

# 多品种装配车间调度研究

## 论文答辩

陈晟恺

健行理工 1001 201002750102

指导老师：鲁建厦、董巧英

2014 年 6 月 1 日

# 研究背景及意义

多品种生产方式出现

- 同步装配流水线方式作业是现在汽车装配的主要方式。

# 研究背景及意义

多品种生产方式出现

- 同步装配流水线方式作业是现在汽车装配的主要方式。
- 技术革新，客户需求的多样化，以及精益思想、环保节能观念的出现，汽车工业的生产模式转变为面向订单的大批量、多品种的生产方式。

# 研究背景及意义

## 多品种生产方式出现

- 同步装配流水线方式作业是现在汽车装配的主要方式。
- 技术革新，客户需求的多样化，以及精益思想、环保节能观念的出现，汽车工业的生产模式转变为面向订单的大批量、多品种的生产方式。
- 为配合汽车生产，汽车电子部件的装配生产也要呈现多品种与批量化。

# 研究背景及意义

## 多品种生产方式出现

- 同步装配流水线方式作业是现在汽车装配的主要方式。
- 技术革新，客户需求的多样化，以及精益思想、环保节能观念的出现，汽车工业的生产模式转变为面向订单的大批量、多品种的生产方式。
- 为配合汽车生产，汽车电子部件的装配生产也要呈现多品种与批量化。
- 合理安排装配产线，优化调度作业单元，对保证汽车零部件装配质量，快速响应需求，提高汽车零部件装配线的生产效率有着重要的现实意义。

# 研究背景及意义

## 多品种生产方式出现

- 同步装配流水线方式作业是现在汽车装配的主要方式。
- 技术革新，客户需求的多样化，以及精益思想、环保节能观念的出现，汽车工业的生产模式转变为面向订单的大批量、多品种的生产方式。
- 为配合汽车生产，汽车电子部件的装配生产也要呈现多品种与批量化。
- 合理安排装配产线，优化调度作业单元，对保证汽车零部件装配质量，快速响应需求，提高汽车零部件装配线的生产效率有着重要的现实意义。

# 研究内容

- 生产线调度现状分析

# 研究内容

- 生产线调度现状分析
- 构建调度模型



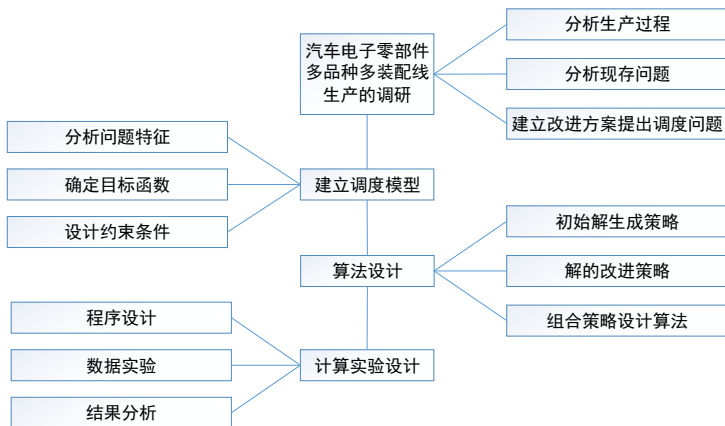
# 研究内容

- 生产线调度现状分析
- 构建调度模型
- 模型求解及算法实现

# 研究内容

- 生产线调度现状分析
- 构建调度模型
- 模型求解及算法实现
- 计算实验

# 研究技术路线



图：课题研究关键技术路线

# 公司基本情况

- 该汽车电子有限公司主要产品为车用电子电器开关、控制模块、控制面板等，是国内外 40 余家汽车主机厂的专业定点配套供应商，配套的产品型号达 6000 余种。

# 公司基本情况

- 该汽车电子有限公司主要产品为车用电子电器开关、控制模块、控制面板等，是国内外 40 余家汽车主机厂的专业定点配套供应商，配套的产品型号达 6000 余种。
- 订单特点是品种多、批量大、小型产品、工艺成熟，所以采用流水线生产是比较合适的，与其合作较多的客户 (主机厂) 一般有其专用线，专门负责该主机厂的订单生产。

