调度问题

(Scheduling Problems)

东北大学 系统工程研究所

2009.09

调度问题

- 一类典型的优化问题。
- 广义地讲，调度问题考虑的是：
  - 随着时间的变化，如何调度有限的资源在执行任务的同时满足特定约束。
  - 资源可能在本质上是很不相同的：
    - 人力、
    - 金钱、
    - 机器、
    - 工具、
    - 材料、
    - 能源等等。
  - 任务也可以有不同的解释，从制造系统的机器划分到计算机系统的信息处理。一项任务通常可以用下面的因素来表示特征：
    - 完成时间、
    - 预期时间、
    - 相对紧急权重、
    - 处理时间
    - 资源消耗。
  - 同时，一组反映任务之间先后约束的结构可以用不同的方式定义。另外还可以考虑度量调度性能好坏的不同判据.

特点

- 调度问题几乎在现实环境(特别是工业工程领域)中无处不在。
- 许多制造工业提出的调度问题从本质上讲非常复杂，难以用传统优化方法求解。
- 通常这些难于求解的问题都表示为满足非常复杂约束的组合优化问题。
- 这些问题带有有限数量的可行解。
- 这些问题属于**NP**—难的问题。

经典调度问题的分类

- 流水车间调度问题
- 作业车间调度问题
- 机器调度问题

- 扩展调度问题：
  - 群体作业调度
  - 资源约束的项目调度
  - 多处理器调度
  - 车辆与路径调度
  - ……

制造业生产模式

- 按生产计划方式分类
  - 面向订单生产，
    - 强调准时高效，客户订单到达后才开始组织生产。
  - 面向库存生产，
    - 在客户订单到达之前就已开始生产，生产计划以库存为基础。
  - 混合生产模式，
    - 一方面根据预测，保留较大的库存，另一方面以一定的实时生产能力来满足高度客户化的订单需求。

制造业生产模式

- 按生产过程的工艺流程特征分类
  - 离散式生产：
    - 产品是由离散的零部件装配而成，零部件以各自的工艺过程通过各个生产环节，物料运动呈离散状态。
    - 例如属于生产资料生产的机械、电子设备制造业，属于生活资料生产的机电整合消费产品制造业。
    - 离散型制造企业的生产方式多为单件、小批量、多品种
  - 流程式生产：
    - 在生产过程中，物料均匀、连续地按一定工艺顺序运动，生产流程具有连续性的特点和要求
    - 例如冶炼、化工、酿造等
  - 混合流程式生产：
    - 这是一种既具有流程式生产特征，又具有离散式生产特征的复杂生产方式，其生产过程并不完全是一个自动生产线。其典型特征是生产分阶段进行，设备按阶段使用，在不同的生产阶段遵循不同的生产方式，一般产品不可数，加工过程是间歇式的。

车间调度问题的几个特点

- **Flaw Shop和Job Shop调度问题是最典型和最重要的两种车间调度问题。**
- 车间调度问题具有以下几个特点：
  - 复杂性：
    - 由于生产车间中工件、机器、缓存及搬运系统之间相互影响、相互作用、每个作业又要考虑它的加工时间、操作顺序、交货期等，因而相当复杂。
  - 动态随机性：
    - 在实际的生产调度系统中存在很多随机的和不确定的因素，比如作业到达时间的不确定性、设备的损坏/修复、作业交货期的改变、紧急定单等。
  - 多目标性：
    - 实际的计划调度往往是多目标的。生产调度的性能指标可以是成本最低、库存费最少、生产周期最短、生产切换最少、设备利用率最高、最短的延迟，最小的提前或者拖期惩罚等。这种多目标性导致调度的复杂性和计算量急剧增加。
  - 多约束性：
    - 生产车间中资源的数量、缓存的数量、工件的加工时间和加工顺序都是约束。此外还有一些人为的约束，如要求各机器上的负荷平衡等等。

流水车间调度问题

(Flow-shop Scheduling Problems)

问题描述

- 一般描述
  - n 个工件要在 m 台机器上加工，
  - 每个工件需要经过 m 道工序，每道工序要求不同的机器。
  - n 个工件在 m 台机器上的加工顺序相同。
  - 工件 i 在机器 j 上的加工时间是给定的，设为 t_{ij} $(i=1,\ldots,n; j=1,\ldots, m)$。
- 问题的目标
  - 求 n 个工件在每台机器上最优的加工顺序，使最大流程时间达到最小。

问题描述

- 对该问题常常作如下假设:
  - 每个工件在机器上的加工顺序是: **1,2,…,*m***;
  - 每台机器同时只能加工一个工件;
  - 一个工件不能同时在不同的机器上加工;
  - 工序不能预订;
  - 工序的准备时间与顺序无关，且包含在加工时间中;
  - 工件在每台机器上的加工顺序相同，且是确定的。

三类FSP问题

- 确定型流水车间问题
  - 假定工件的加工时间是已知的确定量
- 随机型流水车间问题
  - 加工时间按照一定的概率分布而变化
- 模糊型流水车间问题
  - 每个工件的模糊交货期表示为决策者对工件完成时间的满意度

实际调度问题

- 实际调度问题往往更加复杂，例如：
- 通常要满足一定的约束条件
  - 顾客的交货时间、
  - 资源约束、
  - 工艺约束等
- 性能指标通常可以分成多种类型
  - 最大能力指标
    - 最大生产率、最短生产周期等
  - 成本指标
    - 最大利润、最小化运行费用、最小投资、最大收益等
  - 客户满意度指标
    - 最短的延迟，最小提前或者拖后惩罚等

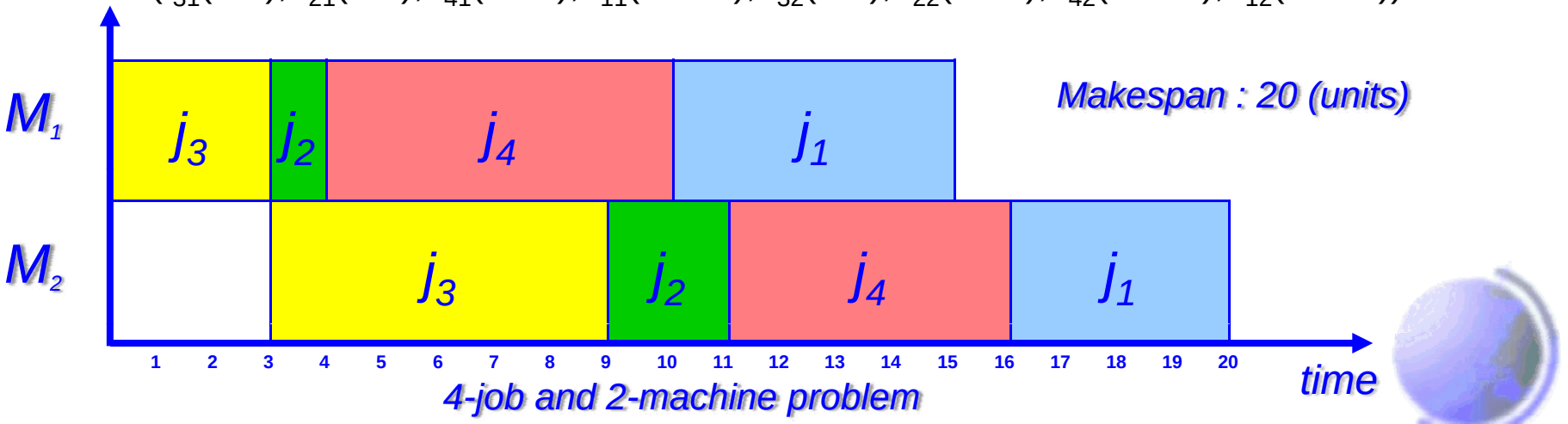
实例

- 确定型FSP的一个实例：4个工件、2台机器
  - 工件顺序：j_3, j_2, j_4, j_1.

