进一步扩大后，IGA-FCSSVNS的优势开始凸显出来: 从30-20-52系列到60-25-61系列，CPLEX在14400 s内都未能求出全局最优解, 而IGA-FCSSVNS在所有算例上的求解结果均优于CPLEX，求解质量提升率在部分算例上甚至高达24.56%和32.34%(30-20-52系列的2号算例和60-25-61系列的2号算例)，求解时间虽然随着算例规模扩大也有所增加，但均远远优于CPLEX的14400 s. 相比之下，虽然GA-FCS在某些算例求得的平均最优解优于CPLEX，但也存在平均最优解逊于CPLEX的情况. 而IGA-FCSSVNS在每一个中、大规模算例上的求解效果均优于GA-FCS，自学习型变邻域搜索策略的优化效果可见一斑.

为进一步对比IGA-FCSSVNS，GA-FCS和CPLEX的实验结果，选取各规模下代表算例的一次迭代过程绘制曲线图, 如图8所示. 与表3的数据一致, 在小规模算例(除10-16-25系列外)中，IGA-FCSSVNS与CPLEX均可求出全局最优解；在中、大规模算例中，ITGA-FCSSVNS均可得到优于甚至远胜于CPLEX的求解结果. 观察可发现，在引入自学习型变邻域搜索策略(第80代)后, IGA-FCSSVNS的曲线相对于GA-FCS呈现出