# 装配车间现状

采用专线生产的方式，当同一客户有多个订单下达时，按照先到先服务 (FCFS) 的规则进行装配生产安排，多条产线并行作业互不干扰。

<sup>1</sup>其中订单  $a - b$  表示主机厂  $a$  的第  $b$  个订单，下同。

## 装配车间现状

采用专线生产的方式，当同一客户有多个订单下达时，按照先到先服务 (FCFS) 的规则进行装配生产安排，多条产线并行作业互不干扰。



图: 3 条生产线的现行调度<sup>1</sup>

<sup>1</sup>其中订单 a - b 表示主机厂 a 的第 b 个订单，下同。

# 装配车间分析

## 存在问题

现行调度方案存在一些改进空间，例如多条装配线负荷不均衡，有的任务过重，有的任务不足，负荷不均衡，一条装配线上装配的产品工艺相似性较低，导致换线时间增加，产生更长的等待。其主要问题如下：

- 产线利用率低



# 装配车间分析

## 存在问题

现行调度方案存在一些改进空间，例如多条装配线负荷不均衡，有的任务过重，有的任务不足，负荷不均衡，一条装配线上装配的产品工艺相似性较低，导致换线时间增加，产生更长的等待。其主要问题如下：

- 产线利用率低
- 生产不够均衡

# 装配车间分析

## 存在问题

现行调度方案存在一些改进空间，例如多条装配线负荷不均衡，有的任务过重，有的任务不足，负荷不均衡，一条装配线上装配的产品工艺相似性较低，导致换线时间增加，产生更长的等待。其主要问题如下：

- 产线利用率低
- 生产不够均衡
- 产线冗余度高

# 装配车间分析

## 存在问题

现行调度方案存在一些改进空间，例如多条装配线负荷不均衡，有的任务过重，有的任务不足，负荷不均衡，一条装配线上装配的产品工艺相似性较低，导致换线时间增加，产生更长的等待。其主要问题如下：

- 产线利用率低
- 生产不够均衡
- 产线冗余度高
- 工期可控性底

# 装配车间分析

## 存在问题

现行调度方案存在一些改进空间，例如多条装配线负荷不均衡，有的任务过重，有的任务不足，负荷不均衡，一条装配线上装配的产品工艺相似性较低，导致换线时间增加，产生更长的等待。其主要问题如下：

- 产线利用率低
- 生产不够均衡
- 产线冗余度高
- 工期可控性底
- 工艺及设备和生产需求不匹配

# 装配车间分析

## 改进设计

目前的生产现状的主要问题是各主机厂有其专用流水线，使得订单的调度安排为较为单一，受到一些限制，不能很合理地利用流水线的生产能力，所以首要的改进是突破专用线的生产界限。如此一来，生产线可以加工多家主机厂的订单，形成所谓的混线生产。

# 装配车间分析

## 改进设计

目前的生产现状的主要问题是各主机厂有其专用流水线，使得订单的调度安排为较为单一，受到一些限制，不能很合理地利用流水线的生产能力，所以首要的改进是突破专用线的生产界限。如此一来，生产线可以加工多家主机厂的订单，形成所谓的混线生产。



图: 3 条产线的混线装配生产示意

# 模型 1

## 基本假设

- 整数变量假设

# 模型 1

## 基本假设

- 整数变量假设
- 数量有限假设



# 模型 1

## 基本假设

- 整数变量假设
- 数量有限假设
- 无差别假设

# 模型 1

## 基本假设

- 整数变量假设
- 数量有限假设
- 无差别假设
- 无插单生产假设

# 模型 1

## 基本假设

- 整数变量假设
- 数量有限假设
- 无差别假设
- 无插单生产假设
- 不可中断假设

# 模型 1

## 基本假设

- 整数变量假设
- 数量有限假设
- 无差别假设
- 无插单生产假设
- 不可中断假设
- 无相关假设

$$\min \quad \lambda_t \sum_{l=1}^m \sum_{k=1}^{|S_l|} wt_{l_k} T_{l_k} + \lambda_c \sum_{l=1}^m \sum_{k=1}^{|S_l|} wc_{l_k} C_{l_k} \quad (1)$$

$$\text{s.t.} \quad \left\{ \begin{array}{l} \sum_{l=1}^m |S_l| = n \end{array} \right. \quad (2)$$

$$\bigcup_{l=1}^m \bar{S}_l = N \quad (3)$$

$$\sum_{l=1}^m \sum_{k=1}^{|S_l|} wt_{l_k} = 1 \quad (4)$$

$$\sum_{l=1}^m \sum_{k=1}^{|S_l|} wc_{l_k} = 1 \quad (5)$$

$$\lambda_c + \lambda_t = 1 \quad (6)$$

$$C_{l_1} = p_{l_1} \quad l = 1, 2, \dots, m \quad (7)$$

$$C_{l_k} = C_{l_{k-1}} + p_{l_k} \quad k = 2, 3, \dots, |S_l|, l = 1, 2, \dots, m \quad (8)$$

$$p_{l_k} = p'_{l_k} + s_{l_k} \quad k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (9)$$

$$T_{l_k} = \max\{0, C_{l_k} - d_{l_k}\} \quad k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (10)$$

$$p'_{l_k}, s_{l_k}, d_{l_k}, wt_{l_k}, \lambda_t, \lambda_c \geq 0 \quad k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (11)$$

