中国科学技术大学 USTC
智能决策博弈与数字创新经济安徽省重点实验室
Anhui Province Key Laboratory of Intelligent Decision Games and Digital Economic Advancements

Extended Arc-Time-Indexed Formulation and Exact Approach for Flow Shop Scheduling

报告人：陈　龙

日　期：2025年2月17日

CONTENTS

1.1 机器调度问题的两类模型

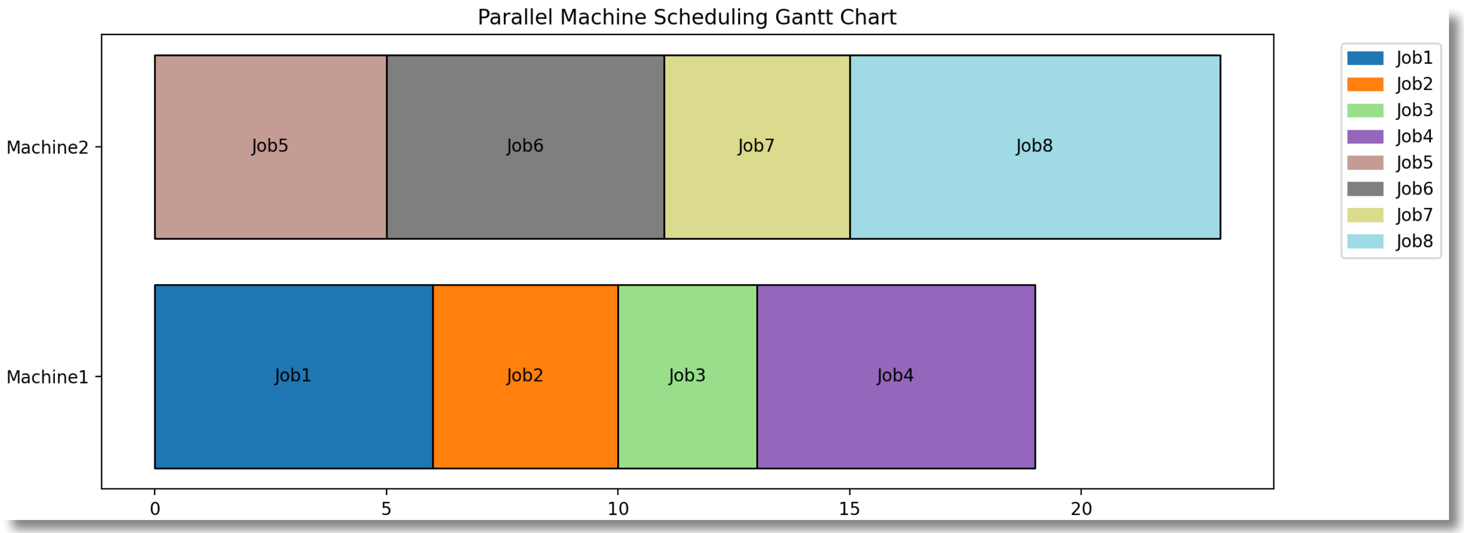
- **Position-based formulation:** 一种比较直观的建模方式，决策变量表示工件是否在某个位置进行加工、以及其对应的加工时间，求解结果通常能够比较直观的表示为排程的甘特图。

Example 1

考虑平行机调度问题$P_2||\sum f_j(C_j)$，目标函数是加权总完工时间，相关参数如下表所示：

$job\ j$	p_j	d_j	w_j
1	6	100	1
2	4	100	1
3	3	100	1
4	6	100	1
5	5	100	1
6	6	100	1
7	4	100	1
8	8	100	1

- 决策变量：
  - 平行机调度：$x_{ij[r]}$, binary, 工件 J_i 是否被分配到机台 M_j 的位置 $[r]$ 进行加工；
  - 流水车间调度：$x_{i[r]}$, binary, 工件 J_i 是否被分配到位置 $[r]$ 进行加工.



1.1 机器调度问题的两类模型

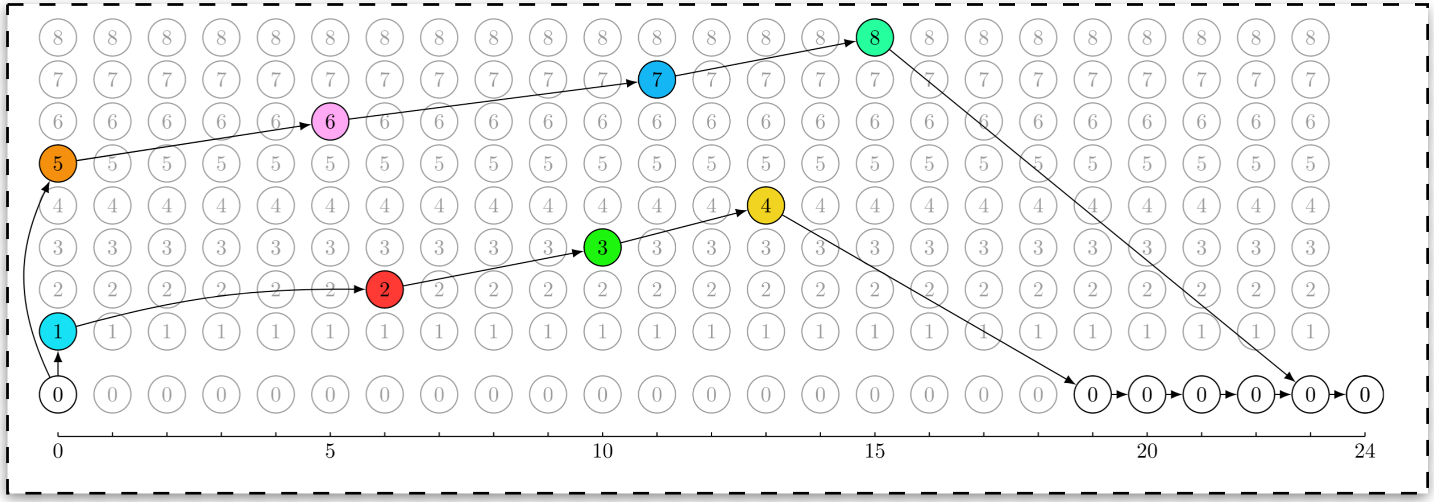
- **Arc-Time-Indexed Formulation:** 一种不太直观的建模方式，使用网络流图 $G=(V,A)$ 来表示调度方案，节点表示某工件在某时刻是否被加工，弧表示调度的先后次序。决策变量表示某个弧是否在可行解中被选择。**(动机?)**

Example 1

考虑平行机调度问题$P_2||\sum f_j(C_j)$，目标函数是加权总完工时间，相关参数如下表所示：

job j	p_j	d_j	w_j
1	6	100	1
2	4	100	1
3	3	100	1
4	6	100	1
5	5	100	1
6	6	100	1
7	4	100	1
8	8	100	1

- 网络图定义 $G=(V,A)$:
  - Vertex: $V=\{(j,t): i\in J_0, t\in\{0,1,2,\ldots,T-1\}\}\cup\{0,T\}$
  - Each arc: $(i,j)^t=((i,t-p_i),(j,t))\in A$
- 决策变量：x_{ij}^t , whether the arc $(i,j)^t$ is selected in the feasible solution or not.



1.2 流水车间调度问题及其应用

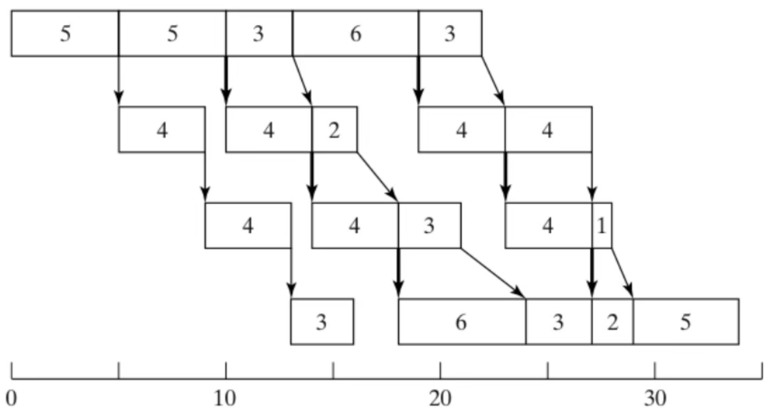
- 流水车间的定义和特征:
  - 定义：工件按照固定的生产顺序依次进入各工序。
  - 特征：每个工件的工序顺序相同、工件在完成当前工序后立即进入下一工序。

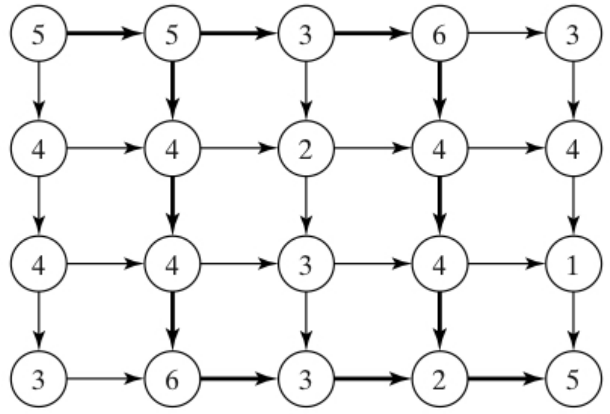
Example 2

考虑流水车间调度问题$F_m||C_{\max}$，目标函数是最大完工时间，相关参数如下表所示：

	p_{1,j_k}	p_{2,j_k}	p_{3,j_k}	p_{4,j_k}
1	5	4	4	3
2	5	4	4	6
3	3	2	3	3
4	6	4	4	2
5	3	4	1	5

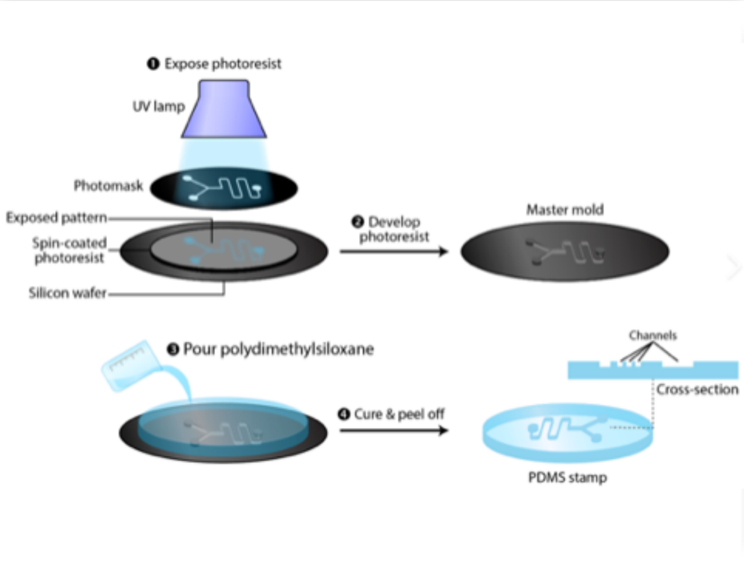
- 假设可行解的 job sequence 为: job1 -> job 2 -> job 3 -> job4 -> job 5
- 观察到流水车间的可视化图天然具有“流”的特征，因此适于采用网络流图的思想进行建模。