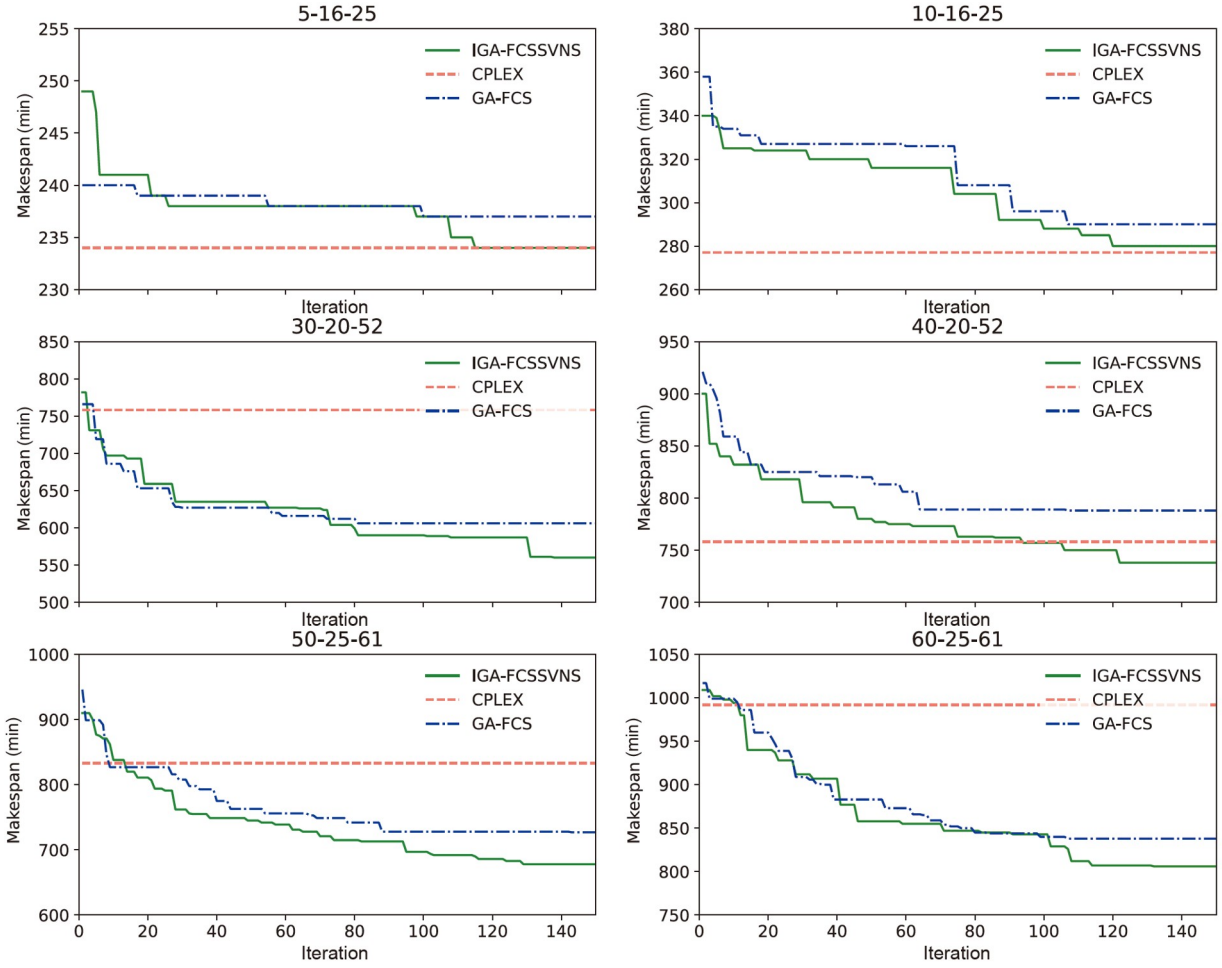

图 8 代表算例迭代曲线图

Figure 8 Iteration curves of representative cases.

更为明显的下降趋势, 并最终获得更好的最优解.

总的来说, 在考虑夹具-托盘组合优化的三资源约束复杂情景下, 随着产品和资源规模的扩大, 基于MRFJSP-FPCO建立的MILP模型无法在理想时间内通过CPLEX求解器求出高质量的解; 而IGA-FCSSVNS凭借其始终保持可行性的修复策略和自学习的变邻域搜索策略, 在相对较短的时间内可求出优于CPLEX和GA-FCS的解.

5　结语

基于考虑夹具-托盘-设备三资源约束的柔性作业车间调度场景, 本文建立了MRFJSP-FPCO的MILP模型, 并提出了一种基于可行性修复与自学习型变邻域搜索的改良遗传算法(IGA-FCSSVNS): 设计了基于工序排序、设备选择、夹具选择的三段式编码方案, 并针对性地引入面向MA与FI染色体的可行性修复策略以保证整个迭代过程中种群个体的有效性, 通过建立VNS策略集和搜索策略知识库, 在迭代中后期实现自学习型变邻域搜索. 经过源于真实生产情景的各规模算例求解, ITGA-FCSSVNS的有效性和优越性得以验证.

IGA-FCSSVNS不同于传统的作业车间调度, 在为每件产品每道工序最优地安排适当的加工设备和先后顺序之外, 同样实现了夹具-托盘这一辅助资源的优化组合和工序选择, 紧贴车间生产实际需求, 在大力倡导制造业智能化转型的今天具有极高的应用潜力. 未来, 如何针对车间异常状况做出及时合理的动态响应值得进一步研究.

参考文献

1 Li X Y, Li Z F, Gao L. Paths for the digital transformation and intelligent upgrade of China's discrete manufacturing industry (in Chinese). China Eng Sci, 2022, 24: 64–74 [李新宇, 李昭甫, 高亮. 离散制造行业数字化转型与智能化升级路径研究. 中国工程科学, 2022, 24: 64–74]

2 Zhang J, Qin W. Intelligent manufacturing scheduling first—A guide of *Manufacturing System Intelligent Scheduling Method and Cloud Service* (in Chinese). China Mech Eng, 2019, 30: 1002–1007 [张洁, 秦威. 智能制造 调度为先——《制造系统智能调度方法与云服务》导读. 中国机械工程, 2019, 30: 1002–1007]

3 Zhang J, Ding G, Zou Y, et al. Review of job shop scheduling research and its new perspectives under Industry 4.0. J Intell Manuf, 2019, 30: 1809–1830

4 Huang X W, Chen S F, Zhou T Y, et al. Survey on genetic algorithms for solving flexible job-shop scheduling problem (in Chinese). Comput Integr Manuf Syst, 2022, 28: 536–551 [黄学文, 陈绍芬, 周阗玉, 等. 求解柔性作业车间调度的遗传算法综述. 计算机集成制造系统, 2022, 28: 536–551]

5 Wang S H, Li Y, Li X Y. An improved whale swarm algorithm for flexible job-shop scheduling problem (in Chinese). J Chongqing Univ, 2020, 43: 1–11 [王思涵, 黎阳, 李新宇. 基于鲸鱼群算法的柔性作业车间调度方法. 重庆大学学报, 2020, 43: 1–11]

6 Lei D, Guo X. Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. Int J Product Res, 2014, 52: 2519–2529

7 Liu L, Song H C, Jiang T H, et al. Modified biology migration algorithm for dual-resource energy-saving flexible job shop scheduling problem (in Chinese). Comput Integr Manuf Syst, 2022, 28: 1–24 [刘璐, 宋海草, 姜天华, 等. 基于改进生物迁徙算法的双资源柔性作业车间节能调度问题. 计算机集成制造系统, 2022, 28: 1–24]

