叶孝生, 张舒曼, 唐天誉,等. 自适应遗传算法与邻域搜索的柔性作业车间调度方法[J]. 智能计算机与应用,2026,16(7):205-211. DOI:10.20169/j.issn.2095-2163.24103103

# 自适应遗传算法与邻域搜索的柔性作业车间调度方法

叶孝生, 张舒曼, 唐天誉, 闫金辉, 刘子珺, 叶春明
(上海理工大学 管理学院, 上海 200093)

**摘 要**: 本文针对柔性作业车间调度问题(Flexible Job Shop Scheduling Problem , FJSP),提出了一种融合自适应遗传算法和邻域搜索的优化策略,旨在最小化最大完工时间。首先,在遗传算法框架下,利用自适应种群初始化策略,动态调整种群结构,以提升算法的全局搜索能力。其次,引入多种邻域结构,结合局部搜索策略,通过邻域搜索算法对解空间进行深入探索,增强算法的局部搜索性能。本文设计了一种基于交叉和变异操作的改进机制,旨在增强种群的丰富性和提升算法的收敛速度。通过 FJSP 的 30 个标准算例验证了该算法在解决 FJSP 中的有效性。测试结果表明:本文算法在 23 个标准算例中的表现均优于或持平于其他智能优化算法,表现出了较好的寻优能力,且通过多次重复实验也显现出了很强的稳定性。

**关键词**: 柔性作业车间调度; 遗传算法; 邻域搜索; 最大完工时间; 优化策略
**中图分类号**: TH165;TP18 **文献标志码**: A **文章编号**: 2095-2163(2026)07-0205-07

# Flexible job shop scheduling method based on adaptive genetic algorithm and neighborhood search

YE Xiaosheng, ZHANG Shuman, TANG Tianyu, YAN Jinhui, LIU Zijun, YE Chunming
(Business School, University of Shanghai for Science and Technology, Shanghai 200093, China)

**Abstract**: This paper proposes an optimization strategy that combines adaptive genetic algorithm and neighborhood search for the Flexible Job Shop Scheduling Problem (FJSP) to minimize the maximum completion time. Firstly, in the framework of genetic algorithm, an adaptive population initialization strategy is used to dynamically adjust the population structure to enhance the algorithm´s global search ability. Secondly, various neighborhood structures are introduced, combined with local search strategies, to explore the solution space deeply through neighborhood search algorithms to enhance the algorithm´s local search performance. This paper designs an improved mechanism based on crossover and mutation operations to enhance the diversity of the population and accelerate the convergence of the algorithm. The effectiveness of the algorithm in solving the FJSP is verified by thirty standard examples of FJSP. The test results indicate that the performance of the algorithm proposed in this paper is superior to or on par with other intelligent optimization algorithms in twenty-three standard examples, showing better optimization ability. Moreover, it also exhibits strong stability through multiple repeated experiments.

**Key words**: flexible job shop scheduling; genetic algorithm; neighborhood search; maximum completion time; optimization strategy

## 0 引 言

柔性作业车间调度问题是一类典型的柔性生产制造问题,是比传统作业车间调度问题(Job Shop Scheduling Problem, JSP)更复杂的 NP-hard 问题[1]。在 FJSP 中,工件不再局限于特定的机器,而是可以在多台机器之间进行调度,每台机器的加工时间也各不相同。这种灵活性使得调度问题的复杂性大大增加,尽管会更贴近实际生产需求,因此相比 JSP,FJSP 需要采用更为复杂的优化方法来应对[2]。针对 FJSP,国内外学者提出了不同的研究方法包括粒子群优化(PSO)算法[3]、蚁群优化(ACO)算法[4]、灰狼优化(GWO)算法[5]等,这些算法在不同规模的 FJSP 问题中表现出色。吴晓雯等学者[6]将改进的 PSO 算法应用在 8×8 的作业车间调度问题中,效果优异。赵小惠等学者[7]将改进蚁群算法应

**基金项目**: 上海市哲学社会科学一般项目(2022BGL010)。
**作者简介**: 叶孝生(2000—),男,硕士研究生。主要研究方向:生产调度优化。
**通信作者**: 叶春明(1964—),男,博士,教授。主要研究方向:生产调度,智能算法,工业工程。Email: yechunming6464@163.com。
**收稿日期**: 2024-10-31