Job i	1	2	3	4
t_{i1}	**5**	**1**	**3**	**6**
t_{i2}	**4**	**2**	**6**	**5**

  - 调度S如下：

S = (t_{31}(0-3), t_{21}(3-4), t_{41}(4-10), t_{11}(10-15), t_{32}(3-9), t_{22}(9-11), t_{42}(11-16), t_{12}(16-20))



应用

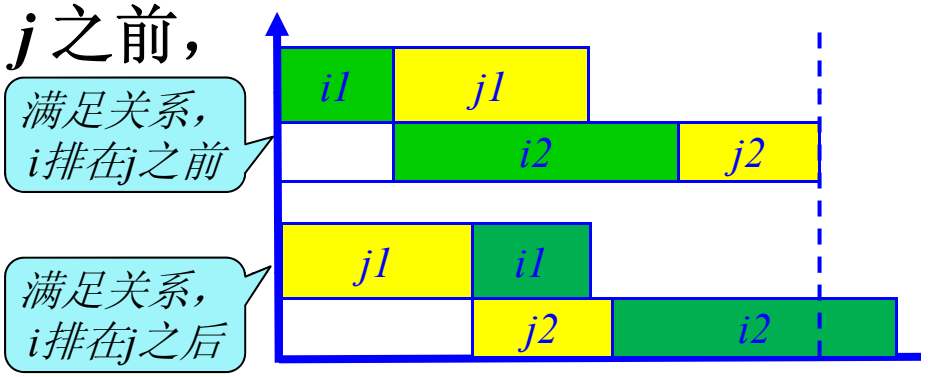
- 墙纸行业生产调度问题
- 装饰布企业车间生产调度
- 飞机维修调度问题
- 化工涂料生产的配料及调度
- 集装箱码头装卸作业的调度问题
- 汽车总装车间调度

- 柔性流水车间调度问题
  - 冷轧厂热处理车间冷卷热处理生产调度问题
  - 炼钢-连铸浇次组合与排序
  - 石油、化工等流程工业

两台机器FSP的最优调度

- **Johnson规则：**
  - 最优调度中工件 *i* 排在工件 *j* 之前，
  - 如果：

$$\min\{t_{i1}, t_{j2}\} \le \min\{t_{i2}, t_{j1}\}$$



- 最优顺序可以直接利用这个结果通过两个工序之间的检查来构造。
  - 令 ***I*** 表示工件序列，***S*** 表示顺序，**Johnson**算法可以描述为：

input: t_{i1}, t_{i2}, job list I,
output: best schedule S
step 1: Let $U = \{i | t_{i1} < t_{i2}\}$ and $V = \{i | t_{i1} \ge t_{i2}\}$
step 2: Sort jobs in U with non-decreasing order of t_{i1}
step 3: Sort jobs in V with non-increasing order of t_{i2}
step 4: An optimal sequence is the ordered set U followed by the ordered set V

8个工件问题的实例

Job i	1	2	3	4	5	6	7	8
t_{i1}	5	2	1	7	6	3	7	5
t_{i2}	2	6	2	5	6	7	2	1

- 最优解构造如下：

step 1: *Job sets U={2, 3, 6} and V={1, 4, 5, 7, 8}*

step 2: *Sort jobs in U as follows:*

Job i : 3 2 6

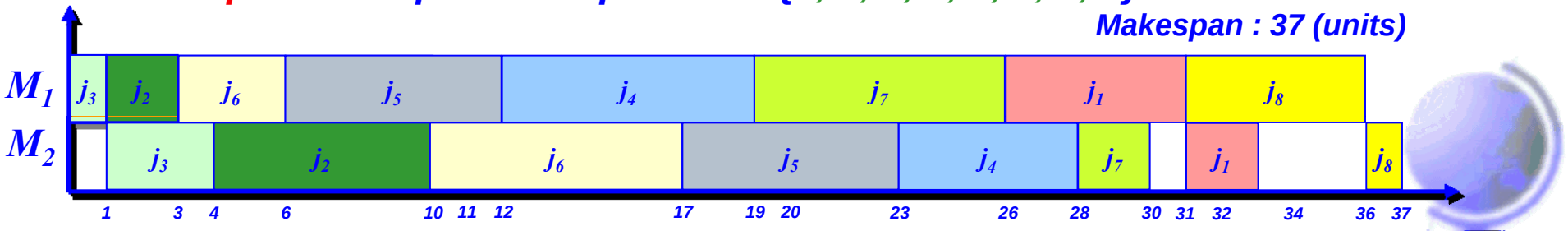
t_{i1} : 1 2 3

step 3: *Sort jobs in V as follows:*

Job i : 5 4 7 1 8

t_{i2} : 6 5 2 2 1

step 4: *An optimal sequence is {3, 2, 6, 5, 4, 7, 1, 8}*



m台机器的启发式算法

- Palmer启发式算法
- Gupta启发式算法
- CDS启发式算法
- RA启发式算法
- NEH启发式算法

GA求解--Gen-Tsujimura-Kubota方法

- 编码
  - 采用工件的换位表达，此类问题的自然表达方法。

	1:	2:	3:	4:
v_k =	3	2	4	1

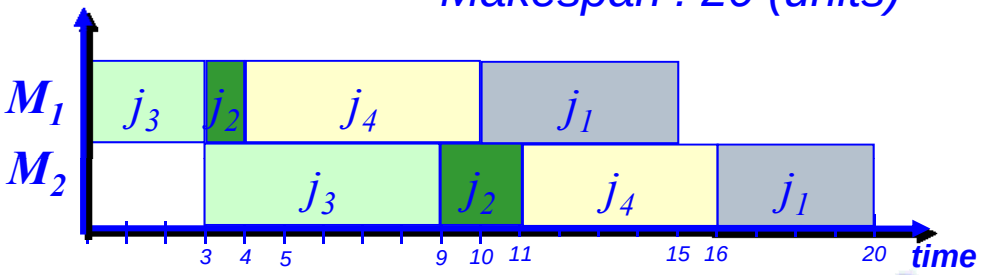
表示工件的加工顺序为：$j_3 \rightarrow j_2 \rightarrow j_4 \rightarrow j_1$

  - 调度S为：
    - $S = (t_{31}(0\text{-}3), t_{21}(3\text{-}4), t_{41}(4\text{-}10), t_{11}(10\text{-}15), t_{32}(3\text{-}9), t_{22}(9\text{-}11), t_{42}(11\text{-}16), t_{12}(16\text{-}20))$

Example of 4-Job and 2-machine Problem

Job i	1	2	3	4
t_{i1}	**5**	**1**	**3**	**6**
t_{i2}	**4**	**2**	**6**	**5**

Makespan : 20 (units)



Gantt chart of the optimal schedule

GA求解--Gen-Tsujimura-Kubota方法

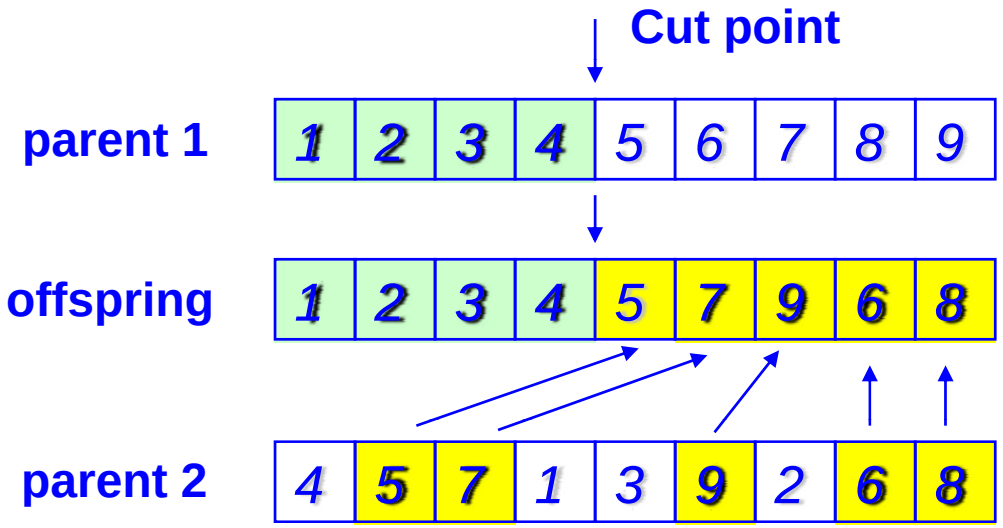
- 评估函数
  - 确定每个染色体适值的简单方法是用最大流程时间的倒数。

$$eval(v_k)=\frac{1}{C_{\max}^{k}}$$

- 交叉变异
  - **PMX、OX、CX、…;**
  - **Swap mutation、…**
  - 基于顺序表达的交叉、变异算子都适用

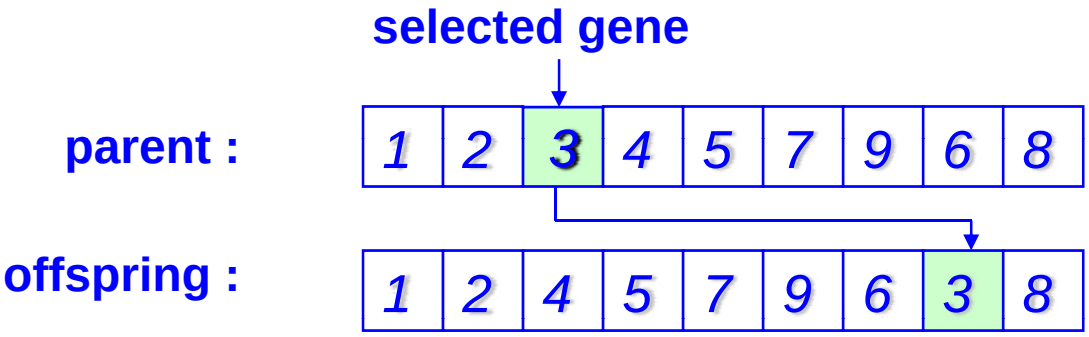
GA求解--Reeves方法

- 编码
  - 采用工件的换位表达，同前。
- 种群初始化
  - 采用NEH启发式算法产生一个染色体；
  - 其余的染色体随机产生
- 交叉
  - 采用单点交叉，
  - 首先随机选取断点，
  - 然后选取第一个双亲的断点前部分作为后代的一部分，
  - 再从第二个双亲中按顺序选取合法基因填充剩余部分。