# 模型 2

## 相关假设

- 插单假设

# 模型 2

## 相关假设

- 插单假设
- 惩罚一致假设

# 模型 2

## 相关假设

- 插单假设
- 惩罚一致假设
- 订单最早可处理时刻假设



## 模型 2

### 相关定义

#### 产线均衡率

考虑订单陆续到达时，更为注重订单的按时交付，同时也关注流水线的生产均衡性。生产均衡性指的是流水线的使用均衡，不要出现某条流水线一直繁忙而有些流水线空闲居多，导致负荷不均衡，损失产能。产线均衡率定义如下：

$$Rb = \frac{\sum_{l=1}^m C_l}{m \times \max_{1 \leq l \leq m} \{C_l\}}$$

## 模型 2

### 相关定义

#### 产线的利用率

各流水线除了切换准备，其余时间都在处理订单，在模型 2 中，流水线上订单间的空闲等待将会出现，其中切换准备同样不计入空闲，流水线利用率定义为：

$$Ru_l = 1 - \frac{\sum_{k=1}^{|S_l|} f_{l_k}}{C_l}$$

其中， $f_{l_k}$  为订单的闲置，定义为：

$$f_{l_k} = \begin{cases} \max\{r_{l_k} - s_{l_k}, 0\} & k = 1 \\ \max\{r_{l_k} - s_{l_k} - C_{l_{k-1}}, 0\} & k \geq 2 \end{cases}$$

$$\min \quad \lambda_1 \sum_{l=1}^m \frac{\sum_{k=1}^{|S_l|} w_{l_k} |L_{l_k}|}{Ru_l} + \lambda_2 e^{-Rb} \sum_{l=1}^m \sum_{k=1}^{|S_l|} w_{l_k} C_{l_k} \quad (1)$$

$$\left. \begin{array}{l} \sum_{l=1}^m |S_l| = n \end{array} \right\} \quad (2)$$

$$\left. \begin{array}{l} \bigcup_{l=1}^m \overline{S_l} = N \end{array} \right\} \quad (3)$$

$$\left. \begin{array}{l} \sum_{l=1}^m \sum_{k=1}^{|S_l|} w_{l_k} = 1 \end{array} \right\} \quad (4)$$

$$\left. \begin{array}{l} \sum_{l=1}^m \sum_{k=1}^{|S_l|} w_{l_k} C_{l_k} = 1 \end{array} \right\} \quad (5)$$

$$\left. \begin{array}{l} \lambda_1 + \lambda_2 = 1 \end{array} \right\} \quad (6)$$

$$\left. \begin{array}{l} C_{l_1} = f_{l_1} + s_{l_1} + p_{l_1} \end{array} \right\} \quad l = 1, 2, \dots, m \quad (7)$$

$$\left. \begin{array}{l} C_{l_k} = C_{l_{k-1}} + f_{l_k} + s_{l_k} + p_{l_k} \end{array} \right\} \quad k = 2, 3, \dots, |S_l|, l = 1, 2, \dots, m \quad (8)$$

$$\left. \begin{array}{l} \sum_{l=1}^m \sum_{k=1}^{|S_l|} r_{l_k} > 0 \end{array} \right\} \quad k = 2, 3, \dots, |S_l|, l = 1, 2, \dots, m \quad (9)$$

$$\left. \begin{array}{l} L_{l_k} = C_{l_k} - d_{l_k} \end{array} \right\} \quad k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (10)$$

$$\left. \begin{array}{l} T_{l_k} = \max\{0, C_{l_k} - d_{l_k}\} \end{array} \right\} \quad k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (11)$$

$$\left. \begin{array}{l} E_{l_k} = \max\{d_{l_k} - C_{l_k}, 0\} \end{array} \right\} \quad k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (12)$$

$$\left. \begin{array}{l} s_{l_k}, d_{l_k}, w_{l_k}, w_{l_k} C_{l_k}, \lambda_1, \lambda_2, r_{l_k} \geq 0 \end{array} \right\} \quad k = 1, 2, \dots, |S_l|, l = 1, 2, \dots, m \quad (13)$$