用于规模为 4×6 的部分柔性作业车间调度小规模算例中取得了良好结果。马随东等学者[8]使用改进的灰狼优化算法在 Hurink 测试实例上进行了实验,算法在 30 个测试实例中获得了最优解,显示了该算法在中到大规模问题中的优越性。

遗传算法因其全局搜索能力强而广泛应用于调度优化问题[9],却容易陷入局部最优。随着问题规模和复杂性增加,传统算法的求解效率低,近些年来一些学者通过融合遗传算法与其他算法提升性能。常镜洳等学者[10]通过增强学习动态调整交叉和变异概率,提升了遗传算法在求解多目标 FJSP 中的效率和解的质量。陆科苗等学者[11]提出一种改进的非支配排序遗传算法(INSGA-Ⅱ)与变邻域搜索算法(VNS)求解多目标柔性作业车间调度问题,效果显著。张淑丽等学者[12]提出岛屿模型遗传算法,有效克服了遗传算法耗时较多和早熟的弱点。陈祉烨等学者[13]提出强化学习的改进遗传算法,使得算法收敛速度得到提高、求解速度更快的同时能够跳出局部最优。李雪花等学者[14]运用混合蛙跳算法提高了局部搜索能力来求解单目标柔性车间调度问题。李浩平等学者[15]为提高收敛速度,提出了改进灰狼优化算法,通过引入工件编码迭代、自适应算子和 POX 交叉,针对钢琴制造企业实木车间数据验证,结果表明其在调度问题中的效率和精度优于遗传算法。本文提出了一种自适应遗传算法,通过动态调整种群初始化策略和交叉变异操作,提高了种群多样性和全局搜索能力。同时引入多种邻域结构,结合局部搜索策略,以此显著增强了算法的搜索效能和解的质量。

## 1 FJSP 概述

### 1.1 问题描述

FJSP 最早由 Brucker 等学者[16]在 1990 年提出。该问题因其灵活性和复杂性,更加贴合实际生产环境的需求。在 FJSP 中,假设有 $n$ 个工件需要通过 $m$ 部机器来完成,每个工件包含多道工序,这些工序按照一定的顺序执行。每道工序可以在不同机器上处理,具有较强的灵活性,而每台机器处理同一作业的时间可能各不相同。因此,FJSP 主要涉及以下 2 个核心子问题:确定工件的加工顺序;合理分配机器以优化加工过程。

解决 FJSP 的核心是优化每道工序的机器分配及其加工顺序,旨在最小化某一或多个优化目标。本文聚焦于最小化最大完工时间这一目标,在模型建立之前,作出以下假设:

(1)各机器之间是相互独立的;

(2)在初始时刻,所有机器和工件均处于可加工状态;

(3)对于同一工件,各工序之间遵循先后顺序,而不同工件则互不干扰;

(4)在任意时间,单台机器仅能加工一个工件;

(5)为简化问题,忽略机器的设置时间和工件在不同机器间的运输时间;

(6)所有加工过程均为不可中断。

### 1.2 相关符号说明

本文所使用参数符号及定义见表 1。

**表 1 模型参数符号和定义**

**Table 1 Parameter symbols and definitions of the model**

| 符号 | 定义 |
|---|---|
| $I$ | 所有的工件集合 $I=\{i_1,i_2,\cdots,i_n\}$ |
| $M$ | 可用加工机器集合 $M=\{M_1,M_2,\cdots,M_m\}$ |
| $O_i$ | 工件 $i$ 工序集合 $O_i=\{o_{i1},o_{i2},\cdots,o_{iq}\}$ |
| $o_{ij}$ | 工件 $i$ 的第 $j$ 道工序 |
| $M_{ij}$ | 工序 $o_{ij}$ 的机器集合 $M_{ij}=\{M_1,M_2,\cdots,M_k\}$ |
| $s_{ijk}$ | 工序 $o_{ij}$ 在机器 $M_k$ 上的开工时间 |
| $e_{ijk}$ | 工序 $o_{ij}$ 在机器 $M_k$ 上的完工时间 |
| $t_{ijk}$ | 工序 $o_{ij}$ 在机器 $M_k$ 上的处理时间 |
| $c_i$ | 工件 $i$ 的完工时间, $1\leqslant i\leqslant n$ |
| $c_{ij}$ | 工序 $o_{ij}$ 的完工时间 |
| $x_{ijk}$ | 决策变量,若工序 $o_{ij}$ 在机器 $M_k$ 上加工则为 1,反之则为 0 |