GA求解--Reeves方法

- 变异
  - 互换变异
    - 随机选取染色体两个基因，进行简单互换
  - 移动变异
    - 随机选取一个基因，向左或右移动一个随机数位。



GA求解--Reeves方法

- 选择
  - 实质上，如果种群包含很多适值接近的染色体，好解与坏解的适值就没有明显的区别，
  - 在这种情况下相对适应性的度量便不能很好地区分染色体的好坏。
  - **Reeves**采用了一种简单的调度方法来作为选择的机理，即依据如下的概率分布选择双亲：

$$p_k = \frac{2k}{M(M+1)}$$

where

k：按适值递增顺序排列的 k^{th} 染色体

M：适应性最好的染色体

  - 这意味着中间值的染色体有 $1/M$ 的机会被选取，
  - 而最好的染色体(第M个染色体)有 $2/(M+1)$ 的机会，近似2倍中间值的机会。

作业车间调度问题

(Job-shop Scheduling Problems)

问题描述

- 古典作业车间问题：
  - 有 m 台机器和 n 个工件；
  - 每个工件包含一个由多道工序组成的工序集合；
  - 工件的工序顺序是预先给定的；
  - 每道工序要在它所要求的机器上加工某一固定时间；
- 问题的目标：
  - 确定机器上工序的加工顺序，使最大流程时间，即完成所有工件的时间最短.

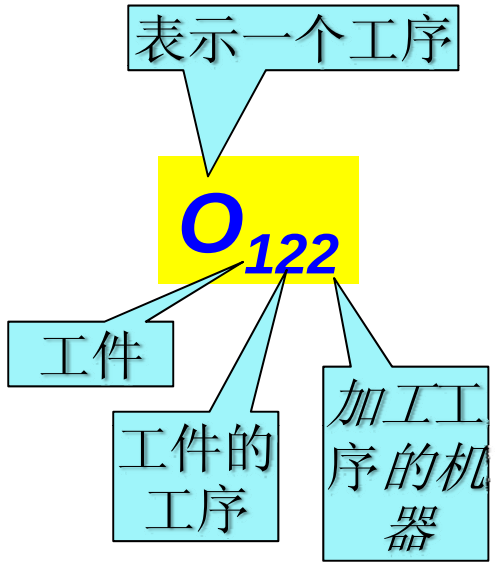
问题描述

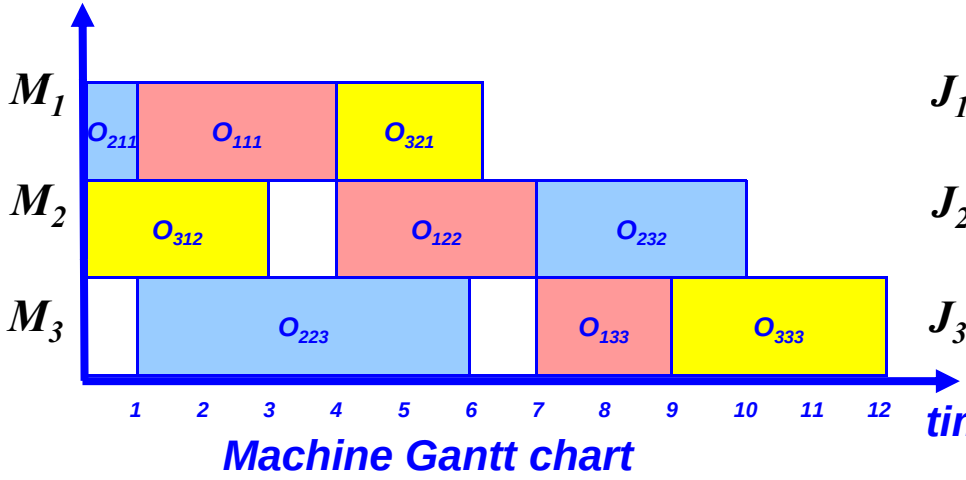
- 对工件和机器有如下约束：
  - 一个工件不能两次访问同一机器；
  - 不同工件的工序之间没有先后约束；
  - 工序一旦进行不能中断；
  - 每台机器一次只能加工一个工件；
  - 下达时间和交货期都不是给定的。

实例

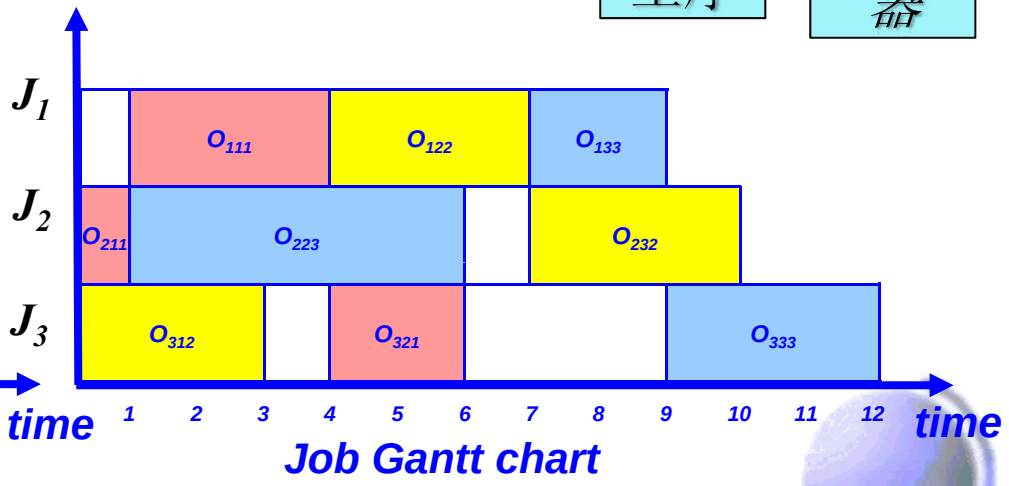
- 3个工件、3个机器的问题，可以用两类甘特图来表示

Processing Time				Machine Sequence			
Job	Operations			Job	Operations		
	1	2	3		1	2	3
J_1	3	3	2	J_1	M_1	M_2	M_3
J_2	1	5	3	J_2	M_1	M_3	M_2
J_3	3	2	3	J_3	M_2	M_1	M_3





Machine Gantt chart



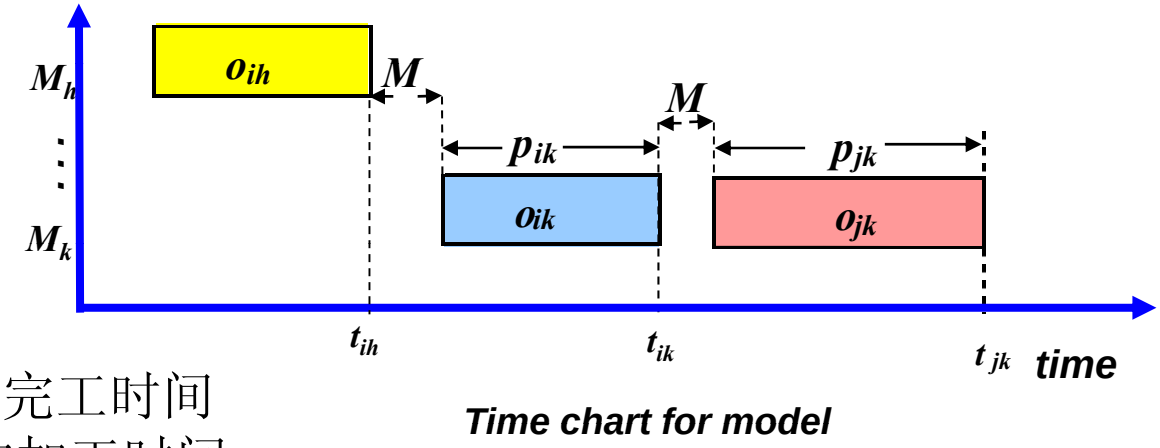
Job Gantt chart

应用

- 离散型企业的车间调度问题
  - 离散型制造业的产品往往由多个零件经过一系列并不连续的工序的加工最终装配而成。
  - 例如属于生产资料生产的机械、电子设备制造业，
  - 属于生活资料生产的机电整合消费产品制造业。
  - 作业车间调度问题是系统运行不可或缺的核心技术
- 模具企业生产调度问题
- 铁路列车运行调整问题
- 船用柴油机生产计划与调度
- 汽车制造业中，冲压车间生产计划和调度
- 数字媒体开发过程中的资源配置问题
- 家纺企业车间调度问题
- 钢铁企业中
  - 钢管生产调度问题
  - 冷轧生产线的机组作业调度问题