1.2 流水车间调度问题及其应用

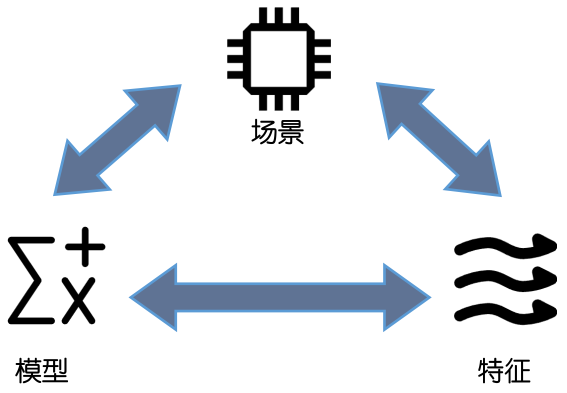
- 流水车间在现实场景的应用:
  - I. 半导体芯片生产制造的过程：炉管区 ➡ 酸槽区 ➡ 黄光区 ➡ …
  - II. 黄光区的生产工艺流程：涂胶 ➡ 曝光 ➡ 显影
  - III. 汽车整车制造工艺流程：冲压 ➡ 焊接 ➡ 涂装 ➡ 总装

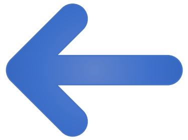


1.3 研究空缺

- ATIF 模型已经广泛应用于单机调度、平行机调度等问题中，并有成熟的算法框架提高计算效率。
  - 研究空缺：现有研究往往聚焦于如何提高ATIF模型在单机、平行机问题的计算效率，而缺乏拓展到其他机器环境的研究
- 流水车间的生产环境在现实场景中广泛应用，其“流”的特征也很适合采用网络流的思想建模和求解。
  - 研究空缺：当前流水车间基于B&B的精确算法可以实现的求解规模较小，难以拓展到工业级大规模数据中

流水车间



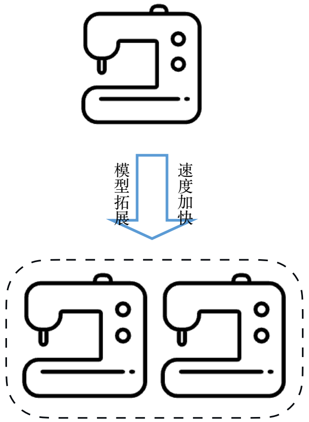


- 如何适配？(Contribution)
  - 添加 Machine-index
  - 添加 idle arc
  - 基于模型特征设计割平面

ATIF的应用与成效

a) (MPC 2010) 首次提出ATIF模型求解调度问题，首次能够精确求解中等规模算例；

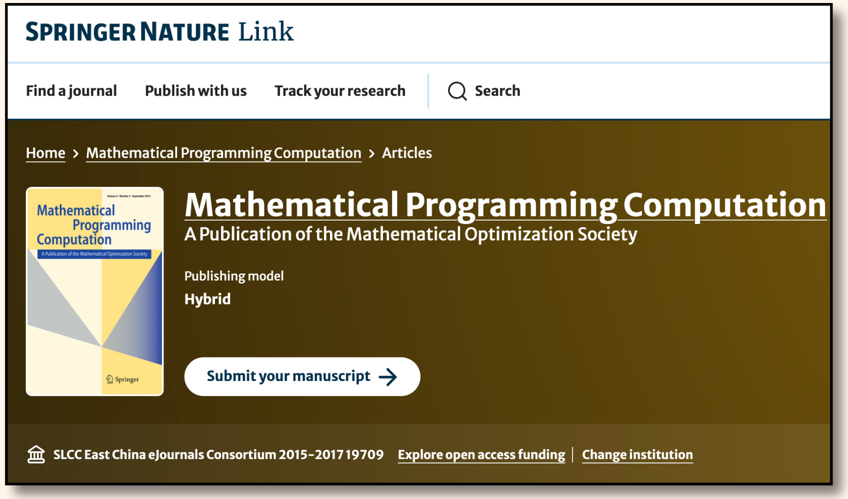
b) (JOC 2020) 为 ATIF 模型提出割平面以加速求解平行机调度，求解效率提升了95%.



2.1 调度问题中的ATIF模型综述

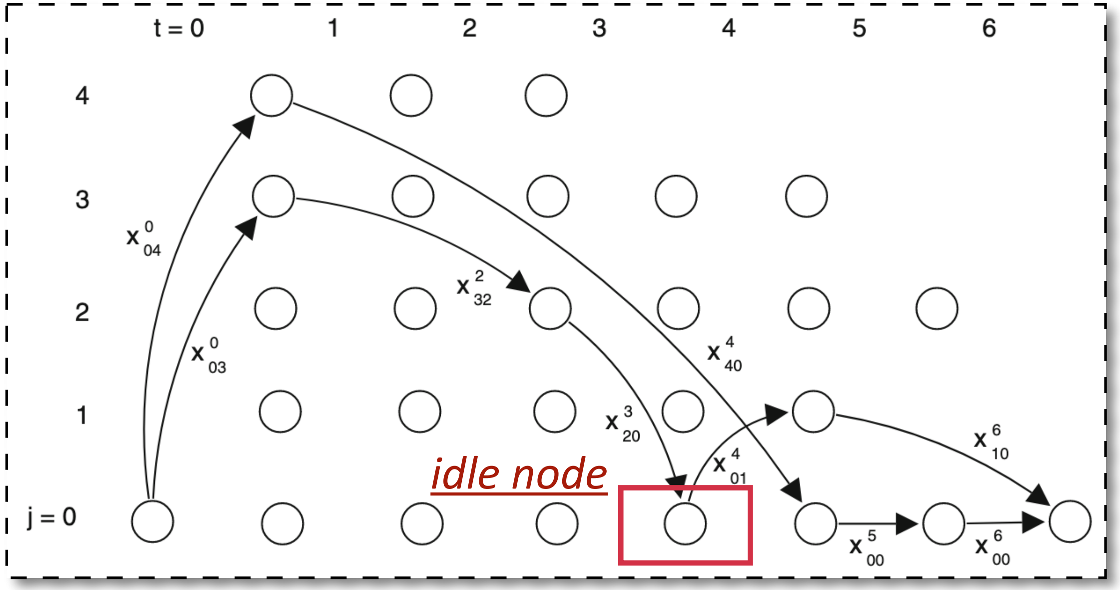
- Exact Algorithm over an **Arc-Time-Indexed Formulation** for Parallel Machine Scheduling Problems (**Mathematical Programming Computation**, Artur Pessoa et al., 2010)

🎁 **期刊：Mathematical Programming Computation (MPC)**



- **分区：**中科院分区应用数学1区，JCR分区Q1区
- **影响因子：**2021-8.06，2022-6.3，2023-4.3
- **发文量：**2023年19篇，2024年21篇

- 首次提出 Arc-Time-Indexed 的模型用于求解平行机调度问题；
- 对于问题 $P||\sum W_j T_j$ 的中等规模算例，首次能够精确求解；
- 求解范式为，基于 Dantzig-Wolfe 重构模型，并使用 B&P 求解。



2.1 调度问题中的ATIF模型综述

- An Improved **Branch-Cut-and-Price** Algorithm for Parallel Machine Scheduling Problems (JOC, Daniel Oliveira & Artur Pessoa, 2020)
  - 完全遵循 (MPC 2010) 的研究问题和求解范式，主要贡献是提出了一族割平面，求解效率平均提高了97%.

$$\text{Minimize} \quad \sum_{i\in J_0}\sum_{j\in J\setminus\{i\}}\sum_{t=p_i}^{T-p_j} f_j(t+p_j)\,x_{ij}^t \qquad (1a)$$

$$\text{Subject to} \quad \sum_{j\in J_0} x_{0j}^0 = m, \qquad (1b)$$

$$\sum_{i\in J_0\setminus\{j\}}\sum_{t=p_i}^{T-p_j} x_{ij}^t = 1, \quad \forall j\in J, \qquad (1c)$$

$$\sum_{\substack{j\in J_0\setminus\{i\},\\ t-p_j\geq 0}} x_{ji}^t - \sum_{\substack{j\in J_0\setminus\{i\},\\ t+p_i+p_j\leq T}} x_{ij}^{t+p_i} = 0, \quad \forall i\in J;\ t=0,\dots,T-p_i, \qquad (1d)$$

$$\sum_{\substack{j\in J_0,\\ t-p_j\geq 0}} x_{j0}^t - \sum_{\substack{j\in J_0,\\ t+p_j+1\leq T}} x_{0j}^{t+1} = 0, \quad t=0,\dots,T-1, \qquad (1e)$$

$$x_{ij}^t \in Z^+, \quad \forall i\in J_0;\ \forall j\in J_0\setminus\{i\};\ t=p_i,\dots,T-p_j, \qquad (1f)$$

$$x_{00}^t \in Z^+, \quad t=0,\dots,T-1. \qquad (1g)$$

Definition 1: Overload Elimination Cuts

$$\sum_{q=t_S}^{r-1} v^q + \sum_{q=r}^{T} 2\,v^q - \sum_{\substack{q=\max\{r-1,\\ T-p(S)+m(t_S-1)+1\}}}^{T-1} u^q \geq 2,$$

$$r = p(S) - (m-1)(t_S-1).$$

Theorem 1

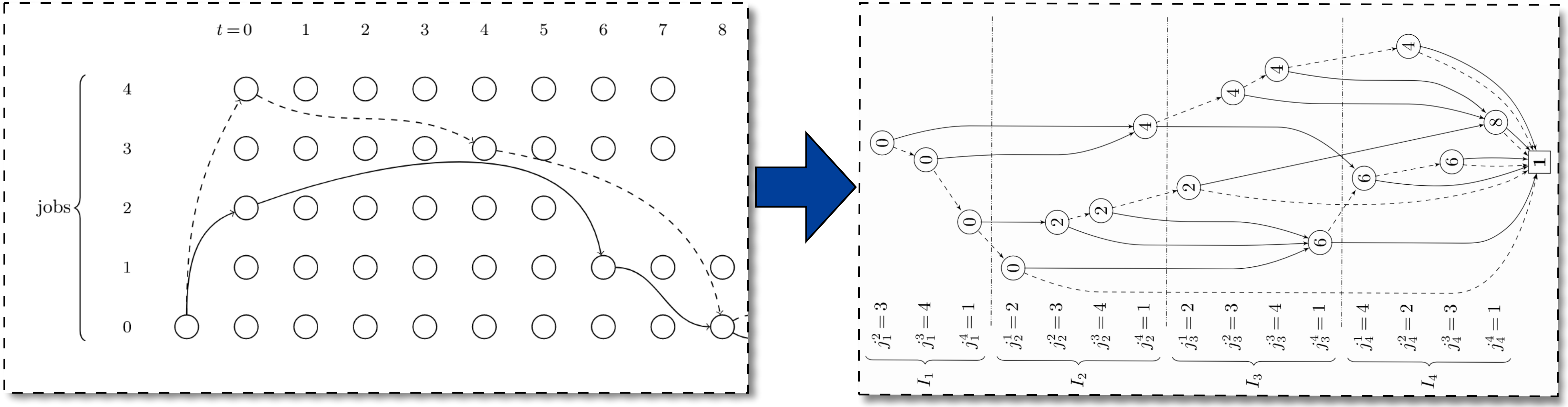
The OEC is valid for ATIF.