# 初始解构造

## 复合分派规则

复合分派规则是综合了许多基本规则的一个表达式，各基本规则都有其各自的比例参数，用来给作业的排序提供参考，没有固定的形式，可以用作调度问题初始解的求解。

# 初始解构造

## 复合分派规则

复合分派规则是综合了许多基本规则的一个表达式，各基本规则都有其各自的比例参数，用来给作业的排序提供参考，没有固定的形式，可以用作调度问题初始解的求解。

### ATC 规则

$$l_j(t) = \frac{wt_j}{p_j} \exp \left( -\frac{\max\{d_j - p_j - t, 0\}}{K\bar{p}} \right)$$

# 初始解构造

## 复合分派规则

模型 1 适合用 ATC 规则进行初始解的构造，按照系统时间  $t$  的进行，动态判断各流水线闲忙状态，若有流水线处于空闲状态，则根据排序指数选出下一个进行处理的订单，将其安排入该空闲流水线，更新流水线状态及待调度订单列表，预估该流水线的下一次空闲时刻，重复这个步骤一直到所有订单都被调度。

# 解的改进

交替调整策略 (Cycly Amend, Cyc)

- 流水线内部调整

# 解的改进

交替调整策略 (Cycly Amend, Cyc)

- 流水线内部调整
  - 使用规则调整



# 解的改进

交替调整策略 (Cycly Amend, Cyc)

- 流水线内部调整
  - 使用规则调整
  - 区域搜索调整

# 解的改进

交替调整策略 (Cycly Amend, Cyc)

- 流水线内部调整
  - 使用规则调整
  - 区域搜索调整
- 流水线之间调整

# 解的改进

## 交替调整策略 (Cycly Amend, Cyc)

- 流水线内部调整
  - 使用规则调整
  - 区域搜索调整
- 流水线之间调整
  - 流水线贡献值

# 解的改进

## 交替调整策略 (Cycly Amend, Cyc)

- 流水线内部调整
  - 使用规则调整
  - 区域搜索调整
- 流水线之间调整
  - 流水线贡献值
  - 订单贡献值

## Cyc – ATC 算法

- Step1** 初始化。  $J = N, \bar{L} = \emptyset, t_l = 0, \bar{S}_l = \emptyset, a_l = 0, (l = 1, 2, \dots, m)$ , 计算各订单处理时间  $p'_j = g(j, n_j)$ , 进一步得到整合订单处理时间  $p_j = p'_j + s_j, (j = 1, 2, \dots, n)$ , 置系统时间  $t = 0$ ;
- Step2** 若存在  $a_l = 0$ , 记  $l^* = \min_{a_l=0} \{l\}$ , 执行 **Step3**, 否则执行 **Step4**;
- Step3** 根据排序指数, 选取预备调度订单  $j^*$ , 使得  $I_{j^*}(t) = \max_{j \in J} \{I_j(t)\}$ , 将订单  $j^*$  安排入流水线  $l^*$  进行处理, 记入调度  $S_{l^*}$ ,  $\bar{S}_{l^*} = \bar{S}_{l^*} \cup \{j^*\}, J = J - \{j^*\}$ , 记录调度订单序列  $\bar{L} = \bar{L} \cup \{j^*\}$ , 更新流水线预计空闲时刻  $t_{l^*} = t + p_{j^*}$ , 修改流水线状态  $a_{l^*} = 1$ 。若  $J = \emptyset$ , 订单初始调度完毕, 执行 **Step5**, 否则执行 **Step2**;
- Step4** 记  $l_t$  使得  $t_{l_t} = \min_{1 \leq l \leq m} \{t_l\}$ , 修改流水线状态  $a_{l_t} = 0$ , 并更新系统时间  $t = t_{l_t}$ , 执行 **Step2**;
- Step5** 设定交替次数  $NR$ , 置  $k = 1$ ;
- Step6** 根据各流水线的贡献值  $H(S_l)$  值, 选出具有最大值与最小值的流水线, 分别记为  $l^+, l^-$ ;
- Step7** 根据流水线  $l^+$  的调度  $S_{l^+}$  中具有最大贡献值  $h(l_k)$  值的订单  $l_k^+$ , 并将其添入流水线  $l^-$  的调度  $S_{l^-}$  末端, 更新流水线  $l^+, l^-$  的订单安排序列;
- Step8** 内部调整初始化。  $J = \bar{S}_l (l = l^+, l^-)$ , 置所选的流水线系统时间  $t_l = 0$ , 重置  $\bar{S}_l = \emptyset$ ;
- Step9** 根据排序指数, 选取预备调度订单  $l_k^*$ , 使得  $I_{l_k^*}(t) = \max_{l_k \in J} \{I_{l_k}(t)\}$ , 将订单  $l_k^*$  进行安排处理,  $J = J - \{l_k^*\}$ , 将该订单排入该流水线的调度  $\bar{S}_l = \bar{S}_l \cup \{l_k^*\}$ 。若  $J = \emptyset$ , 该流水线上的订单调度完毕, 执行 **Step11**, 否则执行 **Step10**;
- Step10** 更新流水线系统时间  $t_l = t_l + p_{l_k^*}$ , 执行 **Step9**;
- Step11** 置  $k = k + 1$ , 若  $k \leq NR$ , 执行 **Step6**, 否则终止算法。