整数规划模型

- **Baker讨论了JSP问题的整数规划模型**
- 索引号
  - i, j : 工件的索引
  - h, k : 机器的索引
- 参数
  - n : 工件数
  - m : 机器数
  - t_{jk} : 工件j 在机器k上的完工时间
  - p_{jk}: 工件 j 在机器k上的加工时间
  - M : 很大的正数
- 决策变量
  - $a_{ihk} = \begin{cases} 1, & \text{工件}i\text{在机器}h\text{上的加工先于机器}k \\ 0, & \text{否则} \end{cases}$
  - $x_{ijk} = \begin{cases} 1, & \text{工件 } i \text{ 先于工件 } j \text{ 来到机器 } k \\ 0, & \text{否则} \end{cases}$



Time chart for model

整数规划模型

- 以最大流程时间最短为目标的JSP问题可描述如下：

$$\min t_M = \max_{1\le k\le m}\left\{\max_{1\le i\le n}\{t_{ik}\}\right\} \tag{1}$$

$$\text{s. t.} \quad t_{ik} - p_{ik} + M(1-a_{ihk}) \ge t_{ih}, \quad i=1,2,\ldots,n, \quad h,k=1,2,\ldots,m \tag{2}$$

$$t_{jk} - t_{ik} + M(1-x_{ijk}) \ge p_{jk}, \quad i=1,2,\ldots,n, \quad h,k=1,2,\ldots,m \tag{3}$$

$$t_{ik} \ge 0, \quad i=1,2,\ldots,n, \quad k=1,2,\ldots,m \tag{4}$$

$$x_{ijk} \ge 0 \text{ or } 1, \quad i,j=1,2,\ldots,n, \quad k=1,2,\ldots,m \tag{5}$$

考虑的是机器的工序非堵塞关系：多个工件不能在同一机器上同时加工。

考虑的是工序之间的前后关系：一个工件不能在不同的机器上同时加工。

线性规划模型

- **Adams等人讨论了作业车间调度问题的线性规划(LP)模型**
- **以最大流程时间最短为目标，模型描述如下：**

$$\min \quad t_n \qquad (1)$$

$$\text{s. t.} \quad t_j - t_i \geq d_i, \quad (i,j) \in A \qquad (2)$$

$$t_j - t_i \geq d_i \ \text{ or } \ t_i - t_j \geq d_j \quad (i,j) \in E_k, \quad k \in M \qquad (3)$$

$$t_i \geq 0, \quad i = 1,2,\ldots,n \qquad (4)$$

保证每个工件的工序顺序满足预先的要求

保证每台机器一次只能加工一个工件

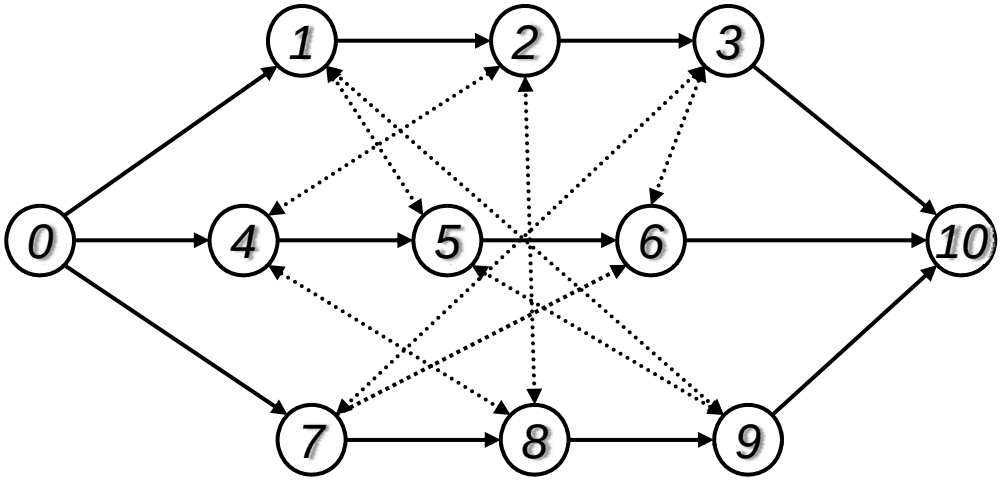
- $N=\{0,1,2,\ldots,n\}$ -- 表示工序的集合，0, n 表示虚设的起始和完成工序；
- $M=\{1,2,\ldots,m\}$ -- 是机器的集合；
- A -- 表示每个工件的工序前后关系约束的工序对集合；
- E_k --是机器 k 上加工的工序对集合(没有明确先后关系)；
- d_i -- 工序 i 的加工时间，是一定的；
- t_i -- 工序的起始时间，优化过程中有待确定的变量。

图模型

- 作业车间调度问题可以用非连接图来表示
- 非连接图： $G = (N, A, E)$ 定义为：
  - N：包含代表所有工序的节点；
  - A：包含连接同一工件的邻接工序的边
  - E：包含连接同一机器上加工工序的非连接边
    - 所谓非连接边是可以有两个可能方向的边。
- 一个可行的调度：
  - 将固定所有非连接边的方向，以确定同一机器上的工序顺序；
  - 一旦对于某台机器的顺序确定下来，连接工序的非连接边将被通常的带优先箭头的连接边取代
  - 非连接边集 E 分解成子集系E_k；
    - $E=E_1 \cup E_2 \cup E_3 ... \cup E_m$，每台机器一个系

图模型—实例

- 3个工件、3台机器的非连接图，每个工件包含3道工序



  - 节点N = {0,1.2,3,4,5,6,7,8,9,10}对应工序，其中0和10是虚设的起始工序和终止工序；
  - 连接边 A={(1,2), (2,3), (4,5), (5,6), (7,8), (8,9)} 对应同一工件的工序前后约束；
  - 非连接边(虚线表示)：
    - E_1={(1,5),(5,9),(9,1)}对应机器1上加工的工序；
    - E_2={(4,2),(2,8),(8,4)}对应机器2上加工的工序；
    - E_3={(7,3),(3,6),(6,7)}对应机器3上加工的工序。

传统的启发式算法

- 分类：
  - 一次通过启发式；
    - 一次通过启发式指通过基于优先分配规则一次固定一个工序，简单地构造一个完全解。有多种规则可用来从特定的子集中选择工序作为下一步的调度。
    - 这种启发式的特点是快，而且能找到不太坏的解。
  - 多次通过启发式；
    - 可以重复应用一次通过启发式来建立更为复杂的多次通过启发式，以花费额外的计算费用来获得更好的调度。
- 算法：
  - 优先分配启发式
  - 随机分配启发式
  - 瓶颈移动启发式
  - …

遗传算法求解

- 由于受工序的加工路线的约束，作业车间调度问题不像旅行商问题(TSP)那样容易确定一个自然表达。
- 到目前为止，还没有一个用系统不等式来表达先后约束的好方法。因此，不便于应用惩罚法来处理这些类型的约束。
- 大多数遗传算法和JSP问题的研究者喜欢使用修复策略来处理非可行性和非法性。
- 构造JSP问题遗传算法的重要主题：
  - 如何设计一个合适的表达解的方法和基于特定问题的遗传算子，使得不管在初期还是在进化过程中产生的所有染色体都将产生可行的调度，而这将是影响遗传算法的各个阶段的关键阶段。

作业车间的表达方法

- 直接方法
  - 一个调度被编码为一个染色体，应用遗传算法来进化染色体以确定一个好的调度。
    - 基于工序的表达法；
    - 基于工件的表达法；
    - 基于工件对关系的表达法；
    - 基于完成时间的表达法；
    - 随机键表达法都属于此类。
- 间接方法
  - 例如基于优先规则表达法，工件调度的分配规则序列被编码为一个染色体，应用遗传算法来进化染色体以确定一个好的分配规则序列。
    - 基于优先表的表达法；
    - 基于优先规则表达法；
    - 基于非连接图表达法；
    - 基于机器表达法都属于此类。

基于工序的表达

- 对于 ***n*** 工件 ***m*** 机器问题，一个染色体包括 ***n*m*** 个基因。
- 每个工件出现在染色体中 ***m*** 次，每个基因不表明一个工件的具体的工序，而是指有上下依赖关系的工序。
- 很容易看出染色体的任意排列总能产生可行调度。

- 例如：3个工件3台机器的调度问题
  - 这个表反映了，工件的各工序的加工时间、在哪台机器上加工、加工顺序是什么。

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

染色体编码的例子：v =

3	2	1	3	2	2	1	1	3

基于工序的表达

- 染色体解码说明：

(1 : job 1, 2 : job 2, 3 : job 3)