### 1.3 模型建立

根据问题描述, FJSP 的数学模型可以建立如下:

目标是最小化最大完工时间,数学公式如下:

$$f=\min(\max c_i) \tag{1}$$

其中, $c_i$ 表示工件 $i$ 的完工时间。约束条件如下:

(1)每道工序只能在一台机器上进行加工。公式如下

$$\sum_{k=1}^{m} x_{ijk}=1 \tag{2}$$

其中, $x_{ijk}$ 表示一个二进制变量, 当工件 $i$ 的第 $j$ 道工序在机器 $M_k$ 上加工时, $x_{ijk}=1$, 否则 $x_{ijk}=0$。

(2)每道工序的完成时间点不应早于其开始时间与所需加工时间之和,这一关系可以表示为:

$$c_{ij}=s_{ij}+\sum_{k=1}^{m} x_{ijk}\cdot t_{ijk} \tag{3}$$

其中, $s_{ij}$ 表示工件 $i$ 的第 $j$ 道工序的开始时间, $t_{ijk}$ 表示在机器 $M_k$ 上的加工时间。

(3)同一工件的相邻工序必须按照工序的优先顺序依次加工,具体公式如下:

$$s_{i(j+1)} \geqslant e_{ij} \tag{4}$$

(4)在同一台机器上,工序的加工需遵循顺序原则,前一道工序必须在后一工序开始前完成,数学公式为:

$$s_{ijk} \geqslant e_{hgk} \tag{5}$$

其中, $e_{hgk}$ 表示工件 $h$ 的第 $g$ 道工序,在机器$M_k$上的加工顺序在工序$o_{ij}$之前。

## 2　自适应遗传算法与邻域搜索的优化策略

### 2.1　编码与解码方法

FJSP 作为一个离散优化问题,涉及 2 个子问题:工序排序与机器分配。针对这一问题,采用两段式的整数编码,根据机器分配和工序排序两个维度设计。这 2 部分的编码长度均与总工序数量保持一致,其中第 1 段编码代表机器分配方案,而第 2 段编码则对应工序的加工顺序[17]。参考文献[17]中随机产生的一个调度解的结构,其调度方案编码如图 1 所示。图 1 中,MS 表示基于机器的编码,OS 表示基于工序的编码。MS 部分中的元素表示在该工序可选机器集中的机器顺序号,即 $o_{11}$ 使用可选机器集中的第 4 台机器, $o_{12}$ 使用可选机器集中的第 1 台机器,以此类推;OS 部分中的元素表示工件号,工件号从左到右统计出现几次就代表是该工件的第几道工序。