## Cyc – Tabu 算法

**Step1** 根据调度分派规则生成初始调度解  $S$ ;

**Step2** 设定交替次数  $NR$ , 置  $k_r = 1$ ;

**Step3** 根据各流水线的贡献值  $H(S_l)$  值, 选出具有最大值与最小值的流水线, 分别记为  $l^+$ ,  $l^-$ ;

**Step4** 根据流水线  $l^+$  的调度  $S_{l^+}$  中具有最大贡献值  $h(l_k)$  的订单  $l_{k^*}^+$ , 并将其添入流水线  $l^-$  的调度  $S_{l^-}$  末端, 更新流水线  $l^+$ ,  $l^-$  的订单安排序列;

**Step5** 初始化。设定迭代次数  $N_l$ , 清空禁忌列表  $TL$ , 设定列表长度  $NL$ , 将构造算法所得的调度作为初始调度, 并记为当前最优调度,  $S^{(0)} = S^{(1)} = S_l(l = l^+, l^-)$ , 并置  $k = 1$ ;

**Step6** 从  $S^{(k)}$  所有不在禁忌列表中的相邻移动  $(l_j, l_k)$  中, 所得调度具有最小函数值的移动, 记为  $(l_j^*, l_k^*)$ , 所得调度记为  $S^*$ , 并置  $S^{(k+1)} = S^*$ ;

**Step7** 将相邻移动  $(l_j^*, l_k^*)$  入栈禁忌列表, 若列表容量已满, 则按 FIFO 规则出栈最早的相邻移动;

**Step8** 若  $G(S^*) < G(S^{(0)})$ , 置  $S^{(0)} = S^*$ ;

**Step9** 置  $k = k + 1$ , 若  $k \leq N_l$ , 执行 **Step6**, 否则禁忌搜索调整完成, 更新调度解  $S$ , 执行 **Step10**;

**Step10** 置  $k_r = k_r + 1$ , 若  $k_r \leq NR$ , 执行 **Step2**, 否则终止算法。

# 解的改进

虚拟序列策略 (Virtual List, Vtr)

## 虚拟序列

将所有流水线上的调度看作一个整体，所有订单都在这个序列上，其排列顺序由初始解的生成规则决定，也就是在调度安排时的记录序列  $L$ ，并按先后顺序记该序列上的订单为  $L_j, (j = 1, 2, \dots, n)$ 。

# 解的改进

虚拟序列策略 (Virtual List, Vtr)

## 虚拟序列

虚拟序列上只有所有订单的先后信息，其订单的一种排序称为一种虚拟调度。

- 相邻订单  $L_j, L_k$  安排在同一条流水线  $l$  上进行处理
- 相邻订单  $L_j, L_k$  分别安排在不同流水线  $l, l'$  上进行处理

## 注意过度禁忌



# 解的改进

虚拟序列策略 (Virtual List, Vtr)

## 虚拟序列

虚拟序列上只有所有订单的先后信息，其订单的一种排序称为一种虚拟调度。

- 相邻订单  $L_j, L_k$  安排在同一条流水线  $l$  上进行处理
- 相邻订单  $L_j, L_k$  分别安排在不同流水线  $l, l'$  上进行处理

## 注意过度禁忌

- 流水线之间相邻订单过度禁忌

# 解的改进

虚拟序列策略 (Virtual List, Vtr)