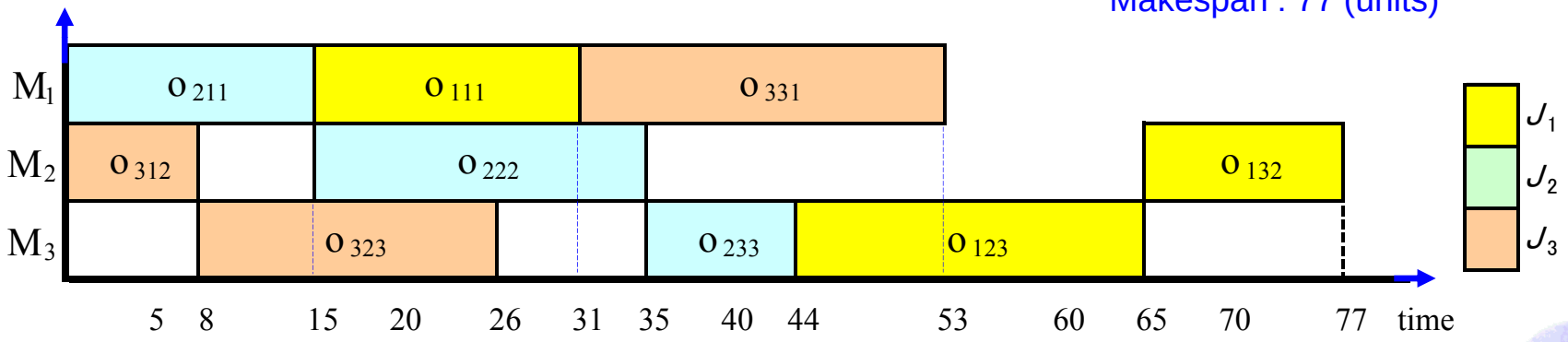
染色体编码: v =

3	2	1	3	2	2	1	1	3

对应的加工机器: 2 1 1 3 2 3 3 2 1

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

Makespan : 77 (units)



Gantt chart of one feasible schedule

基于工件的表达

- 对于 ***n*** 工件 ***m*** 机器问题，一个染色体由包含 ***n*** 个工件的列表组成。
- 每个调度根据工件的顺序来构造。
- 对于一个给定的工件顺序，列表中第一个工件的所有工序将被首先调度，然后考虑列表中第二个工件的工序。
- 加工中的工件第一道工序要求的加工时间应为相应机器最可能得到的加工时间，然后考虑第二道工序，如此进行，直到工件的所有工序排完。
- 这个过程以列表中每个工件的适当顺序重复进行。
- 工件的任意序列对应一个可行的调度。

- 例如：3个工件3台机器的调度问题

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

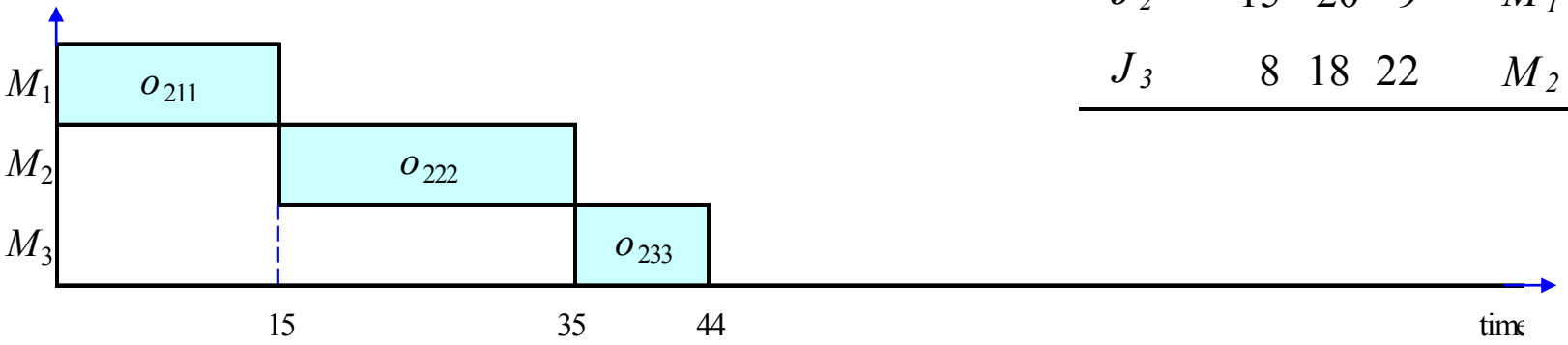
染色体编码的例子：v = | 2 | 3 | 1 |

基于工件的表达

- 染色体解码说明：
  - 假设给定染色体为： v=[2 3 1]
  - 第一个要加工的工件是：J_2,
  - 工件J_2的工序前后约束是：$[m_1, m_2, m_3]$
  - 每台机器的相应的加工时间为：[15, 20, 9]
  - 首先，J_2被调度如下。

染色体编码：v = | 2 | 3 | 1 |

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1



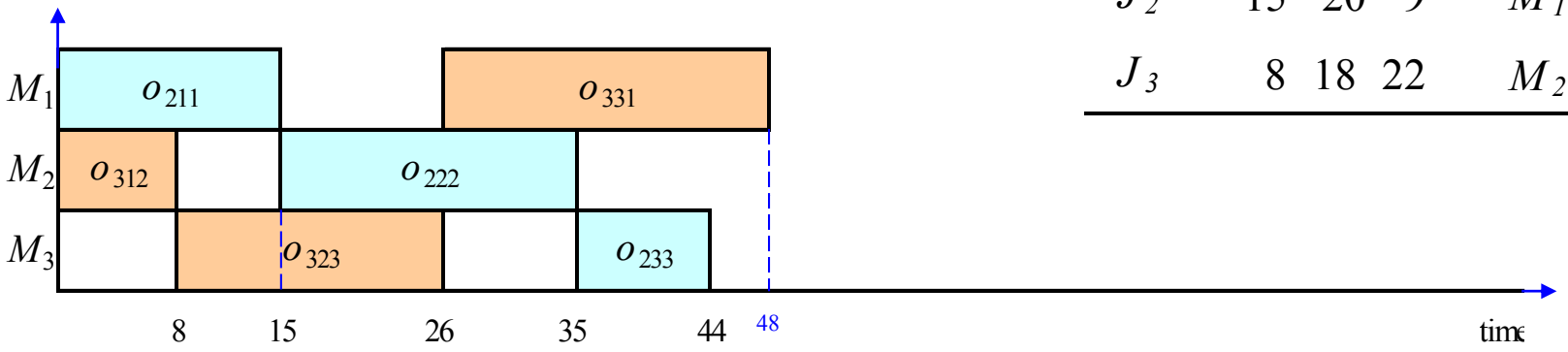
Gantt chart for the first job 2

基于工件的表达

- 染色体解码说明：
  - 第二个要加工的工件是：J_3，
  - 工件J_3的工序前后约束是： $[m_2, m_3, m_1]$
  - 每台机器的相应的加工时间为： [8, 18, 22]
  - 其次， J_3被调度如下。

染色体编码： $v =$ | 2 | 3 | 1 |

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1



Gantt chart for the second job 3

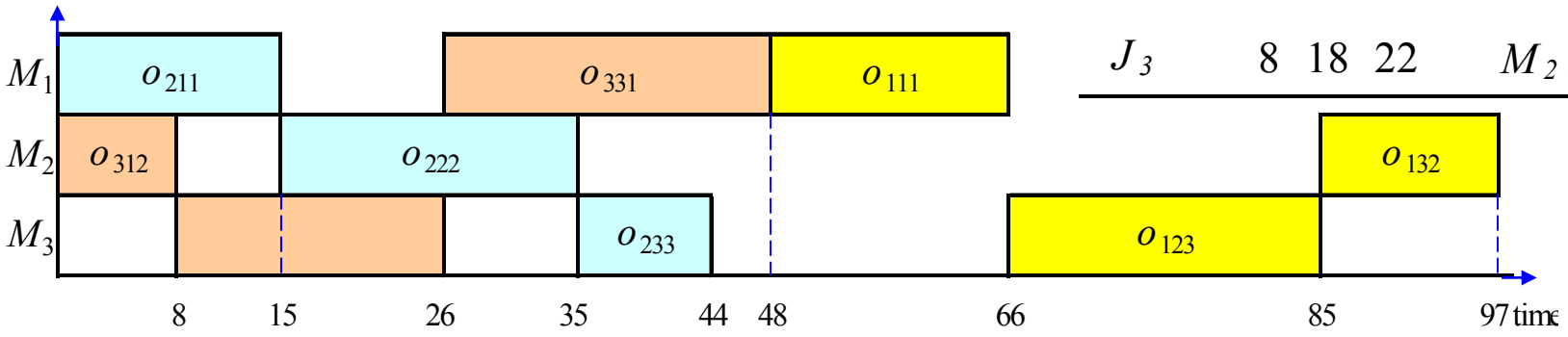
基于工件的表达

- 染色体解码说明：
  - 第三个要加工的工件是：J_1，
  - 工件J_1的工序前后约束是：$[m_1, m_3, m_2]$
  - 每台机器的相应的加工时间为：[16, 21, 12]
  - 最后，J_1被调度如下。

染色体编码：v = | 2 | 3 | 1 |

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

Makespan : 97 (units)