Table 5. Full Results—Summary

		BCP-PMWT		BCP-PMWT-OTI	
n	m	Number solved	Average time (s)	Number solved	Average time (s)
40		50	357.9	50	42.8
50		50	5,734.9	50	142.0
100	2	18	22,523.8	24	852.2
100	4	16	37,667.7	22	7,396.2

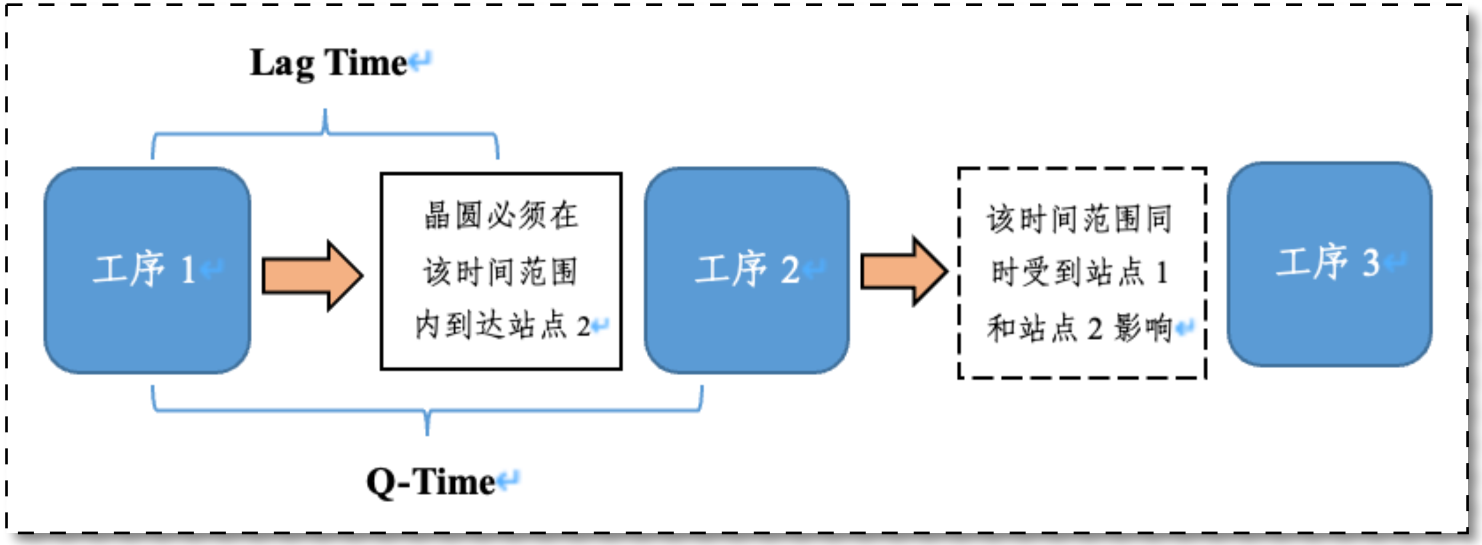
2.1 调度问题中的ATIF模型综述

- A **Flow-Based Formulation** for Parallel Machine Scheduling Using **Decision Diagrams** (JOC, Daniel Kowalczyk et al., 2024)
  - 使用二元决策图 (Binary Decision Diagram, BDD)，对调度的时间进行划分，排除了不可能导致最优的节点和弧，从而大大压缩了网络图的规模；
  - 问题模型和求解范式仍然与 (MPC 2010) 相同，求解算法为基于 Dantzig-Wolfe 重构模型、并使用 B&P 求解.



2.2 半导体制造场景的流水车间调度模型综述

- A Two-Machine Flow Shop Sequencing Problem with limited waiting time constraints (CIE, Yang & Chern, 1995)
  - The first paper to consider **limited waiting time constraints**, proving its NP-hard nature.
  - Presenting theoretical results to eliminate nodes in the enumeration tree, enhancing the algorithm's efficiency.



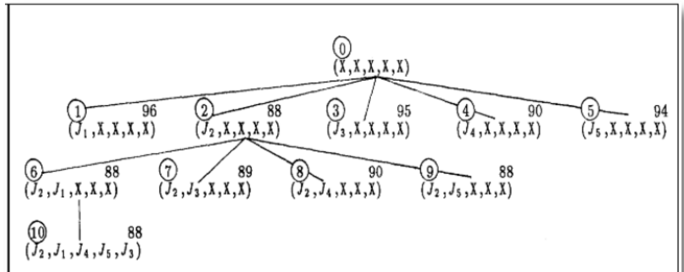


Fig. 5. The enumeration tree.

Abstract

We consider a two-machine flowshop sequencing problem with limited waiting time constraints. This means that for each job the waiting time between two machines cannot be greater than a given upper bound. The objective is to minimize the makespan. There are efficient algorithms for the special cases of infinite waiting time and zero waiting time. The two-machine flowshop sequencing problem with limited waiting time constraints is shown to be NP-hard. A branch-and-bound algorithm is proposed, and computational experiments are provided.

2.2 半导体制造场景的流水车间调度模型综述

- Minimizing Makespan in a Two-Machine Flow Shop with a Limited Waiting Time Constraint and Sequence-Dependent Setup Times (COR, Kim & Choi, 2016)
  - This paper tackles a two-machine flow shop scheduling problem with **limited waiting time constraints** and **sequence-dependent setup times**.
  - It develops a branch-and-bound algorithm, along with dominance properties and lower bounds, to efficiently solve this NP-hard problem. (up to 30 jobs)

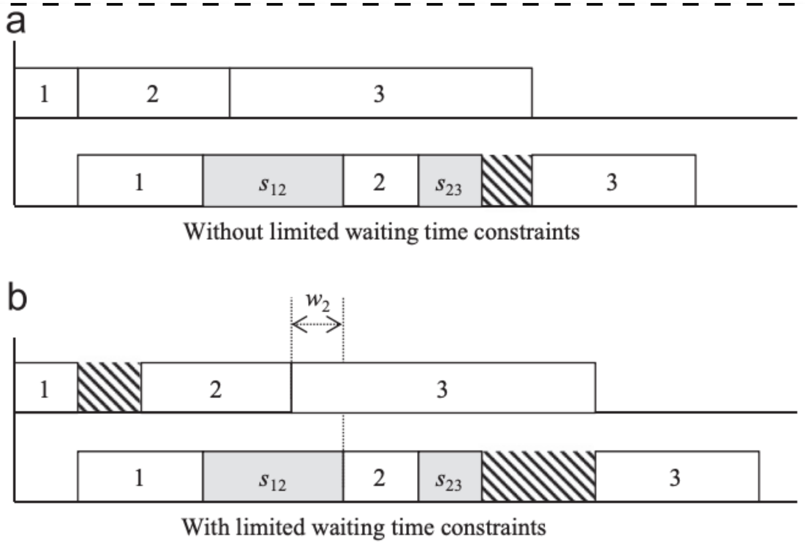
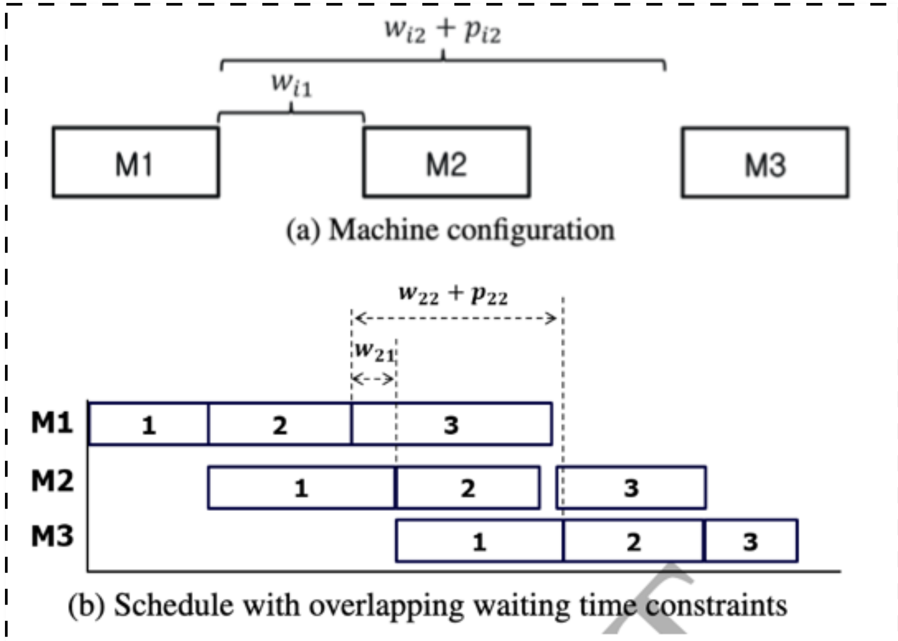


Table 6
Results of the test of the B&B algorithm on set 2 instances.

n	ACPUT[a]	MCPUT[b]	NINS[c]	$CPUT_{CPLEX}$[d]	$NINS_{CPLEX}$[e]
10	0.01	0.01	0	7.28	0
15	0.75	0.17	0	≥ 3600	20
20	10.30	8.30	0	≥ 3600	20
25	656.32	49.58	1	≥ 3600	20
30	1432.06	202.17	3	≥ 3600	20
Overall	419.89	52.07	4	–	80

2.2 半导体制造场景的流水车间调度模型综述

- Three-Machine Flow Shop Scheduling with Overlapping Waiting Time Constraints (COR, 2019)
  - The first paper to addresses a three-machine flow shop scheduling problem with **overlapping waiting time constraints**, aiming to minimize makespan.
  - It develops a branch-and-bound algorithm with dominance properties and lower bounds to find optimal solutions efficiently. (up to 50 jobs)



(a) Machine configuration

(b) Schedule with overlapping waiting time constraints

Table 5: Performance of the B&B algorithm with w_{i1} in {0, 1, . . . , 100} and w_{i2} in {w_{i1}, w_{i1} + 1, . . . , 200}

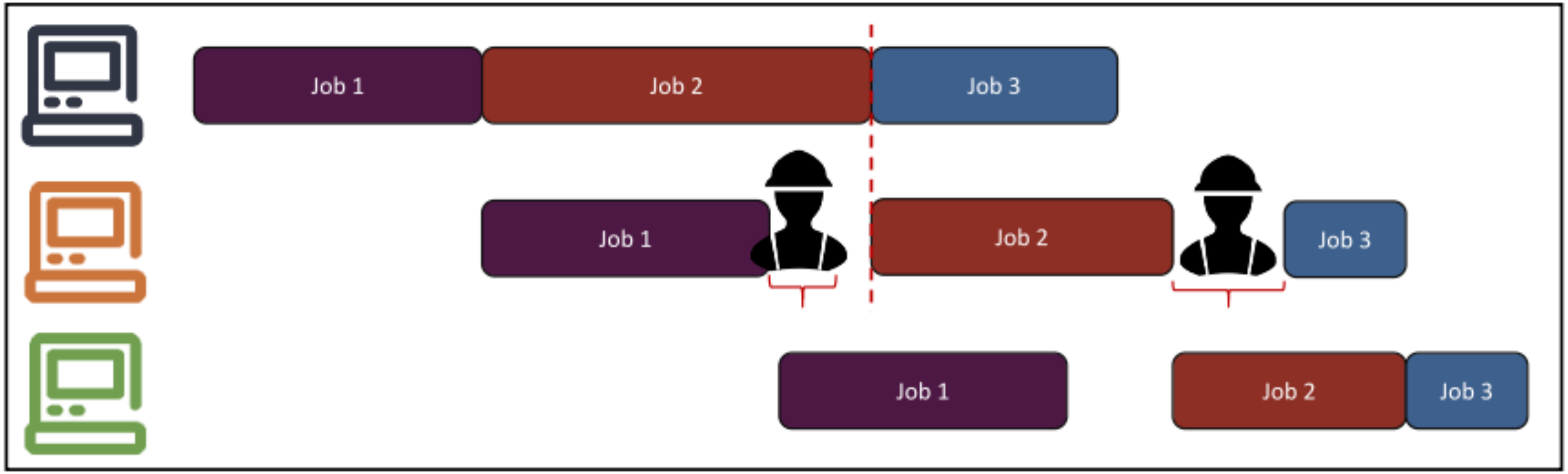
n	B&B algorithm		CPLEX	
	Average CPU time (s)	No. of instances not solved	Average CPU time (s)	No. of instances not solved
10	0.11	0	0.09	0
20	0.20	0	0.33	0
30	0.74	0	0.91	0
40	3.24	0	7.03	0
50	7.43	0	48.16	0
Overall results	2.34	0	11.30	0