## 虚拟序列

虚拟序列上只有所有订单的先后信息，其订单的一种排序称为一种虚拟调度。

- 相邻订单  $L_j, L_k$  安排在同一条流水线  $l$  上进行处理
- 相邻订单  $L_j, L_k$  分别安排在不同流水线  $l, l'$  上进行处理

## 注意过度禁忌

- 流水线之间相邻订单过度禁忌
- 流水线之内相邻订单过度禁忌

## Vtr – Tabu 算法

- Step1** 运用调度规则 (如 ATC、ATCS) 建立流水线全局调度初始解, 得到虚拟序列  $L$  及其初始调度  $S^{(0)}$ , 并将其作为目前最优调度。设定禁忌搜索迭代次数  $N_I$ , 设定列表长度  $NL$ , 并置特赦调度  $A = S^{(0)}$ ;
- Step2** 置  $S^{(1)} = S_{(0)}$ , 清空禁忌列表  $TL$ , 置  $k = 1$ ;
- Step3** 在  $L$  所生成的邻域中, 按顺序选取  $(L_m, L_n)$ , 记当前调度为  $S^-$ , 若  $L_m, L_n$  当前均安排在同一流水线的调度中, 则执行 **Step4**, 否则执行 **Step5**;
- Step4** 交换订单对顺序, 得到新的调度为  $S^+$  型;
- Step5** 将订单  $L_m$  重派入流水线  $l'$ , 得到调度为  $S^{a+}$  型, 或将订单  $L_n$  重派入流水线  $l$  得到调度为  $S^{b+}$  型, 或将订单  $L_m, L_n$  交换位置, 得到调度为  $S^+$  型;
- Step6** 更新虚拟序列中这两个订单的位置为  $(L_m, L_n)$ ;
- Step7** 检查禁忌列表中的订单对, 若它们别安排在不同的流水线, 则只对其交换位置的移动禁忌; 若移动后的调度为特赦调度, 一样认定为可行移动。计算  $S^{(k)}$  中所有可行移动组成的邻域, 选取它们中具有最小函数值调度的移动, 记该订单对为  $(L_m^*, L_n^*)$ , 所得调度记为  $S^*$ , 并置  $S^{(k+1)} = S^*$ ;
- Step8** 若相邻移动所得调度属于  $S^+$  型, 则将  $(L_m^*, L_n^*)$  入栈禁忌列表, 若列表容量已满, 则按 FIFO 规则出栈最早的相邻移动, 检查禁忌列表, 删除过禁忌项;
- Step9** 若  $G(S^*) < G(S^{(0)})$ , 置  $S^{(0)} = S^*$ ;
- Step10** 置  $k = k + 1$ , 若  $k \leq N_I$ , 执行 **Step3**, 否则终止算法,  $S^{(0)}$  为最终所得调度。

# 模型 2 解的改进

## 变动邻域策略 (Variate Neighbor, VN)

Vtr - Tabu 算法可以得到较有的结果，然而由于其邻域结构的特点，可能需要很大的迭代次数才能将解改进。采用变动邻域的策略可以人为切换邻域结构，放弃一些需要过多迭代次数的邻域结构，以减少计算时间，这样的综合策略称为 VVT (变动邻域结构的虚拟序列禁忌搜索)

## VVT 算法

- Step1** 运用调度规则 (如 ATC、ATCS) 建立流水线全局调度初始解, 得到虚拟序列  $L$  及其初始调度  $S^{(0)}$ , 并将其作为目前最优调度, 将其邻域集合  $S^{(c)}$  中的调度按函数值的非减排列, 记为  $S_{[1]}, S_{[2]}, \dots, S_{[|S^{(c)}|]}$ , 置  $i = 1$ 。设定禁忌搜索迭代次数  $N_I$ , 设定列表长度  $NL$ , 并置特赦调度  $A = S^{(0)}$ ;
- Step2** 若  $i \leq |S^{(c)}|$ , 置  $S^{(1)} = S_{[i]}$ , 清空禁忌列表  $TL$ , 置  $k = 1$ , 否则终止算法;
- Step3** 在  $L$  所生成的邻域中, 按顺序选取  $(L_m, L_n)$ , 记当前调度为  $S^-$ , 若  $L_m, L_n$  当前均安排在同一流水线的调度中, 则执行 **Step4**, 否则执行 **Step5**;
- Step4** 交换订单对顺序, 得到新的调度为  $S^+$  型;
- Step5** 将订单  $L_m$  重派入流水线  $l'$ , 得到调度为  $S^{a+}$  型, 或将订单  $L_n$  重派入流水线  $l$  得到调度为  $S^{b+}$  型, 或将订单  $L_m, L_n$  交换位置, 得到调度为  $S^+$  型;
- Step6** 更新虚拟序列中这两个订单的位置为  $(L_m, L_n)$ 。
- Step7** 计算  $S^{(k)}$  中所有可行移动组成的邻域, 选取它们中具有最小函数值调度的移动, 记该订单对为  $(L_m^*, L_n^*)$ , 所得调度记为  $S^*$ , 并置  $S^{(k+1)} = S^*$ ;
- Step8** 若相邻移动所得调度属于  $S^+$  型, 则将  $(L_m^*, L_n^*)$  入栈禁忌列表, 若列表容量已满, 则按 FIFO 规则出栈最早的相邻移动;
- Step9** 若  $G(S^*) < G(S^{(0)})$ , 置  $S^{(0)} = S^*$ ;
- Step10** 置  $k = k + 1$ , 若连续 50 次采用没有更新  $S^{(0)}$ , 则置  $i = i + 1$ , 执行 **Step2**, 否则若  $k \leq N_I$ , 执行 **Step3**, 否则终止算法,  $S^{(0)}$  为最终所得调度。