Gantt chart for the last job 1

基于优先表的表达

- 对于 ***n*** 工件 ***m*** 机器问题，一个染色体由 ***m*** 个子染色体组成，
- 每个子染色体对应一台机器，由一串长度为 ***n*** 的符号表示，每个符号代表一道在相关机器上处理的工序。
- 子染色体不能描述机器上工序的先后顺序，因为它们是优先列表，每台机器有其自身的优先列表。
- 实际的调度是由染色体通过模拟分析机器前等待排队的状态得到的，
- 如果有必要，用优先列表来确定调度，即选择首先出现在优先列表中的工序。
- 例如：3个工件3台机器的调度问题

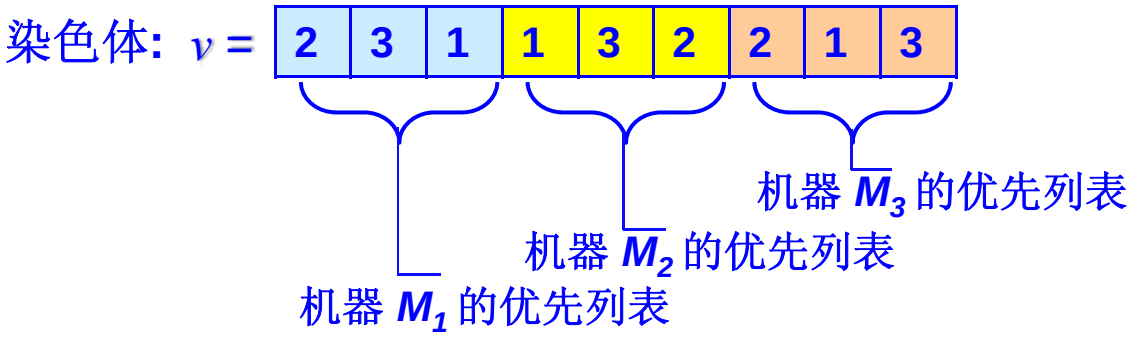
	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

染色体编码的例子：v =

2	3	1	1	3	2	2	1	3

基于优先表的表达

- 染色体解码说明：
  - 根据优先列表可知第一道优先的工序是：机器M_1上加工工件J_2，M_2上加工J_1，M_3上加工J_2。
  - 根据给定的工序先后约束，只有机器M_1上加工工件J_2是可行的调度。因此首先对机器M_1调度。

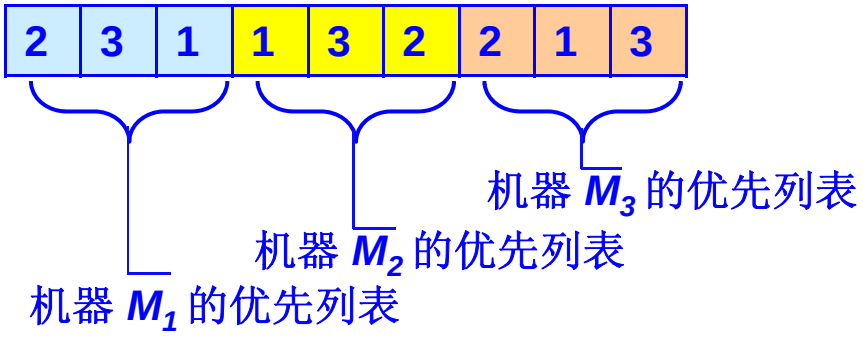


	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

基于优先表的表达

- 染色体解码说明：
  - 优先工序为：M_2上加工J_1，M_3上加工J_2。
  - 第二道优先工序为：机器M_1上加工工件J_3，M_2上加工J_3，M_3上加工J_1。
  - 只有机器M_2上加工工件J_3是可行的调度。因此对机器M_2调度。

染色体: v =

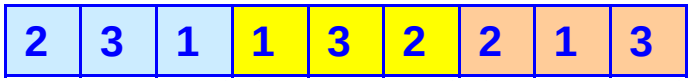


	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

基于优先表的表达

- 染色体解码说明：
  - 优先工序为：M_2上加工J_1，M_3上加工J_2，M_1上加工J_3，M_3上加工J_1。
  - 下一道优先工序为：机器M_1上加工工件J_1，M_2上加工J_2，M_3上加工J_3。
  - 可调度的工序是：M_1上加工J_1，M_2上加工J_2，M_3上加工J_3。

染色体：v =



机器 M_3 的优先列表

机器 M_2 的优先列表

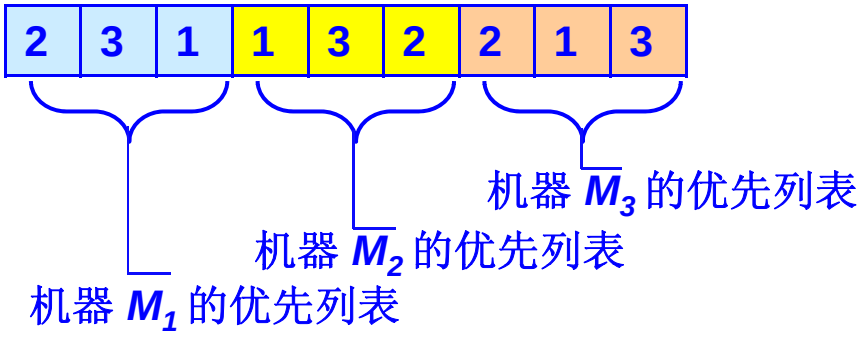
机器 M_1 的优先列表

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

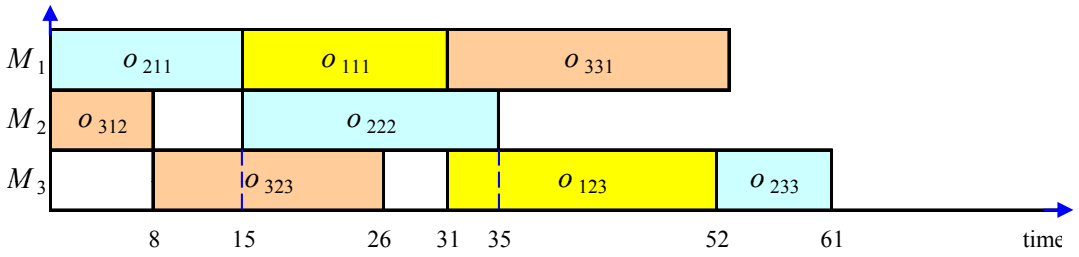
基于优先表的表达

- 染色体解码说明：
  - 优先工序为：M_2上加工J_1，M_3上加工J_2，M_1上加工J_3，M_3上加工J_1。
  - 可调度的工序是：M_3上加工J_1，M_3上加工J_2，M_1上加工J_3。

染色体：v =

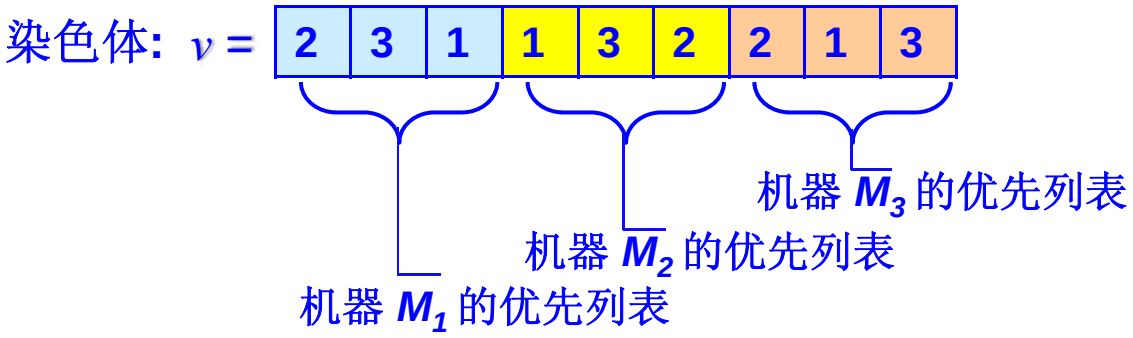


	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

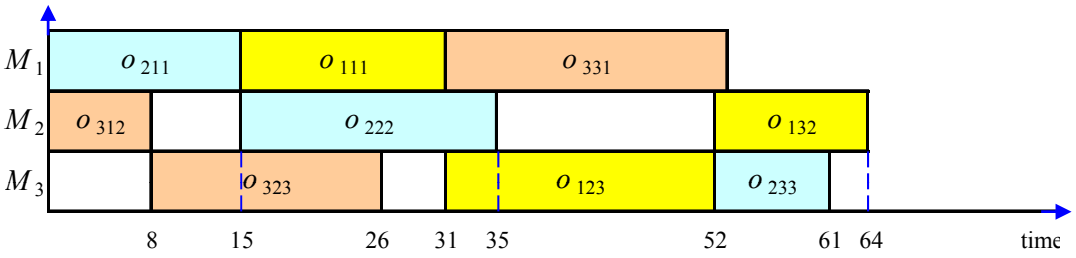


基于优先表的表达

- 染色体解码说明：
  - 优先工序为：M_2上加工J_1。
  - 可调度的工序是：M_2上加工J_1。



	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1



基于工件对关系的表达

- 采用二进制矩阵编码一个调度。矩阵根据相应机器上工件对的前后关系来确定。
- 例如：3个工件3台机器的调度问题。
  - 工件的工序前后约束和问题的一个可行调度如表所示。