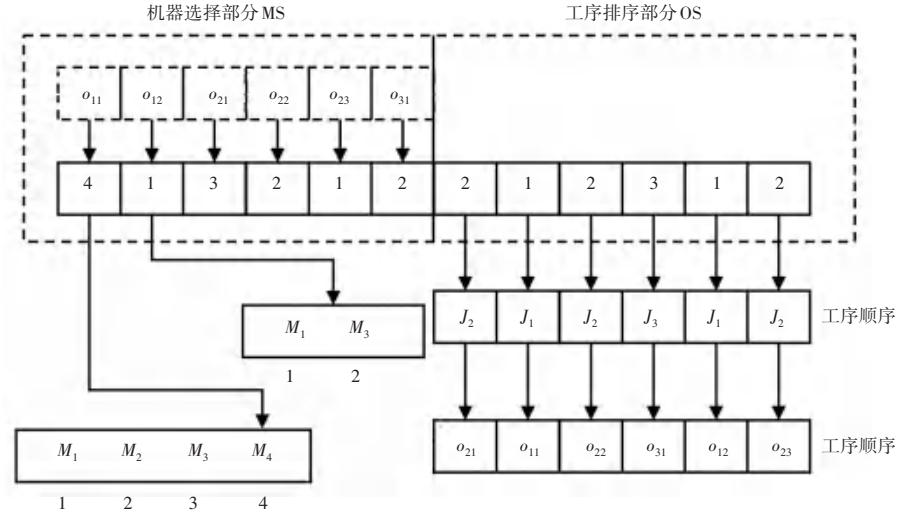



**图 1　调度方案编码**

**Fig. 1　Scheduling scheme encoding**

### 2.2　种群初始化

OS 编码根据各工件的操作数生成初始序列,每个元素代表一个操作,序列顺序反映操作的执行顺序。为增加种群多样性,随机打乱序列,确保初始种群结构多样。MS 编码用于为每个工序选择合适的加工机器,结合全局选择、邻域选择和随机选择三种策略。其中,全局选择优先选出加工效率高的机器,提升解质量;邻域选择深入局部区域以探索更优解;随机选择保持种群多样性,避免解集中。通过自适应调整 3 种策略权重,MS 编码为遗传算法提供多样化和优质的初始解,提升全局搜索和局部优化能力。

### 2.3　选择操作

在求解 FJSP 的遗传算法中,选择操作用于决定哪些个体将被保留到下一代,本文采用了锦标赛选择、随机锦标赛选择、基于记忆的锦标赛选择三种选择策略。

### 2.4　交叉操作

本文针对不同编码类型(OS 和 MS)采用了多种交叉策略,包括均匀交叉、部分映射交叉(POX)和随机交叉,以增强种群多样性和全局搜索能力。对于 MS 编码,表示每个操作的机器选择,采用均匀交叉策略:随机决定每个基因来自哪一个父代,从而确保机器选择的合理性,并保持父代优良特性,同时引入新的基因组合,增加种群多样性。对于 OS 编码,表示工件操作顺序,采用部分映射交叉和随机交叉两种策略。其中,部分映射交叉首先选取 2 个交叉点,复制父代部分序列到子代的对应位置,保持操作顺序的合理性。随机交叉通过随机选择父代基因,增加了更大的随机性,提升种群多样性,但可能影响解的质量。交叉操作结合自适应策略,动态调整交叉概率,平衡全局搜索与局部开发,提升全局最优解的寻优效率。

### 2.5 变异操作

变异操作通过随机改变个体基因,探索新的解空间,从而增强算法的搜索能力。针对 OS 编码和 MS 编码,本文设计了最优引导变异和随机变异两种策略,以提升全局搜索能力和局部搜索精度。对于 MS 编码,表示每个操作的机器选择,变异操作旨在调整机器分配。最优引导变异策略基于当前最优解进行变异,随机选择一个基因位置,并将其变为与最优解相似的值,从而探索最优解邻域内的其他可能性,促进全局收敛。随机变异则随机选择一个或多个基因位置,并将其更改为其他可行的机器选择。对于 OS 编码,表示操作顺序,变异操作通过改变操作顺序生成新解。主要采用随机变异策略,即随机选择一个或多个基因位置并交换或重新排列,从而产生多样化解,帮助跳出局部最优解,进行更广泛的解空间探索。

### 2.6 变邻域搜索算法

本文引入了 3 种局部搜索策略:单机器变邻域搜索、双机器变邻域搜索和快速局部搜索,以增强全局搜索能力并避免陷入局部最优。单机器变邻域搜索通过随机选择工件及操作,将其分配至不同机器,并选择最优分配方案来提升局部优化效果。双机器变邻域搜索在单机器优化的基础上,增加了工件操作顺序的交换,进一步优化调度,适用于更复杂的调度环境。快速局部搜索结合了前 2 种方法的优点,通过快速迭代同时优化操作顺序和机器选择,生成多个优质邻域解,增加搜索深度,提升局部搜索效率。

### 2.7 改进的遗传算法执行流程

本文提出了一种改进的遗传算法旨在最小化 FJSP 的完工时间。算法流程如图 2 所示,步骤如下:

**步骤 1** 初始化算法参数,设定种群大小、交叉及变异概率、最大迭代次数等。

**步骤 2** 混合策略初始化种群。OS 编码通过随机排列生成,MS 编码采用全局选择、局部选择和随机选择生成,确保种群多样性。

**步骤 3** 计算适应度值并选取精英个体,解码种群中的个体,计算完工时间,并根据适应度选出精英个体。

**步骤 4** 检查终止条件。若精英个体的适应度达到要求,输出最优解并终止算法;否则进入步骤 5。

**步骤 5** 选择操作。依据适应度采用锦标赛策略选取父代个体,包括锦标赛选择、记忆库锦标赛选择和随机锦标赛选择等。

**步骤 6** 交叉操作。对父代个体进行均匀交叉和 POX 交叉,每对父代按交叉概率多次交叉,选取最优个体进入子代种群。

**步骤 7** 变异操作。对交叉后产生的个体,进行随机变异和最优引导变异操作,若变异后的个体适应度更高,则替换当前个体。

**步骤 8** 局部搜索优化。对变异个体进行局部搜索,采用单机器变邻域搜索、双机器变邻域搜索和快速局部搜索方法优化个体解的质量。

**步骤 9** 生成新一代种群。将通过选择、交叉、变异和局部搜索后的个体组成新一代种群。

**步骤 10** 转到步骤 3。重复上述步骤,直至满足终止条件。

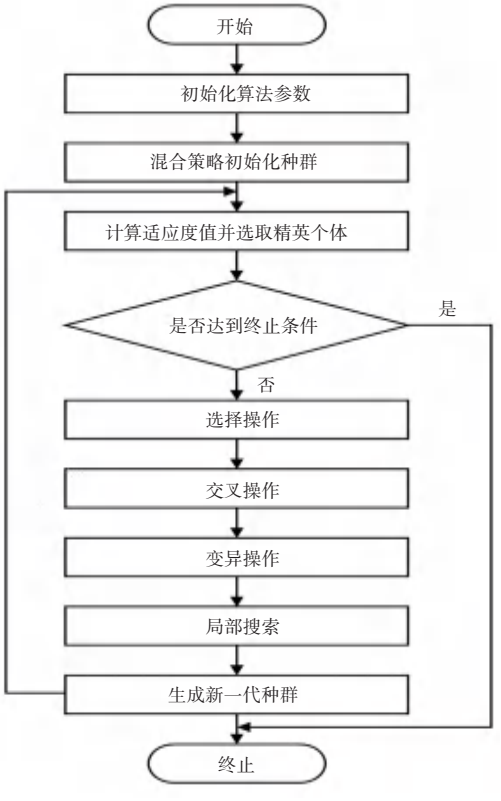



**图 2 改进遗传算法求解 FJSP 流程图**

**Fig. 2 Flowchart of improving genetic algorithm for solving FJSP**

## 3 算例分析

本文使用 Python3.11.8 软件作为编程工具,对所提出的算法进行仿真实验。编程环境如下:CPU 为 Inteli5,内存容量为 16 GB,操作系统为 Windows10。设定参数如下:种群规模为 800,最大迭代次数为 200,锦标赛规模为 4,记忆大小占比为 40%,交叉与变异操作的概率分别为 50%,VNS 比例为 0.01。

为验证本文的算法应用至 FJSP 的有效性，基于 10 个 Brandimarte 算例[18]进行测试，算例属于中小规模的 FJSP。为了增加实验的广泛性，还引入了 20 个 Fattahi 算例，该算例属于中大规模的 FJSP。为保证测试结果不失一般性，对不同算例独立运行 10 次后通过与其他文献提出的算法进行对比，对比结果见表 2 和表 3。表 2、表 3 中，使用加粗字体表示对应算例的最佳求解结果；$n \times m$ 表示算例的工件数和加工机器数；RPD 表示平均相对百分比偏差，可用如下公式来计算：

$$\mathrm{RPD} = \left(\frac{(\mathrm{Best} - \mathrm{Min})}{\mathrm{Min}}\right) \times 100\% \tag{6}$$