# 实验设计

## 生产装配信息生成

- 数量信息

# 实验设计

## 生产装配信息生成

- 数量信息
- 时间信息

# 实验设计

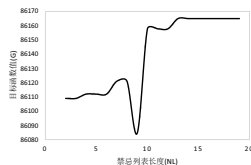
## 生产装配信息生成

- 数量信息
- 时间信息
- 惩罚系数和有限系数

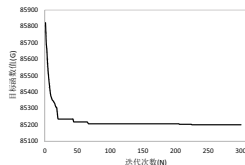


# 实验设计

## 相关参数确定



(a) 禁忌列表长度

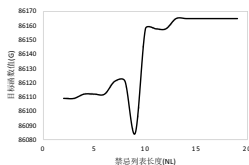


(b) 迭代次数

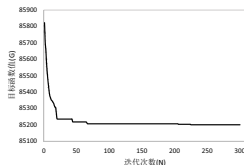
图: 100 件订单的目标函数值和相关参数的关系

# 实验设计

## 相关参数确定



(a) 禁忌列表长度



(b) 迭代次数

图: 100 件订单的目标函数值和相关参数的关系

可以看出,  $NL = 9$  是最佳列表长度, 取迭代次数  $N = 70$  是较为适宜的

# 实验结果示例

以模型 1 为例，示例订单量  $n = 20$ ，决策参数为  $\lambda_1 = 0.6, \lambda_2 = 0.4$

## 实验结果示例

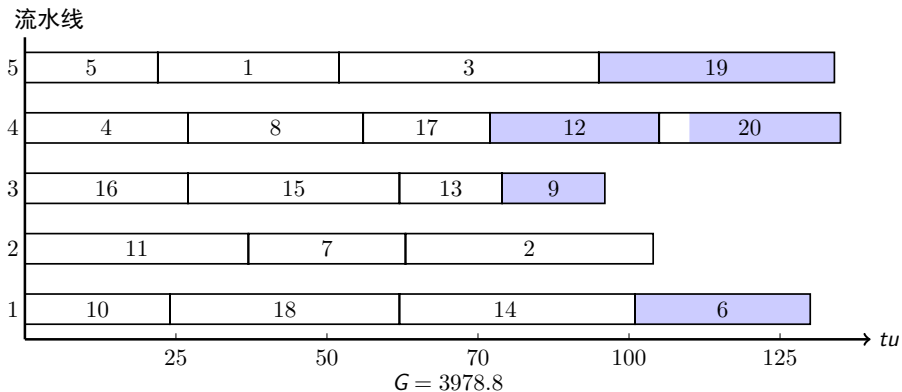


图: Cyc - ATC 算法调度结果

## 实验结果示例

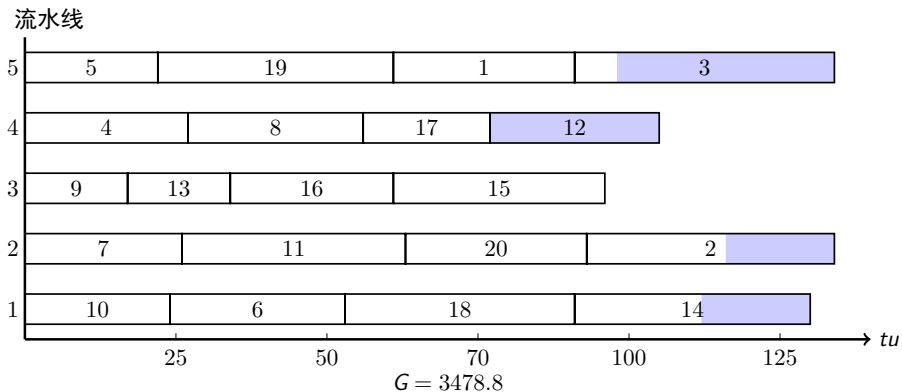


图: Cyc - Tabu 算法调度结果