Table 6: Performance of the B&B algorithm with w_{i1} in {0, 1, . . . , 50} and w_{i2} in {w_{i1}, w_{i1} + 1, . . . , 100}

n	B&B algorithm		CPLEX	
	Average CPU time (s)	No. of instances not solved	Average CPU time (s)	No. of instances not solved
10	0.10	0	0.07	0
20	0.24	0	1.01	0
30	0.67	0	255.39	1
40	2.82	0	664.77	3
50	368.24	2	1251.45	6
Overall results	74.41	2	434.54	10

3.1 问题描述

- $F_m||C_{\max}$：考虑一个 general 的流水车间调度问题，其中
  - F_m：机台数为 m 的流水车间调度问题，每个工件需依次进入各个机台进行加工
  - $C_{\max}$：目标函数是最小化 makespan，即最小化最大完工时间
  - 补充：后续在模型设定和数值试验中会考虑 w_{il} 和 s_{ij}，但是研究的重点是 general 问题的 ATIF 模型



3.2 网络图定义

- 约翰逊算法：能够最优求解两机台流水车间调度问题的一种启发式算法
  - 缺点：无法拓展到三机台；无法添加其他Q-time、setup times的设定
  - 作用：用简单算例辅助验证模型正确性

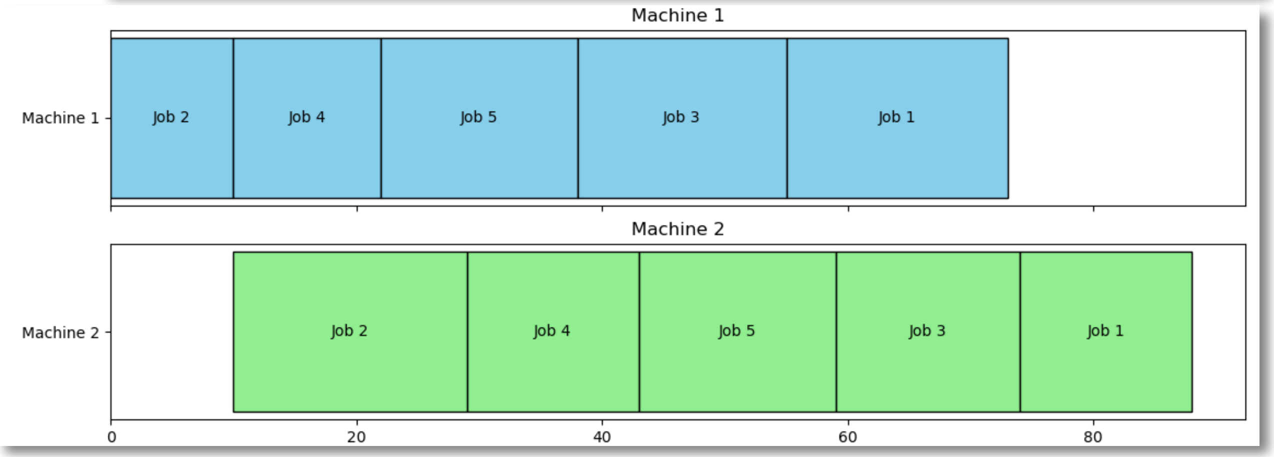
Algorithm 1 Johnson's Algorithm for Two-Machine Flow Shop Scheduling

Input: Processing times p_{ij} for each job j on machine i $(i = 1, 2)$
Output: Optimal job sequence S
Initialize an empty schedule list S
Initialize two lists: L_1 for jobs with $p_{1j} < p_{2j}$, and L_2 for jobs with $p_{1j} \geq p_{2j}$
Initialize an empty list S_1 for jobs to be scheduled first, and an empty list S_2 for jobs to be scheduled last
for each job j **do**
    if $p_{1j} < p_{2j}$ **then**
        Add job j to L_1
    else
        Add job j to L_2
    end if
end for
Sort L_1 in non-decreasing order of p_{1j}
Sort L_2 in non-increasing order of p_{2j}
for each job j in L_1 **do**
    Add job j to S_1
end for
for each job j in L_2 **do**
    Add job j to S_2
end for
Concatenate S_1 and S_2 to form the final schedule $S = S_1 \cup S_2$
return S

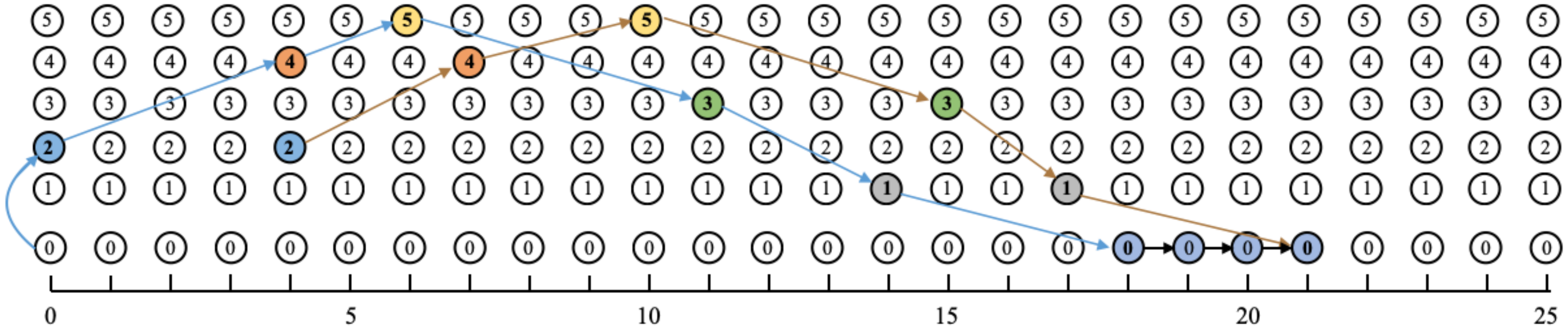
Example 3

考虑流水车间调度问题$F_2||C_{\max}$，目标函数是最大完工时间，相关参数如下表所示：

J_i	J_1	J_2	J_3	J_4	J_5
p_{i1}	18	10	17	12	16
p_{i2}	14	19	15	14	16



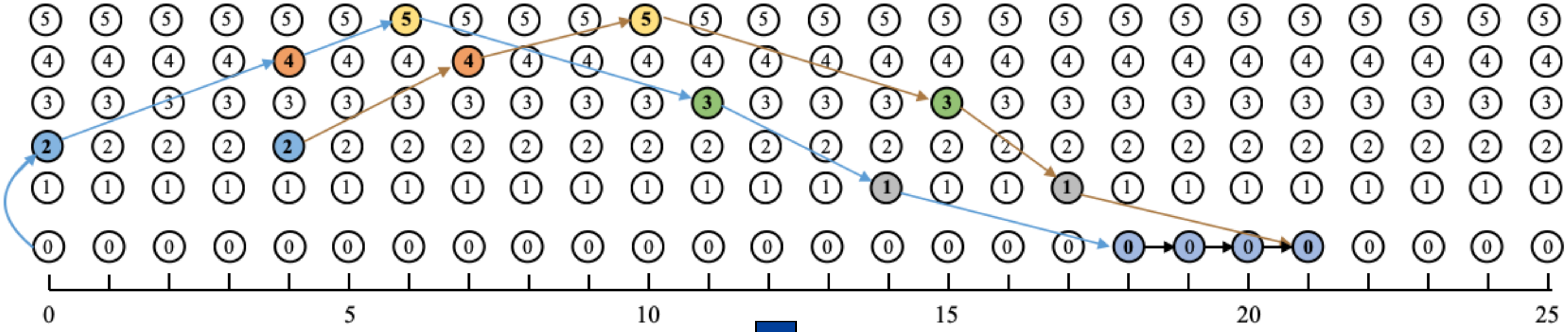
3.2 网络图定义



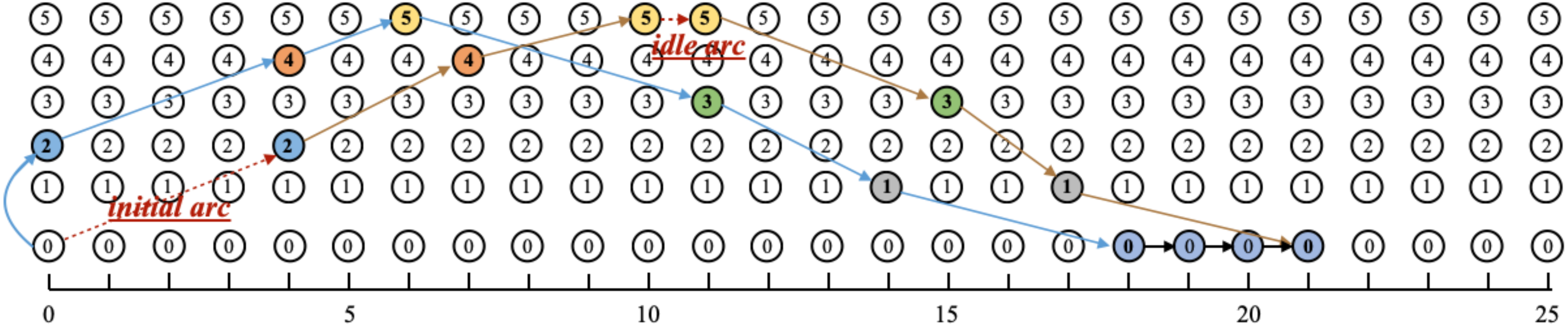
3.2 网络图定义


没有与source节点相连、违背网络图中最基本的流平衡约束
工件进行加工的先后次序有违流水车间调度中基本的次序约束
5
10
15
20
25

3.2 网络图定义

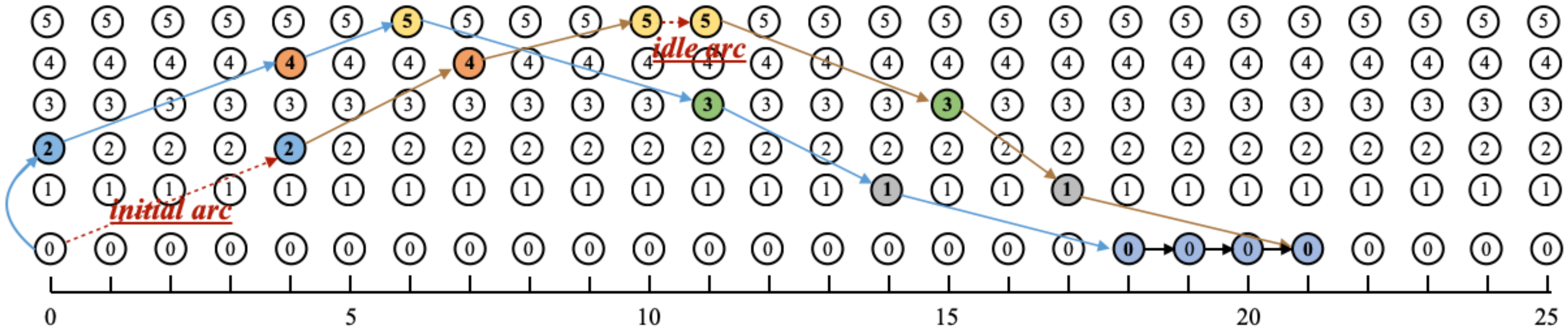


调整网络图结构



3.2 网络图定义

- 网络图的形式化定义: $G=(V,A)$，其中 $V=R\cup O, A=A_1\cup A_2\cup A_3\cup A_4$
  - $R=\{(j,t):j\in J, t=\bar{s}_j+p_j\}$
  - $O=\{(0,t):t=0,\ldots,T\}$
  - $A_1=\{((0,0),(j,t)):(0,0)\in O,(j,t)\in R\}$
  - $A_2=\{(i,t),(j,t+p_i):(i,t)\in R,(j,t+p_i)\in R\}$
  - $A_3=\{((j,t),(j,t+1)):(j,t)\in V,(j,t+1)\in V\}$
  - $A_4=\{((j,t),(0,T)):(j,t)\in R,(0,T)\in O\}$