工序先后约束				可行调度			
Job	机器顺序			Machine	工件调度		
J_1	M_1	M_2	M_3	M_1	J_2	J_1	J_3
J_2	M_1	M_3	M_2	M_2	J_3	J_1	J_2
J_3	M_2	M_1	M_3	M_3	J_2	J_1	J_3

  - 用一个二进制变量来表示工件对的先后关系，定义如下：

$$x_{ijm}=\begin{cases}1, & \text{如果工件 } i \text{ 优先于工件 } j \text{ 在机器 } m \text{ 上加工}\\ 0, & \text{其它}\end{cases}$$

  - 对于给定的可行调度，可以得到一个二进制矩阵表达法：

$$\begin{matrix}(J_1,J_2)\text{ on }(M_1,M_2,M_3):\\(J_1,J_3)\text{ on }(M_1,M_2,M_3):\\(J_2,J_3)\text{ on }(M_1,M_3,M_2):\end{matrix}\begin{pmatrix}x_{121} & x_{122} & x_{123}\\ x_{131} & x_{132} & x_{133}\\ x_{231} & x_{233} & x_{232}\end{pmatrix}=\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 1 & 1 & 0\end{pmatrix}$$

基于优先规则的表达

- **Dorndorf和Pesch**提出了一种基于优先规则的遗传算法，
- 其中染色体编码为一个工件分配规则的序列。
- 基于分配规则序列，用优先分配启发式构造调度。

- 对于 n 工件 m 机器问题，染色体是一个 $n*m$ 个项的串（$p_1, p_2 \ldots, p_{nm}$），p_i 代表预先设定的优先分配规则集合的一个规则。处于第i个位置的元素表达在算法的第i步迭代中的冲突应该由优先规则 p_i 来重新处理。更精确地说，用规则p_i 从冲突集合中选取工序，采用随机选择来断开链。

- 例如：选择的优先规则如下表

序号	规则	描述
1	SPT	选择具有最短加工时间的工序
2	LPT	选择具有最长加工时间的工序
3	MWR	选择剩余加工时间最长的工序
4	LWR	选择剩余加工时间最短的工序

染色体编码的例子：v =

1	2	2	1	4	4	2	1	3

基于优先规则的表达

- 算法参数说明：

$PS_t =$ 包含 t 步已调度工序的部分调度;

$S_t =$ 对应于 PS_t 在迭代 t 时可调度工序的集合;

$\sigma_i =$ 可调度工序集中,工序能够开始的最早时间 $i \in S_t$;

$\phi_i =$ 可调度工序集中,工序能够完成的最早时间 $i \in S_t$;

$C_t =$ 迭代 t 时冲突工序的集合.

- 基于优先规则的调度过程：

step 1. 让 t=1 ， PS_t 为空， S_t 包含所有无紧前工序的工序。

step 2. 确定 $\phi_t^*=\min_{i\in St}\{\phi_i\}$ 和能实现 ϕ_t^* 的机器 m^* ；
如果存在多个这样的机器，则通过随机选择来断开链。

step 3. 形成一个冲突集合C_t，它包含需要m^*进行加工的工序且满足 $i\in S_t$, $\sigma_i<\phi_t^*$。
从 C_t 中根据优先规则 p_t 选择一道工序尽可能早地添加到 PS_t 中，
产生一个新的部分调度 PS_{t+1}，如果存在多道工序可选，则随机选择一个。

step 4. 从 S_t 中移出被选择的工序，将直接后续工序添加到S_t，
更新 PS_{t+1}， t=t+1

step 5. 返回 step 2 直到产生一个完整调度

基于优先规则的表达

- 染色体解码说明：

$v =$

1	2	2	1	4	4	2	1	3

第1步，$t = 1$：

$PS_t = \{\}$;

$S_t = \{o_{111}, o_{211}, o_{312}\}$;

$\sigma_i = \{0,0,0\}$;

$\phi_i^* = \min\{16,15,\ 8\} = 8$;

$m^* = 2$

$C_t = \{o_{312}\}$

$PS_{t+1} = \{o_{312}\}$

$S_{t+1} = \{o_{111}, o_{211}, o_{323}\}$;

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

No	规则	描述
1	SPT	选择具有最短加工时间的工序
2	LPT	选择具有最长短加工时间的工序
3	MWR	选择剩余加工时间最长的工序
4	LWR	选择剩余加工时间最短的工序

基于优先规则的表达

- 染色体解码说明：

$v =$ | 1 | 2 | 2 | 1 | 4 | 4 | 2 | 1 | 3 |

第2步，$t = 2$：

$PS_t = \{o_{312}\};$

$S_t = \{o_{111}, o_{211}, o_{323}\};$

$\sigma_i = \{0,0,8\}$;

$\phi_i^* = \min\{16,15,\ 26\} = 15;$

$m^* = 1$

$C_t = \{o_{111}, o_{211}\}$

$PS_{t+1} = \{o_{312}, o_{111}\}$

$S_{t+1} = \{o_{123}, o_{211}, o_{323}\};$

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

No	规则	描述
1	SPT	选择具有最短加工时间的工序
2	LPT	选择具有最长加工时间的工序
3	MWR	选择剩余加工时间最长的工序
4	LWR	选择剩余加工时间最短的工序

基于优先规则的表达

- 染色体解码说明：

$v =$

1	2	2	1	4	4	2	1	3

第3步，$t = 3$：

$PS_t = \{o_{312}, o_{111}\};$

$S_t = \{o_{123}, o_{211}, o_{323}\};$

$\sigma_i = \{16,16,8\}$;

$\phi_i^* = \min\{37,31, 26\} = 26;$

$m^* = 3$

$C_t = \{o_{123}, o_{323}\}$

$PS_{t+1} = \{o_{312}, o_{111}, o_{123}\}$

$S_{t+1} = \{o_{132}, o_{211}, o_{323}\};$

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

No	规则	描述
1	SPT	选择具有最短加工时间的工序
2	LPT	选择具有最长加工时间的工序
3	MWR	选择剩余加工时间最长的工序
4	LWR	选择剩余加工时间最短的工序

基于优先规则的表达

- 染色体解码说明：

v =

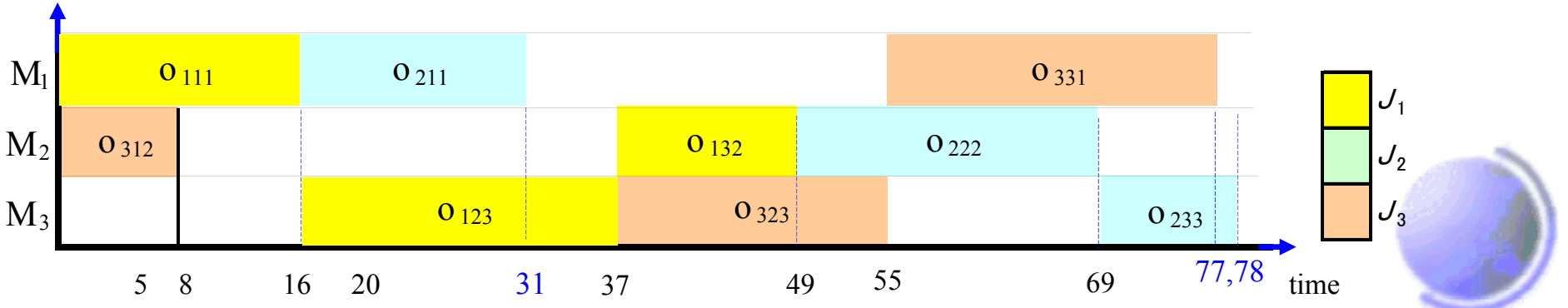
1	2	2	1	4	4	2	1	3

  - 以此类推，
  - 最后得到一个完全调度

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

No	规则	描述
1	SPT	选择具有最短加工时间的工序
2	LPT	选择具有最长加工时间的工序
3	MWR	选择剩余加工时间最长的工序
4	LWR	选择剩余加工时间最短的工序

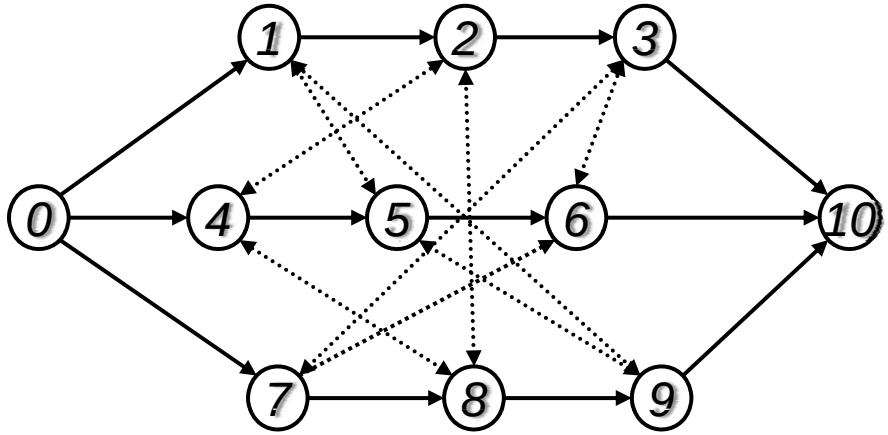
Makespan : 78 (units)