其中，Best 表示对应算法获得的最佳数值；Min 表示对于每一个算例，所有算法的最优解。

**表 2　Brandimarte 算例计算结果对比**

**Table 2　Comparison of computational results for Brandimarte instances**

| 算例 | $n \times m$ | HGWO[19] | | IALO[20] | | IMMBBO[21] | | GATS+HM[22] | | NIMASS[23] | | 本文算法 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Best | RPD/% | Best | RPD/% | Best | RPD/% | Best | RPD/% | Best | RPD/% | Best | 平均值 | RPD/% |
| Mk1 | 10×6 | **40** | 0 | **40** | 0 | **40** | 0 | 42 | 5.000 | **40** | 0 | **40** | 41.53 | 0 |
| Mk2 | 10×6 | 29 | 7.410 | 28 | 3.700 | 28 | 3.700 | 28 | 3.700 | 28 | 3.700 | 27 | 28.00 | 0 |
| Mk3 | 15×8 | **204** | 0 | **204** | 0 | **204** | 0 | **204** | 0 | **204** | 0 | **204** | 204.00 | 0 |
| Mk4 | 15×8 | 65 | 6.560 | 67 | 9.840 | 66 | 8.200 | 75 | 22.950 | **64** | 4.920 | **61** | 63.60 | 0 |
| Mk5 | 15×4 | 175 | 1.160 | 174 | 0.580 | **173** | 0 | 179 | 3.470 | **173** | 0 | 174 | 178.30 | 0.580 |
| Mk6 | 10×15 | 79 | 27.200 | 73 | 17.740 | 72 | 16.130 | 69 | 11.300 | 66 | 6.450 | **62** | 65.10 | 0 |
| Mk7 | 20×5 | 149 | 3.470 | 149 | 3.470 | **144** | 0 | 149 | 3.470 | **144** | 0 | 145 | 145.500 | 0.690 |
| Mk8 | 20×10 | **523** | 0 | **523** | 0 | **523** | 0 | 555 | 6.120 | 523 | 0 | **523** | 523.00 | 0 |
| Mk9 | 20×10 | 325 | 5.860 | 323 | 5.210 | 325 | 5.860 | 342 | 11.400 | 310 | 0.980 | **307** | 310.23 | 0 |
| Mk10 | 20×15 | 253 | 16.590 | 234 | 7.830 | 257 | 18.430 | 242 | 11.520 | 230 | 5.990 | **217** | 226.63 | 0 |
| Mean | | | 6.825 | | 4.837 | | 5.232 | | 7.893 | | 2.204 | | | 0.127 |

**表 3　Fattahi 算例计算结果对比**

**Table 3　Comparison of computational results for Fattahi instances**

| 算例 | $n \times m$ | AIA[24] | | HHS[25] | | HBWO[26] | | 本文算法 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Best | RPD/% | Best | RPD/% | Best | RPD/% | Best | 平均值 | RPD/% |
| SFJS1 | 2×2 | **66** | 0 | **66** | 0 | **66** | 0 | **66** | 66.0 | 0 |
| SFJS2 | 2×2 | **107** | 0 | **107** | 0 | **107** | 0 | **107** | 107.0 | 0 |
| SFJS3 | 3×2 | **221** | 0 | **221** | 0 | **221** | 0 | **221** | 221.0 | 0 |
| SFJS4 | 3×2 | **355** | 0 | **355** | 0 | **355** | 0 | **355** | 355.0 | 0 |
| SFJS5 | 3×2 | **119** | 0 | **119** | 0 | **119** | 0 | **119** | 119.0 | 0 |
| SFJS6 | 3×3 | **320** | 0 | **320** | 0 | **320** | 0 | **320** | 320.0 | 0 |
| SFJS7 | 3×5 | **397** | 0 | **397** | 0 | **397** | 0 | **397** | 397.0 | 0 |
| SFJS8 | 3×4 | **253** | 0 | **253** | 0 | **253** | 0 | **253** | 253.0 | 0 |
| SFJS9 | 3×3 | **210** | 0 | **210** | 0 | **210** | 0 | **210** | 210.0 | 0 |
| SFJS10 | 4×5 | **516** | 0 | **516** | 0 | **516** | 0 | **516** | 516.0 | 0 |
| MFJS1 | 5×7 | **468** | 0 | **468** | 0 | **468** | 0 | **468** | 468.7 | 0 |
| MFJS2 | 5×7 | 448 | 0.448 | **446** | 0 | **446** | 0 | **446** | 453.8 | 0 |
| MFJS3 | 6×7 | 468 | 0.429 | **466** | 0 | **466** | 0 | **466** | 476.5 | 0 |
| MFJS4 | 7×7 | **554** | 0 | **554** | 0 | **554** | 0 | **554** | 572.0 | 0 |
| MFJS5 | 7×7 | 527 | 2.529 | **514** | 0 | 539 | 4.864 | 527 | 547.0 | 2.529 |
| MFJS6 | 8×7 | 635 | 0.158 | **634** | 0 | **634** | 0 | 635 | 645.0 | 0.158 |
| MFJS7 | 8×7 | 879 | 0 | **879** | 0 | 881 | 0.228 | 881 | 903.0 | 0.228 |
| MFJS8 | 9×8 | **884** | 0 | **884** | 0 | **903** | 2.149 | **884** | 900.2 | 0 |
| MFJS9 | 11×8 | 1 088 | 3.128 | **1 055** | 0 | 1 180 | 11.848 | 1 097 | 1 127.0 | 3.981 |
| MFJS10 | 12×8 | 1 267 | 5.936 | **1 196** | 0 | 1 349 | 12.793 | 1 248 | 1 282 | 4.348 |
| Mean | | | 0.631 | | 0 | | 1.594 | | | 0.562 |