8 Li H. Research on tool remaining useful life prediction considering data completeness and job-shop scheduling problem with tool life constraint (in Chinese). Dissertation for Doctoral Degree. Chengdu: University of Electronic Science and Technology of China, 2022. 1–13 [李海. 考虑数据完全性的刀具剩余寿命预测及其约束下的作业车间调度研究. 博士学位论文. 成都: 电子科技大学, 2021. 1–13]

9 Wu X, Peng J, Xiao X, et al. An effective approach for the dual-resource flexible job shop scheduling problem considering loading and unloading. J Intell Manuf, 2021, 32: 707–728

10 Demir Y, Kürşat İşleyen S. Evaluation of mathematical models for flexible job-shop scheduling problems. Appl Math Model, 2013, 37: 977–988

11 Zhang G H, Gao L, Li P G, et al. Improved genetic algorithm for the flexible job-shop scheduling problem (in Chinese). J Mech Eng, 2009, 45: 145–151 [张国辉, 高亮, 李培根, 等. 改进遗传算法求解柔性作业车间调度问题. 机械工程学报, 2009, 45: 145–151]

12 Sousa R A, Varela M L R, Alves C, et al. Job shop schedules analysis in the context of industry 4.0. In: Proceedings of International Conference on Engineering, Technology and Innovation (ICE/ITMC). IEEE, 2017. 711–717

13 Kaban A K, Othman Z, Rohmah D S. Comparison of dispatching rules in job-shop scheduling problem using simulation: A case study. Int J

Simul Model, 2012, 11: 129–140

14 Zhang C, Zhan Z H. Comparisons of selection strategy in genetic algorithm (in Chinese). Comput Eng Design, 2009, 30: 5471–5474, 5478 [张琛, 詹志辉. 遗传算法选择策略比较. 计算机工程与设计, 2009, 30: 5471–5474, 5478]

15 Qu X H, Ji F, Meng G J, et al. Green scheduling of flexible job-shop based on hyper heuristic genetic algorithm (in Chinese). J Mech Electr Eng, 2022, 39: 255–261 [屈新怀, 纪飞, 孟冠军, 等. 超启发式遗传算法柔性作业车间绿色调度问题研究. 机电工程, 2022, 39: 255–261]

16 Zhang G, Zhang L, Song X, et al. A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem. Cluster Comput, 2019, 22: 11561–11572

17 Zhao S K. Hybrid algorithm based on improved neighborhood structure for flexible job shop scheduling (in Chinese). Comput Integr Manuf Syst, 2018, 24: 3060–3072 [赵诗奎. 柔性作业车间调度的改进邻域结构混合算法. 计算机集成制造系统, 2018, 24: 3060–3072]

18 Gu X L. Application research of flexible job-shop scheduling problem based on improved genetic algorithm (in Chinese). Dissertation for Doctoral Degree. Dalian: Dalian Jiaotong University, 2020. 35–37 [谷晓琳. 基于改进遗传算法的柔性作业车间调度问题的应用研究. 博士学位论文. 大连: 大连交通大学, 2020. 35–37]

Multiresource constrained flexible job shop intelligent scheduling considering fixture-pallet combinatorial optimization

LIU MoLin[1], ZHOU YuLu[1], WANG SiYang[2], ZHANG ChunMing[2], DU ShiChang[1] & XI LiFeng[1]

[1] *Department of Industrial Engineering and Management, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China;*
[2] *Weichai Holding Group Co., Ltd., Weifang 261061, China*

Research on multiresource flexible job shop intelligent scheduling that considers fixture-pallet combination optimization is proposed to address the productivity restriction problem caused because of limited fixture-pallet resources in multiproduct mixed manufacturing workshops. First, a mixed integer linear programming model with machine-fixture-pallet constraints is created with the goal of minimizing the makespan. Then, an improved genetic algorithm hybrid with a feasibility correction strategy and self-learning variable neighborhood search (VNS) is proposed to solve the problem. The feasibility correction strategy is designed to solve the potential conflicts of chromosome structure, and the VNS repository is constructed in the semilate period of iterations, which helps in finding a better solution. The effectiveness and efficiency of the proposed algorithm are demonstrated using computational experiments based on industrial big data, which would provide convincing intelligent support for actual production scheduling with constrained resources.

production scheduling, flexible job shop, fixture-pallet constraint, feasibility correction strategy, self-learning VNS

doi: 10.1360/SST-2022-0334