3.3 数学模型

符号	类型	含义
x_{ij}^t	决策变量	弧 $(i,j)^t$ 是否被选择
α	决策变量	机台完成加工全部工件的时间
p_{ik}	参数	工件在机台上的加工时间
c_{ijk}	参数	可行解中工件完成加工的时间
J	集合	工件集合
P	集合	可行路径的集合

$$\min \alpha$$

$$\text{s.t.} \sum_{j\in J} x_{0j}^t = m, \quad t=0,\ldots,T-p_j$$

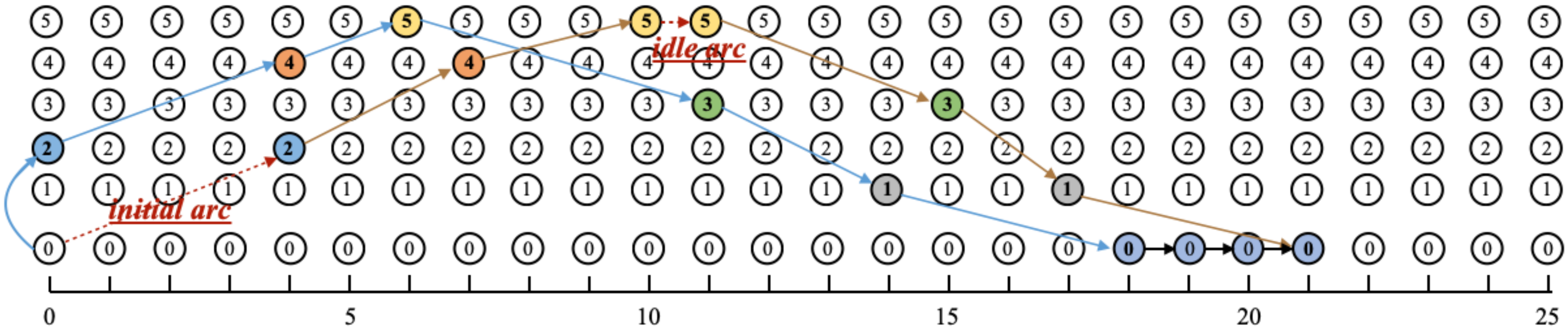
$$\sum_{i\in J_0\backslash\{j\}} \sum_{t=p_i}^{T-p_j} x_{ij}^t = m, \quad \forall j \in J$$

$$\sum_{j\in J_0\backslash\{i\}} x_{ji}^t - \sum_{j\in J_0\backslash\{i\}} x_{ij}^{t+p_i} = 0 \quad \forall i\in J, t=0,\ldots,T-p_j$$

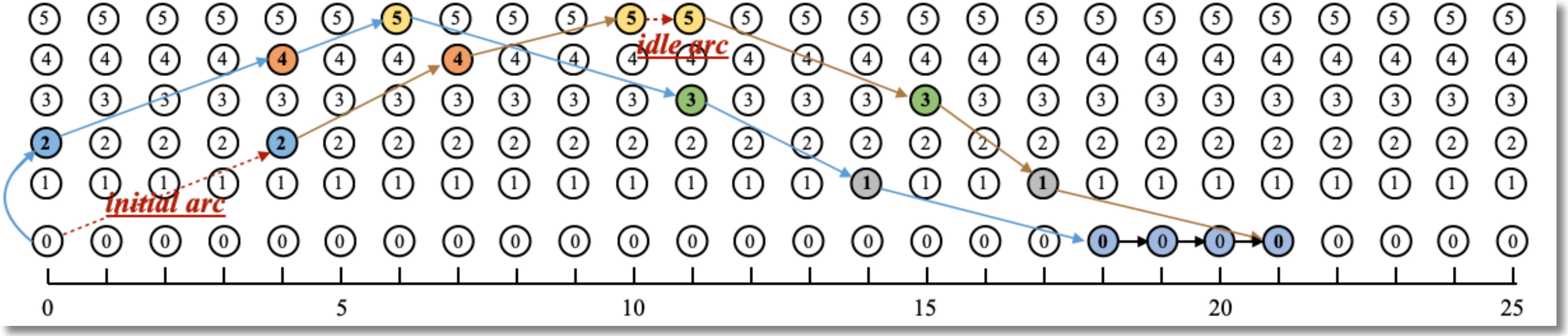
$$\sum_{j\in J_0\backslash\{i\}} x_{ji}^t - \sum_{j\in J_0\backslash\{i\}} x_{ij}^{t+1} = 0 \quad \forall i\in J, t=0,\ldots,T-1$$

$$\alpha \geq \sum_{(i,j)^t\in P} c_{ij}x_{ij}^t \quad \forall P\in\mathbb{P}$$

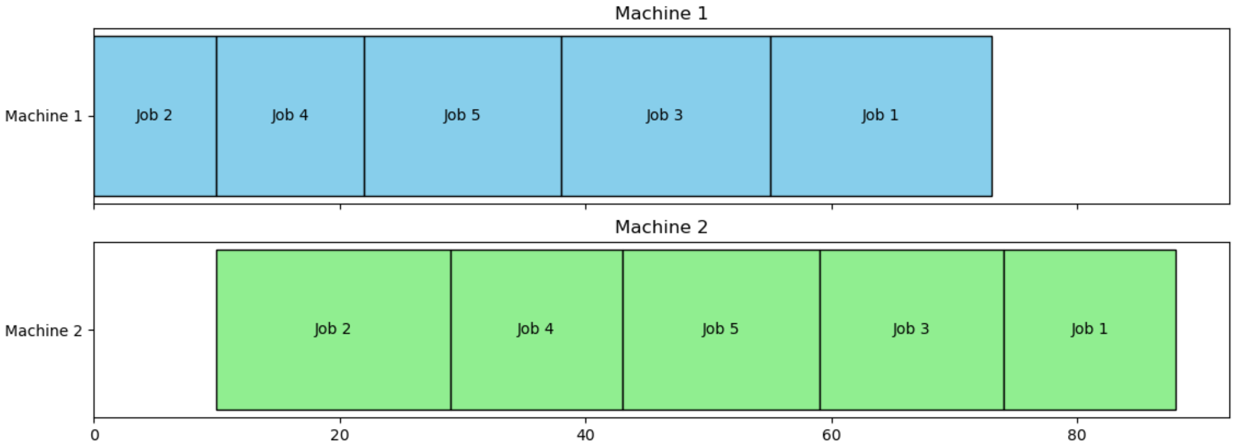
$$x_{ij}^t \in \{0,1\}, \quad \forall i\in J_0, \forall j \in J_0\backslash\{i\}, t=0,\ldots,T-1$$



3.3 数学模型



- 求解结果验证
  - ATIF模型的求解结果为:
    - $x_{0,2}^{0} = x_{2,4}^{10} = x_{4,5}^{22} = x_{5,3}^{38} = x_{3,1}^{55} = x_{1,0}^{73} = 1$
    - $x_{0,2}^{10} = x_{2,4}^{29} = x_{4,5}^{43} = x_{5,3}^{59} = x_{3,1}^{74} = x_{1,0}^{88} = 1$
  - 约翰逊算法的求解结果为:
    - optimal sequence: 2 -> 4 -> 5 -> 3 -> 1
    - optimal makespan: 88



3.4 模型优超关系

$$
\begin{aligned}
\text{(PBF)} \quad & \min\ C_{\max} \\
\text{s.t.} \quad & \sum_{i=1}^{n} x_{ir} = 1, && r = 1,\dots,n \\
& \sum_{r=1}^{n} x_{ir} = 1, && i = 1,\dots,n \\
& T_{[r]k} + \sum_{i=1}^{n} p_{ik} x_{ir} \le T_{[r+1]k}, && r = 1,\dots,n-1,\ k = 1,\dots,m \\
& T_{[r]k} + \sum_{i=1}^{n} p_{ik} x_{ir} \le T_{[r]\,k+1}, && r = 1,\dots,n,\ k = 1,\dots,m-1 \\
& x_{ir} + x_{j(r+1)} - 1 \le y_{ijr} && \forall i \ne j, r = 1,2,\dots,n-1 \\
& y_{ijr} \le x_{ij} && \forall i \ne j, r = 1,2,\dots,n-1 \\
& y_{ijr} \le x_{j(r+1)} && \forall i \ne j, r = 1,2,\dots,n-1 \\
& C_{\max} \ge T_{[n]m} + \sum_{i=1}^{n} p_{im} x_{in} \\
& x_{ir}, y_{ijr} \in \{0,1\} && \forall r = 1,\dots,n, \forall i \ne j \\
& T_{[r]k} \ge 0 && \forall r = 1,\dots,n, \forall k = 1,\dots,m
\end{aligned}
$$

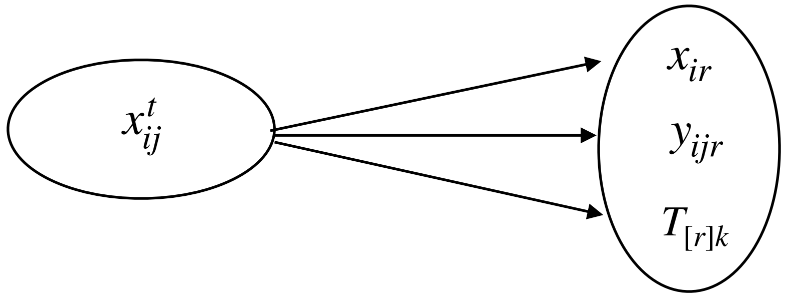
Theorem 1

The Position-Based Formulation (PBF) is dominated by the Arc-Time-Indexed Formulation (ATIF) for flow shop scheduling.

Proof:

1 ATIF is at least as good as PBF.

$$\alpha \ge \sum_{a \in A_j} c_a x_a \qquad C_{\max} \ge T_{[n]m} + \sum_{i=1}^{n} p_{im} x_{in}$$



2 ATIF can be strictly better than PBF.

ATIF的线性松弛 $\ge$ PBF的线性松弛

经验沉淀：如何证明模型更好?