# 实验结果示例

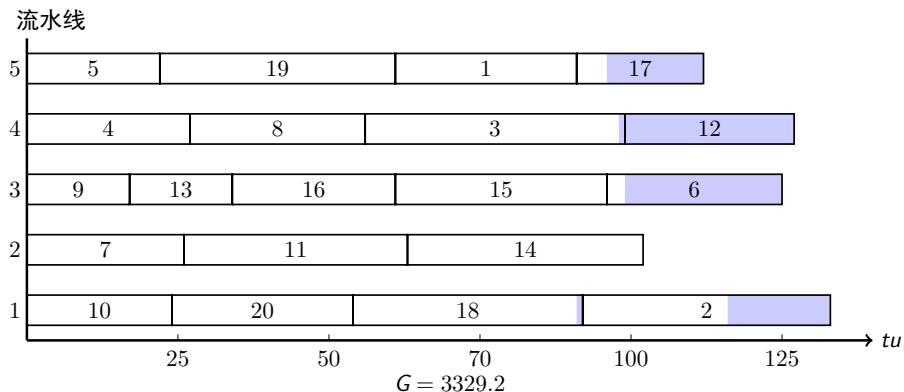


图: Vtr – Tabu 算法调度结果

# 模型 1 求解结果与分析

## 不同决策环境分析

以  $m = 6, n = 200$  为例

# 模型 1 求解结果与分析

## 不同决策环境分析

以  $m = 6, n = 200$  为例

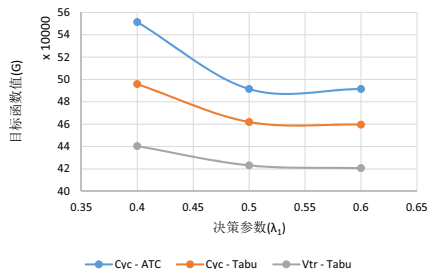
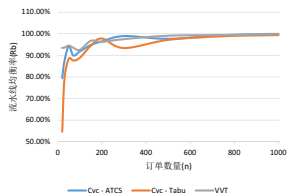


图: 决策参数和目标函数值关系示例

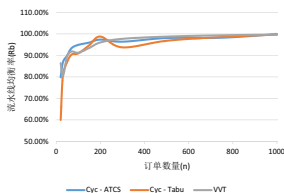


# 模型 2 求解结果与分析

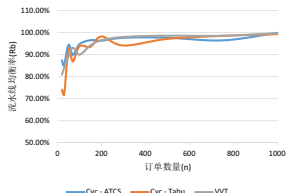
## 流水线均衡率分析



(a)  $m = 5$



(b)  $m = 6$

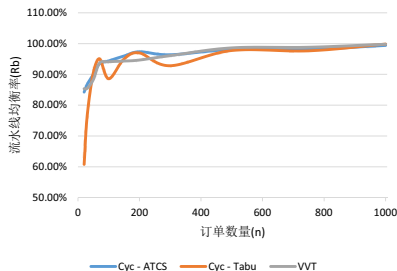


(c)  $m = 7$

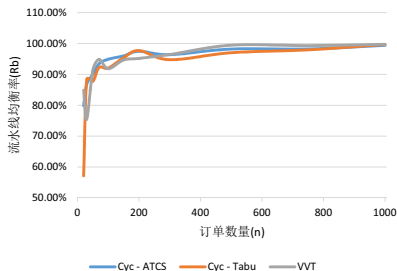
图: 不同流水线数量和流水线均衡率关系 ( $\lambda_1 = 0.5$ )

# 模型 2 求解结果与分析

## 不同决策环境分析



(a)  $\lambda_1 = 0.4$



(b)  $\lambda_1 = 0.6$

图: 决策环境和流水线均衡率关系 ( $m = 6$ )

谢 谢！