根据表 2 可知，除了在 Mk5 和 Mk7 算例上略有逊色外，本文算法在其余 8 个算例中均优于或持平于其他算法，并在部分算例中表现出显著优势。以 Mk9 算例为例，其最优调度甘特图如图 3 所示，其中，“9-3”表示工件 9 的第 3 道工序。通过多次实验验证，本文算法相较于其他算法更稳定，且其平均 PRD 值为 0.127%，低于大多数对比算法，显示出优越的整体解质量。

根据表 3,本文算法在求解大部分 Fattahi 算例时取得了较为优越的调度结果,在 20 个算例中获得了 15 个更优的调度方案,显示出较强的优化能力。虽然在少数算例上表现略逊于某些对比算法,但整体上与其他文献算法的差距不大。此外,本文算法在处理复杂且大规模的算例(如 11×8 和 12×8)时,表现出不错的求解效果,进一步证明了其在解决高复杂度和大规模 FJSP 上的有效性和优越性。因此,本文算法在 FJSP 的求解中具有明显优势。

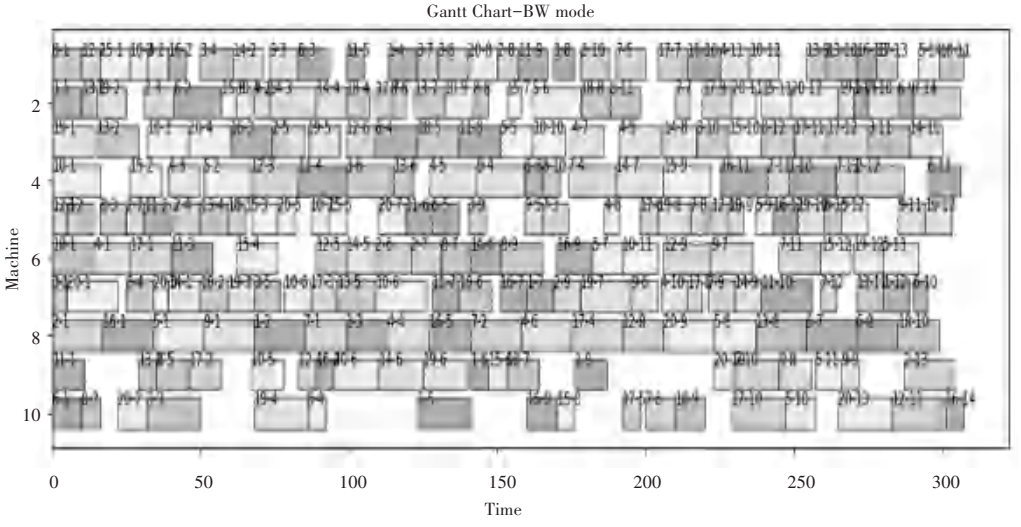



**图 3 Mk9 算例甘特图**

**Fig. 3 Gantt chart for Mk9 instance**

## 4 结束语

本文针对 FJSP,提出了一种结合自适应遗传算法与邻域搜索的优化方法,旨在最小化最大完工时间。通过自适应种群初始化策略与多种邻域结构的结合,提升了算法的全局搜索能力和局部搜索性能。结果表明,本文算法在 30 个基准算例中获得了 23 个更优的调度方案,并在复杂算例中表现出显著优势。本文算法不仅能够有效地解决 FJSP,同时在全局搜索和局部优化方面表现出色,特别是在复杂算例中,能够迅速收敛到优质解。这些结果验证了本文提出的自适应遗传算法与邻域搜索方法的优越性,并为解决实际生产中的调度问题提供了一种高效的优化工具。未来可以进一步研究该算法在多目标优化问题中的应用。

## 参考文献

[1] 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[J]. 重庆大学学报, 2020, 43(1): 1-11.

[2] 周鹏鹏, 翟志波, 戴玉森. 基于改进遗传算法的柔性作业车间调度问题研究[J]. 组合机床与自动化加工技术, 2023 (3): 183-186.