- 什么是更好的模型?
  - 在整数规划下，更好的模型 (stronger formulation, even ideal formulation) 意味着该模型拥有比其他更好的连续松弛；
  - 若两个MIP的Formulation F_1 和 F_2 的整数解集相同，且 $\text{conv}(F_1) \subsetneq \text{conv}(F_2)$，则称 F_1 stronger than F_2 .
- 如何证明? （仅供参考，更多还是 case by case）
  - intuitive的思路：找到整数解集 S 👉 构造其 convex hull 👉 验证模型的线性松弛是否等于这个 convex hull
  - general的思路：$F = \{(x, z) \in \mathbb{R}^n \times \mathbb{R}^k : Ax + Cz \leq b\}$, $\text{Proj}_x(F) = \text{conv}(S)$

Example 4

考虑如下背包问题：背包容量为3，备选物品有二，其体积分别为 (2, 3)，价值分别为 (3, 4). 求能使价值最高的物品选择方式。

$$\begin{aligned} \max \quad & 3x_1 + 4x_2 \\ \text{s.t.} \quad & 2x_1 + 3x_2 \leq 3 \\ & x_1, x_2 \in \{0,1\} \end{aligned}$$

松弛解：$x_1^* = 0.5,\ x_2^* = 0.5,\ z^* = 3.5$

整数解：$x_1^* = 0,\ x_2^* = 1,\ z^* = 4$

整数可行域：$S = \{(0,0), (0,1), (1,0)\}$

$\text{conv}(S) = \{x_1 + x_2 \leq 1,\ x_1 \leq 1, x_2 \leq 1\}$

(ideal formulation)
$$\begin{aligned} \max \quad & 3x_1 + 4x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 1 \\ & 0 \leq x_1, x_2 \leq 1 \end{aligned}$$

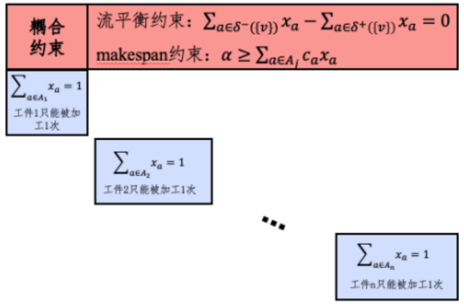
[1] Li, Y. (2025). Strong Formulations and Algorithms for Regularized A-optimal Design. *arXiv preprint arXiv:2505.14957*.
[2] Aghaei, S., Gómez, A., & Vayanos, P. (2025). Strong optimal classification trees. *Operations Research*, *73*(4), 2223-2241.
[3] Deza, A., Gómez, A., & Atamtürk, A. (2024). Fair and accurate regression: Strong formulations and algorithms. *arXiv preprint arXiv:2412.17116*.

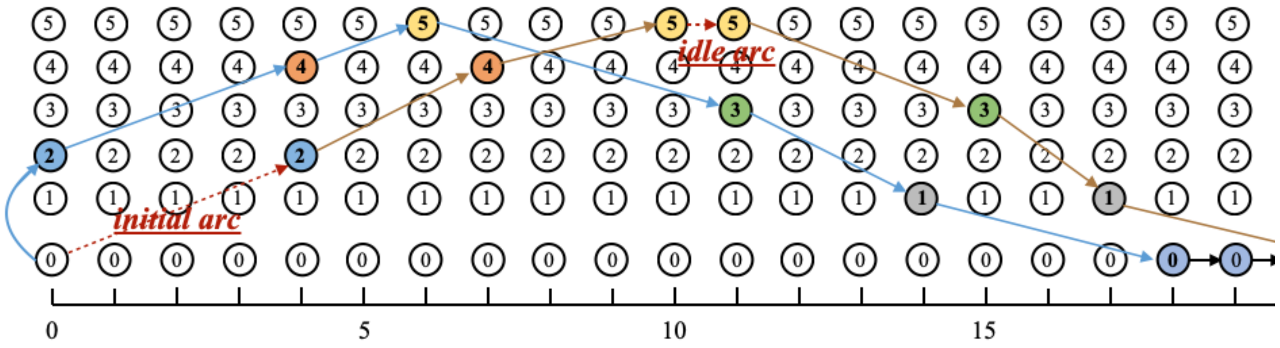
4.1 模型结构分析

- 网络流模型的分块结构特征
  - 特征：模型包含耦合约束和分块约束，每个分块之间相互独立并只包含一部分决策变量，各分块通过耦合约束与其他子问题联系在一起
  - 求解策略：Danzig-Wolfe分解适于求解该分块结构，将复杂约束与一个或多个具有易处理的特殊结构的线性约束分解开

$$
\begin{aligned}
\min\ & \alpha \\
\text{s.t.}\ & \sum_{j\in J} x_{0j}^{t} = m, && t = 0,\dots,T-p_j \\
& \sum_{i\in J_0\setminus\{j\}} \sum_{t=p_i}^{T-p_j} x_{ij}^{t} = m, && \forall j \in J \\
& \sum_{j\in J_0\setminus\{i\}} x_{ji}^{t} - \sum_{j\in J_0\setminus\{i\}} x_{ij}^{t+p_i} = 0 && \forall i \in J,\ t = 0,\dots,T-p_j \\
& \sum_{j\in J_0\setminus\{i\}} x_{ji}^{t} - \sum_{j\in J_0\setminus\{i\}} x_{ij}^{t+1} = 0 && \forall i \in J,\ t = 0,\dots,T-1 \\
& \alpha \ge \sum_{(i,j)^t\in P} c_{ij} x_{ij}^{t} && \forall P \in \mathbb{P} \\
& x_{ij}^{t} \in \{0,1\}, && \forall i \in J_0, \forall j \in J_0\setminus\{i\}, t = 0,\dots,T-1
\end{aligned}
$$

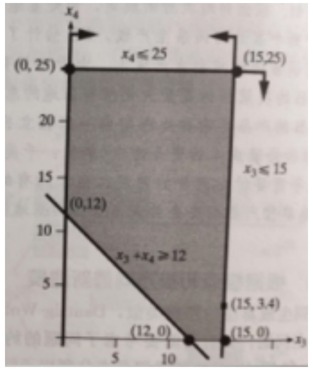
$$
Ax = \begin{bmatrix} B_0 & B_1 & B_2 & \cdots & B_K \\ & A_1 & & & \\ & & A_2 & & \\ & & & \ddots & \\ & & & & A_K \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix}
$$





4.2 模型重构与Dantzig-Wolfe分解

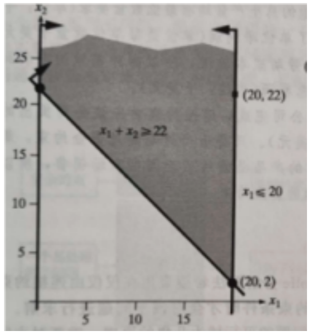
- **Danzig-Wolfe 分解算法思想**
  - Minkowski 表示：考虑线性规划问题的可行域 $P = \{x \mid Ax = b, x \geq 0\}$，可行解可以表示为极点的凸组合和极射线的非凸组合
  - 列生成算法求解：由于可行域所表示的多面体的极点和极射线难以穷尽，因此在求解中往往使用列生成来进行求解



$$x = \sum_j \lambda_j x^{(j)}$$

$$\sum_j \lambda_j = 1$$

$$\lambda_j \geq 0$$

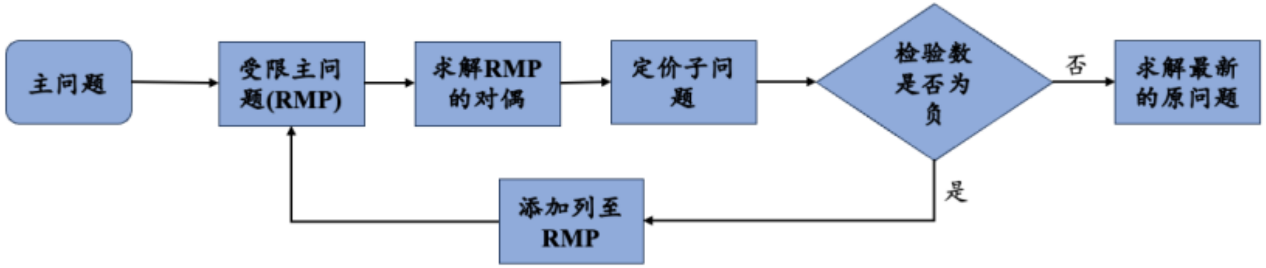


$$x = \sum_j \lambda_j x^{(j)} + \sum_i \mu_i r^{(i)}$$

$$\sum_j \lambda_j = 1$$

$$\lambda_j \geq 0$$

$$\mu_i \geq 0$$



4.2 模型重构与Dantzig–Wolfe分解

Definition 1 (Pseudo-Schedule) [Van Den Acker, 2000, IJOC]

The extreme points of polyhedron $\mathbb{P}$ represents schedules that satisfy capacity constraints but not necessarily assignment constraints. Theses schedules allow jobs to be started multiple times, once, or not at all, and are termed *pseudo-schedule*

$$x_{ij}^{t} = \sum_{p \in P} q_{ij}^{tp} \lambda_p, \quad (i,j)^t \in A$$

where λ_p indicate whether pseudo-schedule p is included in the solution.

Example: $n = 3$

λ_1 : job 1 → job 2 → job 3 → job 2

λ_2 : job 3 → job 2

4.2 模型重构与Dantzig-Wolfe分解

- **Danzig-Wolfe 分解算法思想**
  - Minkowski 表示：考虑线性规划的可行域 $P=\{x\,|\,Ax=b, x\geq 0\}$，可行解可以表示为极点的凸组合和极射线的非凸组合
    - 本研究中极点索引 p 的含义：***pseudo schedule = partial schedule + repeated schedule***
    - 使用常数集合 $\{q_{ij}^{tp}\in\{0,1\}:(i,j)^t\in A\}$ 表示是否弧 $(i,j)^t$ 在路径 p 中出现
  - 列生成算法求解：由于可行域所表示的多面体的极点和极射线难以穷尽，因此在求解中往往使用列生成来进行求解

$$
\begin{aligned}
\text{(ATIF)}\quad & \min\ \alpha \\
\text{s.t.}\quad & \sum_{j\in J} x_{0j}^t = m, && t=0,\dots,T-p_j \\
& \sum_{i\in J_0\setminus\{j\}}\ \sum_{t=p_i}^{T-p_j} x_{ij}^t = m, && \forall j\in J \\
& \sum_{j\in J_0\setminus\{i\}} x_{ji}^t - \sum_{j\in J_0\setminus\{i\}} x_{ij}^{t+p_i} = 0 && \forall i\in J,\ t=0,\dots,T-p_j \\
& \sum_{j\in J_0\setminus\{i\}} x_{ji}^t - \sum_{j\in J_0\setminus\{i\}} x_{ij}^{t+1} = 0 && \forall i\in J,\ t=0,\dots,T-1 \\
& \alpha \geq \sum_{(i,j)^t\in p} c_{ij}x_{ij}^t && \forall p\in P \\
& x_{ij}^t\in\{0,1\}, && \forall i\in J_0, \forall j\in J_0\setminus\{i\}, t=0,\dots,T-1
\end{aligned}
$$

耦合约束（前两个约束）

分块约束（第三、四个约束）

- 决策变量为 λ_p，表示路径 p 是否在解中出现

$$x_{ij}^t=\sum_{p\in P} q_{ij}^{tp}\lambda_p,\ (i,j)^t\in A$$

$$
\begin{aligned}
\text{(DWM)}\quad & \min\ \alpha \\
\text{s.t.}\quad & \sum_{p\in P}\Big(\sum_{(0,j)^t\in A} q_{0j}^{tp}\Big)\lambda_p = m, && t=0,\dots,T-p_j \\
& \sum_{p\in P}\Big(\sum_{(i,j)^t\in A} q_{ij}^{tp}\Big)\lambda_p = m, && \forall j\in J \\
& \alpha \geq \sum_{p\in P}\Big(\sum_{(i,j)^t\in p} c_{ij}q_{ij}^{tp}\Big)\lambda_p \\
& \lambda_p\geq 0, && \forall p\in P
\end{aligned}
$$

4.2 模型重构与Dantzig-Wolfe分解

- **Danzig-Wolfe 分解算法思想**
  - Minkowski 表示：考虑线性规划的可行域 $P=\{x\,|\,Ax=b, x\geq 0\}$，可行解可以表示为极点的凸组合和极射线的非凸组合
    - 本研究中极点索引 p 的含义：***pseudo schedule = partial schedule + repeated schedule***
    - 使用常数集合 $\{q_{ij}^{tp}\in\{0,1\}:(i,j)^t\in A\}$ 表示是否弧 $(i,j)^t$ 在路径 p 中出现
  - 列生成算法求解：由于可行域所表示的多面体的极点和极射线难以穷尽，因此在求解中往往使用列生成来进行求解

$$
\begin{aligned}
\text{(DWM)}\quad & \min\ \alpha \\
\text{s.t.}\quad & \sum_{p\in P}\Big(\sum_{(0,j)^t\in A} q_{0j}^{tp}\Big)\lambda_p = m, && t=0,\dots,T-p_j \\
& \sum_{p\in P}\Big(\sum_{(i,j)^t\in A} q_{ij}^{tp}\Big)\lambda_p = m, && \forall j\in J \\
& \alpha \geq \sum_{p\in P}\Big(\sum_{(i,j)^t\in p} c_{ij}q_{ij}^{tp}\Big)\lambda_p \\
& \lambda_p \geq 0, && \forall p\in P
\end{aligned}
$$

dual

$$
\begin{aligned}
\text{(SP)}\quad & \min\ \bar{c}_{ij} = \alpha - \sum_{l=1}^{m}\beta_{ijl}^{t}\pi_l \\
& \text{s.t. } \textit{block constraint}
\end{aligned}
$$

column

4.3 算法流程总结

Algorithm 1 DW Decomposition and Branch-and-Price

```
Input: Flow shop scheduling problem instance
Output: Optimal schedule
Initialize the arc-time-indexed formulation (ATIF) for the problem
Reformulate ATIF using Dantzig-Wolfe decomposition to obtain the master
problem (DWM) and pricing subproblem (SP)
Initialize the restricted master problem (RDWM) with a subset of columns
Solve the linear relaxation of the RMP to obtain an initial dual solution
Branch-and-Price:
Initialize the search tree with the root node
while the search tree is not empty do
    Select a node from the search tree
    if the node is infeasible then
        Prune the node and continue
    end if
    Initialize the RMP for the current node with a subset of columns
    Column Generation:
        repeat
            Solve the pricing subproblem (SP) to find new columns with
negative reduced costs
            if new columns are found then
                Add the new columns to the RDWM for the current node
                Update the dual solution for the current node
            else
                Terminate column generation for the current node
            end if
        until no new columns are found or the solution is integer
    if the solution for the current node is integer then
            Update the best integer solution
            Prune the node
    else
            Choose a branching variable $\lambda_p$ and create two child nodes
            Add the child nodes to the search tree
    end if
end while
return the optimal schedule
```

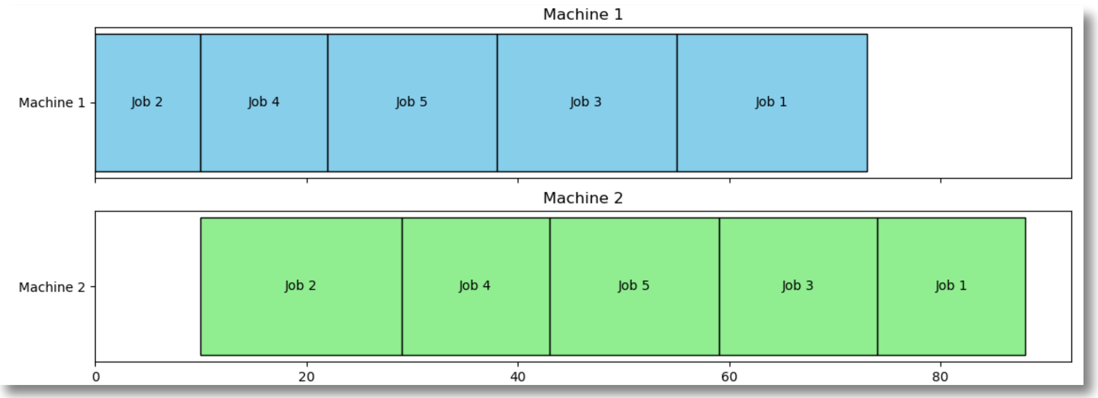
4.4 补充算法：适用于PBF的B&B

- 节点表示：树中的每一个节点对应一个 partial schedule, 剩余工件集合视作待扩展分支;
- 分支方式：从当前部分序列中，尝试将每一个未排工件依次追加到序列末尾，生成子节点；
- 搜索策略：使用深度优先策略，优先扩展已排工件数较多、下界较小的节点，希望能够更快收敛。

Example 3

考虑流水车间调度问题$F_2||C_{\max}$，目标函数是最大完工时间，相关参数如下表所示：

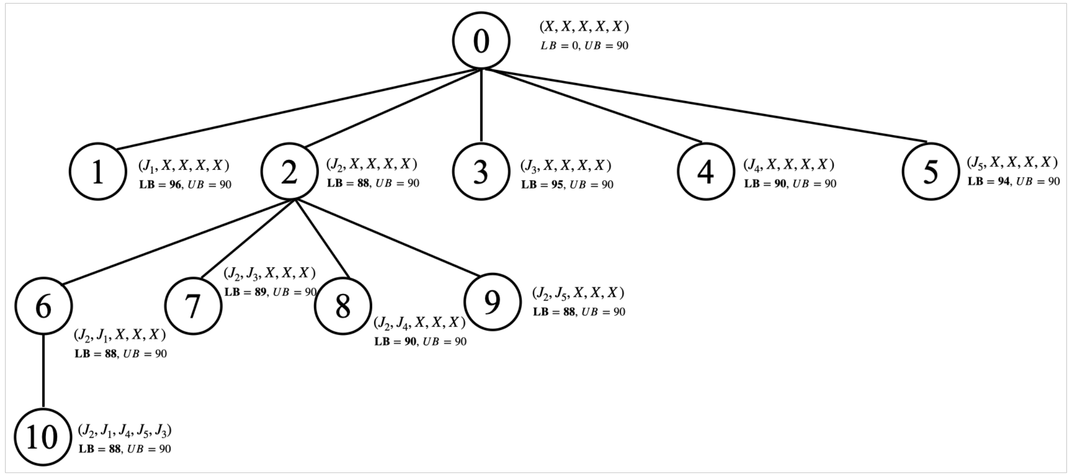
J_i	J_1	J_2	J_3	J_4	J_5
p_{i1}	18	10	17	12	16
p_{i2}	14	19	15	14	16



step 1: 启发式初始解直接作为上界

$$\sigma_0 = J_1 \to J_2 \to J_3 \to J_4 \to J_5, \quad Z_0 = 90$$

step 2: 通过分支定界树的方式不断构造 partial schedule σ_i



4.5 再补充三个启发式算法

- 改进的约翰逊算法（**Campbell–Dudek–Smith Algorithm** 的思想）
  - 将 m 个机台的 **Flowshop** 转换为 $m-1$ 个等效两机问题，对每个问题应用 **Johnson** 法得到候选调度序列，再从中选择使 **makespan** 最小者作为最终调度序列。

Algorithm 1: Improved Johnson's Algorithm (CDS Method)

Input: n jobs, m machines; processing times $p_{i,k}$ for job i on machine k
Output: A job sequence π

```
BestSequence ← ∅, BestC_max ← +∞;
for k = 1 to m − 1 do
    for each job i do
        P1_i ← Σ_{t=1}^{k} p_{i,t};
        P2_i ← Σ_{t=k+1}^{m} p_{i,t};
    end
    π^(k) ← JohnsonSort(P1, P2);
    Compute C_max^(k) for π^(k) on the real m-machine system;
    if C_max^(k) < BestC_max then
        BestC_max ← C_max^(k);
        BestSequence ← π^(k);
    end
end
return BestSequence;
```

Example 4

考虑流水车间调度问题$F_2||C_{\max}$，目标函数是最大完工时间，相关参数如下表所示：

J_i	J_1	J_2	J_3	J_4	J_5
p_{i1}	18	10	17	12	16
p_{i2}	14	19	15	14	16
p_{i3}	11	5	19	12	4

1. 两机台问题1：合并前两个的机台的加工时间，使用约翰逊算法，得到最优序列1
2. 两机台问题2：合并后两个机台的加工时间，使用约翰逊算法，得到最优序列2
3. 将二者比较，取加工时间最短的最优序列，作为最终的结果

4.5 再补充三个启发式算法

- 改进的NEH算法：
  - 目前而言 m 机台流水车间最小化 **makespan** 中表现最优秀、最被广泛认可的启发式算法；
  - 首先根据工件在各加工阶段的总负载对工件进行降序排序，然后采用递增构造策略，在每次插入新工件时，枚举所有可能插入位置，选择能产生最小 makespan 的插入位置，从而逐步构造高质量的可行调度方案。

Algorithm 1: Improved NEH Algorithm for m-Machine Flowshop

```
Input: n jobs, m machines; processing times p_{i,k}
Output: A job sequence S
for each job i do
  | T_i ← Σ_{k=1}^{m} p_{i,k}
end
Sort jobs in non-increasing order of T_i to obtain sequence π = (i_1, i_2, ..., i_n);
S ← (i_1);
for j = 2 to n do
    best_Cmax ← +∞;
    for pos = 1 to |S| + 1 do
        S' ← Insert i_j into position pos of S;
        Cmax ← ComputeMakespanWithConstraints(S', p);
        if Cmax < best_Cmax then
            best_Cmax ← Cmax;
            best_sequence ← S';
        end
    end
    S ← best_sequence;
end
return S;
```

Example 4

考虑流水车间调度问题$F_2||C_{\max}$，目标函数是最大完工时间，相关参数如下表所示：

J_i	J_1	J_2	J_3	J_4	J_5
p_{i1}	18	10	17	12	16
p_{i2}	14	19	15	14	16
p_{i3}	11	5	19	12	4

总加工时间：$T_1 = 43, T_2 = 34, T_3 = 51, T_4 = 38, T_5 = 36$

按照降序排序：$J_3 \to J_1 \to J_4 \to J_5 \to J_2$

逐个构造序列与评估：$J_3 \to J_1$ or $J_1 \to J_3$

$J_1 \to J_3 \to J_2$ or $J_1 \to J_2 \to J_3$ or $J_2 \to J_1 \to J_3$

4.5 再补充三个启发式算法

- 贪心算法（前两个算法是现成的，这个是本研究提出的）：
  - 在构造加工顺序时，始终选择那个放在当前序列末尾时对完成时间增加最小的 job；
  - 通过最小化相邻工件之间的“**增量代价**”，使流水系统尽量保持连续加工，从而降低整体 makespan。

Algorithm 1: Greedy Algorithm

Input: n jobs, m machines; processing times $p_{i,k}$
Output: A job sequence S

```
S ← (),   U ← {1, 2, ..., n};
i* ← arg max_{i∈U} Σ_{k=1}^{m} p_{i,k};
S ← (i*),   U ← U \ {i*};
while U ≠ ∅ do
    best ← +∞,   j* ← null;
    foreach j ∈ U do
        Δ ← IncrementalCostApprox(S, j, p);
        if Δ < best then
            best ← Δ,   j* ← j;
        end
    end
    S ← S ⊕ j*,   U ← U \ {j*};
end
return S;
```