[3] 曲鹏举, 唐向红. 改进粒子群算法在柔性车间调度问题的研究[J]. 机械设计与制造, 2024 (7): 227-231.

[4] 赵小惠, 卫艳芳, 赵雯, 等. 基于混合遗传蚁群算法的多目标 FJSP 问题研究[J]. 组合机床与自动化加工技术, 2023 (1): 188-192.

[5] 王玉芳, 曾亚志, 蒋亚飞. 基于自适应灰狼优化算法的柔性作业车间调度问题[J]. 现代制造工程, 2022 (7): 1-10.

[6] 吴晓雯, 郑巧仙. 基于改进粒子群的柔性作业车间调度问题优化研究[J]. 湖北大学学报(自然科学版), 2022, 44(5): 501-507.

[7] 赵小惠, 卫艳芳, 王凯峰, 等. 改进蚁群算法的柔性作业车间调度问题研究[J]. 组合机床与自动化加工技术, 2022(2): 165-168.

[8] 马随东, 艾尔肯·亥木都拉, 郑威强. 改进 GWO 算法求解柔性作业车间调度问题[J]. 机床与液压, 2024, 52(4): 132-139.

[9] Rahimi V, Arkat J, Farughi H. A vibration damping optimization algorithm for the integrated problem of cell formation, cellular scheduling, and intercellular layout [J]. Computers & Industrial Engineering, 2020, 9(12): 143-148.

[10] 常镜洳, 于东. 一种求解多目标 FJSP 的自学习遗传算法[J]. 小型微型计算机系统, 2022, 43(12): 2465-2470.

[11] 陆科苗, 何利力. 基于改进 NSGA-Ⅱ混合算法求解多目标柔性作业车间调度问题[J]. 智能计算机与应用, 2022, 12(7): 46-51.

[12] 张淑丽, 刘胜辉, 沙晨明. 求解柔性作业车间调度的岛屿模型遗传算法[J]. 哈尔滨理工大学学报, 2016, 21(5): 101-106.

[13] 陈祉烨, 胡毅, 刘俊, 等. 利用强化学习的改进遗传算法求解柔性作业车间调度问题[J]. 科学技术与工程, 2024, 24(25): 10848-10856.

[14] 李雪花, 高全力, 赵辉, 等. 求解柔性作业车间调度问题的混合遗传算法[J]. 计算机技术与发展, 2022, 32(8): 185-190.

[15] 李浩平, 杜昕毅, 朱成彪, 等. 基于改进灰狼算法的柔性作业车间重调度问题研究[J]. 太原理工大学学报, 2024, 55(4): 603-611.

[16] Brucker P, Schlie R. Job-shop scheduling with multi-purpose

machines[J]. Computing, 1990, 45(4): 369-375.
[17]李峥峰, 丁其聪, 张东方, 等. 改进离散麻雀搜索算法求解柔性作业车间调度问题[J]. 现代制造工程, 2023(9): 18-27.
[18]Brandimarte P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41(3): 157-183.
[19]姜天华. 混合灰狼优化算法求解柔性作业车间调度问题[J]. 控制与决策, 2018, 33(3): 503-508.
[20]王彦杰,向凤红. 基于改进蚁狮优化算法的柔性作业车间调度研究[J]. 机电工程, 2022, 39(9): 1325-1332.
[21]杜凌浩,向凤红. 改进多邻域候鸟优化算法的柔性作业车间调度研究[J]. 兵器装备工程学报, 2022, 43(12): 299-306.
[22]Nouri H E, Belkahla D O, Ghédira K. Solving the flexible job shop problem by hybrid metaheuristics-based multi-agent model [J]. Journal of Industrial Engineering International, 2018, 14(1): 1-14.
[23]Xiong W, Fu D. A new immune multi-agent system for the flexible job shop scheduling problem[J]. Journal of Intelligent Manufacturing, 2018, 29(4): 857-873.
[24]Li X Y, Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem[J]. International Journal of Production Economics, 2016, 174: 93-110.
[25]Nowicki E, Smutnicki C. A fast taboo search algorithm for the job shop problem[J]. Management Science, 1996, 42(6): 797-813.
[26]孟冠军, 黄江涛, 魏亚博. 混合白鲸优化算法求解柔性作业车间调度问题[J]. 计算机工程与应用, 2024, 60(12): 325-333.