增量代价的计算公式：

- 基本公式：$\mathrm{Cost}(u \to v) = C_{\max}(u, v) - C_{\max}(u)$
- 近似公式：$\mathrm{Cost}(u \to v) = \sum_{k=1}^{m} \max(0,\ p_{v,k} - p_{u,k})$

数值实例：

- 总加工时间：$T_1 = 43, T_2 = 34, T_3 = 51, T_4 = 38, T_5 = 36$，因此先排 J_3
- 计算评估增量代价：
  - $\Delta(J_3 \to J_1) = \max(18-17,0) + \max(14-15,0) + \max(11-19,0) = 1$
  - $\Delta(J_3 \to J_2) = \max(10-17,0) + \max(19-15,0) + \max(5-19,0) = 4$
  - $\Delta(J_3 \to J_4) = \max(12-17,0) + \max(14-15,0) + \max(12-19,0) = 0$ ✅
  - $\Delta(J_3 \to J_5) = \max(16-17,0) + \max(16-15,0) + \max(4-19,0) = 1$

5.1 先比较一下启发式算法

Table 4 Results of the test on the effectiveness of the heuristics

	MJ[1]			MG[2]			MN[3]			ARCPUT[4]	
(n, m)	Avg gap[5] (%)	Avg time (s)	NBS[6]	Avg gap (%)	Avg time (s)	NBS	Avg gap (%)	Avg time (s)	NBS	$T_{\mathrm{MG}}/T_{\mathrm{MJ}}$	$T_{\mathrm{MN}}/T_{\mathrm{MJ}}$
(10,3)	0.15	0.01	3	0.02	0.00	16	0.10	0.00	1	0.00	0.00
(10,5)	0.55	0.98	4	0.83	1.39	12	0.48	0.80	4	1.42	0.82
(20,3)	4.73	1.47	1	2.95	0.95	10	3.43	1.12	9	0.64	0.76
(20,5)	10.21	14.33	0	2.67	3.00	12	4.55	3.42	8	0.21	0.32
(30,3)	17.14	4.41	0	2.41	1.91	13	4.12	3.44	7	0.43	0.78
(30,5)	13.59	10.05	0	3.01	6.94	12	4.01	4.09	8	0.69	0.41
(40,3)	19.88	6.09	0	2.09	4.38	10	2.62	6.58	10	0.72	1.08
(40,5)	17.35	32.14	0	2.98	10.45	11	4.10	14.76	9	0.33	0.46
(50,3)	19.02	21.65	0	2.65	7.87	7	2.44	10.12	13	0.36	0.47
(50,5)	17.30	52.00	0	3.41	16.39	9	2.01	33.79	11	0.32	0.65
Overall	11.99	-	8	2.30	-	112	2.79	-	80	-	-

Note: There are 20 instances for each group size. Since the Johnson's algorithm can find the optimal solution for the flow-shop scheduling problem with 2 machines, this table only includes instances with 3 and 5 machines.

[1]Modified Johnson's algorithm.

[2]Modified Greedy algorithm.

[3]Modified NEH algorithm.

[4]Average ratio of CPU times.

[5]Since the optimal solutions of the synthesized instances are unknown, the gap here is calculated by taking the best solution among the three heuristic algorithms as the benchmark, and then measuring the gap of other heuristic solutions relative to this solution, i.e., gap = $(T_{MJ} - \min\{T_{MJ}, T_{MG}, T_{MN}\})/T_{MJ}$, where the notation T denotes the completion time obtained by the current algorithm. The calculation method for the other two is the same.

[6]Number of instances (out of 20 instances) for which the algorithm found the best solution among the three heuristic algorithms.

5.2 再比较一下模型直接用求解器求解

Table 5 Results of the test on model strength between Arc-Time-Indexed Formulation (ATIF) and Position-Based Formulation (PBF)

	PBF-G[1]			ATIF-G[2]		
(n, m)	Avg gap[3]	Avg time[4]	Max time	Avg gap	Avg time	Max time
(10,2)	0.00	0.00	0.00	0.00	0.01	0.03
(10,3)	0.00	0.07	0.13	0.00	0.06	0.13
(10,5)	0.00	23.31	34.11	0.00	3.17	4.12
(13,2)	0.00	4.72	6.29	0.00	0.01	0.03
(13,3)	0.00	9.45	13.33	0.00	4.42	6.13
(13,5)	0.00	179.87	298.54	0.00	19.68	24.39
(15,2)	0.00	5.39	6.21	0.00	1.17	1.89
(15,3)	0.00	19.77	23.02	0.00	6.82	10.30
(15,5)	0.00	787.10	1311.00	0.00	122.09	168.09
(18,2)	0.00	7.22	10.09	0.00	2.57	4.33
(18,3)	0.00	69.86	97.85	0.00	27.65	35.58
(18,5)	4.98	≥3600	≥3600	0.00	675.32	987.89
(20,2)	0.00	11.39	14.21	0.00	4.41	6.55
(20,3)	0.00	197.03	231.49	0.00	89.43	126.93
(20,5)	12.07	≥3600	≥3600	0.00	1198.79	1655.71

Note: There are 20 instances for each group size.

[1]Solve the PBF model directly by using the Gurobi solver.

[2]Solve the ATIF model directly by using the Gurobi solver.

[3]Since every instance can be solved to optimality by directly invoking the solver on the ATIF formulation, the optimality gap of the PBF model returned by the solver is computed as gap $= (T - T_{\text{opt}})/T$, where the notation T denotes the completion time obtained by the current algorithm.

[4]Average CPU time (or its lower bound), obtained by assigning a value of 3600s to any instance not solved to optimality within that limit.

5.3 还可以比较一下精确算法

	PBF-BB[1]				ATIF-BP[2]			
(n, m)	Avg gap[3] (%)	Avg time[4] (s)	Max time (s)	Unsolved[5]	Avg gap (%)	Avg time (s)	Max time (s)	Unsolved
(10,2)	0.00	0.00	0.00	0	0.00	0.00	0.00	0
(10,3)	0.00	0.02	0.03	0	0.00	0.00	0.01	0
(10,5)	0.00	3.19	5.00	0	0.00	0.67	0.77	0
(15,2)	0.00	0.94	1.13	0	0.00	0.01	0.02	0
(15,3)	0.00	1.57	1.99	0	0.00	0.01	0.02	0
(15,5)	0.00	17.20	19.24	0	0.00	1.32	1.57	0
(20,2)	0.00	1.38	1.67	0	0.00	0.11	0.16	0
(20,3)	0.00	7.80	10.02	0	0.00	1.16	1.57	0
(20,5)	0.00	98.10	134.99	0	0.00	2.78	3.02	0
(25,2)	0.00	8.22	11.21	0	0.00	0.65	0.98	0
(25,3)	0.00	39.92	48.67	0	0.00	9.25	10.04	0
(25,5)	0.00	249.88	288.75	0	0.00	15.21	17.92	0
(30,2)	0.00	16.90	18.90	0	0.00	1.99	3.02	0
(30,3)	0.00	127.03	155.09	0	0.00	10.12	13.34	0
(30,5)	1.19	1906.58	2328.96	2	0.00	24.33	30.53	0
(35,2)	0.00	21.30	23.49	0	0.00	4.88	5.01	0
(35,3)	0.00	472.03	500.00	0	0.00	37.09	43.01	0
(35,5)	9.89	$\geq$3600	$\geq$ 3600	9	0.00	68.35	81.09	0
(40,2)	0.00	43.21	49.11	0	0.00	11.12	15.58	0
(40,3)	8.91	2662.76	$\geq$ 3600	6	0.00	96.32	133.65	0
(40,5)	15.34	$\geq$3600	$\geq$ 3600	11	0.00	128.09	142.20	0
(45,2)	0.00	139.56	176.89	0	0.00	16.59	21.54	0
(45,3)	23.75	$\geq$ 3600	$\geq$ 3600	17	0.00	232.55	265.22	0
(45,5)	26.65	$\geq$3600	$\geq$ 3600	18	0.00	657.76	743.09	0
(50,2)	0.00	651.31	883.65	0	0.00	26.11	31.00	0
(50,3)	39.98	$\geq$ 3600	$\geq$ 3600	20	0.00	1112.92	1410.83	0
(50,5)	40.05	$\geq$3600	$\geq$ 3600	20	13.22	$\geq$ 3600	$\geq$ 3600	9

Note: There are 20 instances for each group size.

5.4 最后比较一下比较抽象的数据

		PBF-BB[1]				ATIF-BP[2]			
(n, m)	PDS[3]	Avg gap[4]	Avg time[5]	CVG[6]	CVT[7]	Avg gap	Avg time	CVG	CVT
(10,3)	UNI	0.00	0.03	0.00	0.11	0.00	0.00	0.00	0.01
(10,3)	MIX	0.00	0.02	0.00	0.12	0.00	0.09	0.00	0.02
(10,3)	FIX	0.00	0.03	0.00	0.09	0.00	0.06	0.00	0.01
(10,5)	UNI	0.00	3.77	0.00	0.12	0.00	0.62	0.00	0.07
(10,5)	MIX	0.00	3.98	0.00	0.12	0.00	1.45	0.00	0.08
(10,5)	FIX	0.00	4.01	0.00	0.13	0.00	1.89	0.00	0.08
(15,3)	UNI	0.00	1.06	0.00	0.63	0.00	0.01	0.00	0.10
(15,3)	MIX	0.00	1.65	0.00	0.21	0.00	0.22	0.00	0.19
(15,3)	FIX	0.00	1.62	0.00	0.22	0.00	0.81	0.00	0.16
(15,5)	UNI	1.00	18.05	0.05	0.10	0.00	1.58	0.00	0.12
(15,5)	MIX	3.32	23.74	0.39	0.25	0.00	2.31	0.00	0.10
(15,5)	FIX	4.54	26.65	0.27	0.71	0.00	3.02	0.00	0.19
(20,3)	UNI	0.00	7.42	0.00	0.15	0.00	1.17	0.00	0.15
(20,3)	MIX	0.00	7.88	0.00	0.20	0.00	1.99	0.00	0.13
(20,3)	FIX	0.00	8.94	0.00	0.28	0.00	1.85	0.00	0.17
(20,5)	UNI	17.44	$\geq$30.00	0.61	0.30	0.00	2.61	0.00	0.18
(20,5)	MIX	16.53	$\geq$30.00	0.57	0.69	0.00	6.45	0.00	0.20
(20,5)	FIX	17.92	$\geq$30.00	0.62	0.24	0.00	5.04	0.00	0.18

Note: There are 20 instances for each group size. The upper limit of the solution time is set at 30 seconds.

[1]Solve the PBF model by branch-and-bound algorithm. Initial solutions are obtained via the modified greedy algorithm.

[2]Solve the ATIF model by branch-and-price algorithm. Initial solutions are obtained via the Dijkstra's algorithm in the network graph.

[3]The processing time distribution pattern. UNI for purely uniform processing times in [5,10]; MIX for a mixed distribution with 90% in U[5,10] and 10% in U[50,60]; FIX for a fixed bottleneck machine where processing times are in U[50,60] and others in U[5,10].

[4]The gaps of both algorithms are uniformly measured by the relative distance between their upper and lower bounds, i.e., gap = (UB – LB)/UB, where the notation UB denotes upper bound, and the notation LB denotes lower bound.

[5]Average CPU time (or its lower bound), obtained by assigning a value of 3600s to any instance not solved to optimality within that limit.

[6]Coefficient of Variation of the optimality Gap.

[7]Coefficient of Variation of the solve Time.

感谢聆听！