基于非连接图表达

- Tamaki和Nishikawa提出了一种基于非连接图的表达法，
- 也可视为一种基于工件对关系的表达法。
- 一个染色体包含了一个对应 E 中非连接边有序列表的二进制串，其中 e_{ij} 代表节点 i 与 j 之间的非连接边，其定义为:

$$e_{ij}=\begin{cases}1, & \text{标定非连接边的方向是从节点 } j \text{ 到节点 } i\\ 0, & \text{标定非连接边的方向是从节点 } i \text{ 到节点 } j\end{cases}$$



染色体编码的例子: v =

e_{15}	e_{19}	e_{59}	e_{24}	e_{28}	e_{48}	e_{36}	e_{37}	e_{67}
0	0	1	1	0	0	0	1	1

基于完成时间的表达

- Yamada和Nakano提出了一种基于完成时间的表达法，
- 一个染色体被编码为一个工序完成时间的有序列表。

染色体编码: v =

C_{111}	C_{123}	C_{132}	C_{211}	C_{222}	C_{233}	C_{312}	C_{323}	C_{331}

- 其中：C_{jir}表示工件 j 在机器 r上加工的第 i 道工序的完成时间。

➢ 这种表达法不实用于大多数的遗传算子，因为将产生非法调度。

➢ 他们为此设计了专用的遗传算子

	p_{ikj}			M_j		
Operations	1	2	3	1	2	3
J_1	16	21	12	M_1	M_3	M_2
J_2	15	20	9	M_1	M_2	M_3
J_3	8	18	22	M_2	M_3	M_1

基于机器的表达

- **Dorndorf和Pesch** 提出了一种基于机器表达的遗传算法，
- 染色体编码为机器的序列，
- 根据该序列用瓶颈移动启发式构造调度。

染色体编码: $v =$

m_1	m_2	m_3

- 这里的遗传算法被用于进化染色体，
- 并为瓶颈启发式算法找到一个好的机器顺序。
- 选择下一个机器的决策不再由瓶颈决定，
- 而是由染色体控制。

随机键表达

- 随机键表达法首先由Bean提出。
- 采用这种技术遗传算子可以产生可行的后代，而且不会因大量的调度和优化问题而增加总的计算开销。
- 随机键表达法用随机数编码成一个解，其值作为索引键以便对解进行解码。
- 对于*n*个工件、*m*台机器的调度问题，染色体由*m***n*个基因构成。
- 每个基因(一个随机键)包含两部分：
  - 整数部分{1,2,…,m}，用来分配给工件的机器；
  - 小数部分(0,1)，用来为每台机器上的工件排序。
- 例如：3工件3机器的JSP问题

染色体: *v* =

1.34	1.09	1.88	2.66	2.91	2.01	3.23	3.21	3.44

随机键表达

- 染色体解码说明：

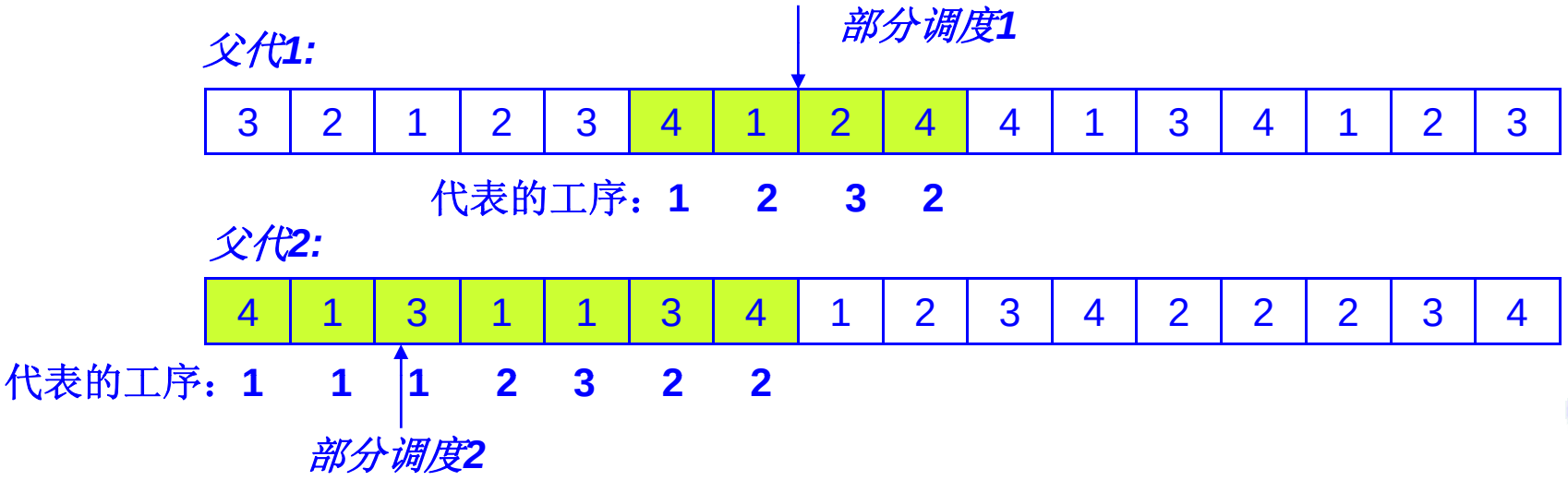
工件编号：	J1	J2	J3	J1	J2	J3	J1	J2	J3
染色体：v =	1.34	1.09	1.88	2.66	2.91	2.01	3.23	3.21	3.44
加工顺序：	2	1	3	2	3	1	2	1	3

  - 按索引键的增加顺序
    - 对于机器1，工件的顺序为：2→1→3
    - 对于机器2，工件的顺序为：3→1→2
    - 对于机器3，工件的顺序为：2→1→3
  - 令o_{jm}表达机器m上的工件j，染色体能转化为一个有序工序的唯一列表：

$$[o_{21}\ o_{11}\ o_{31}\ o_{32}\ o_{12}\ o_{22}\ o_{23}\ o_{13}\ o_{33}]$$

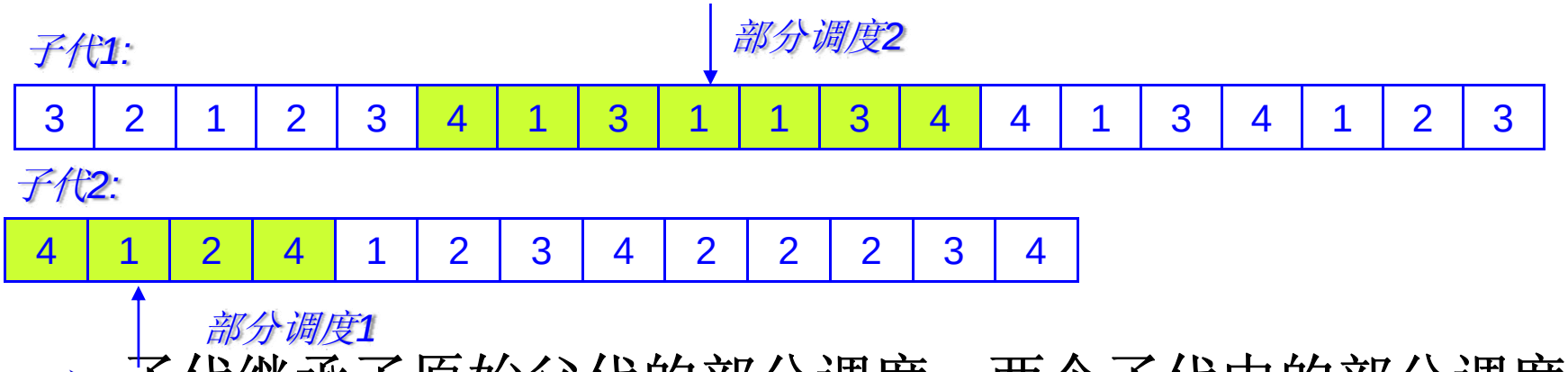
Gen-Tsujimura-Kubota方法

- 这种方法在“基于工序的表达方法”的基础上，设计了一个“部分调度交换交叉算子”，试图在后代中保留建筑块（building blocks）
- 部分调度用调度中起始和最终位置相同的工件来识别
- 例如：考虑两个双亲染色体如下：
  - 在父代1随机选择一个位置，假设是第6个位置，对应工件4
  - 在父代1中找到下一个工件4的位置9，这样得到一个部分调度【4 1 2 4】
  - 从父代2中，查找一个以工件4开始和结束的部分调度，其基因组成为【4 1 3 1 1 3 4】
  - 两个部分调度的起始工件4所代表的工序号要保持一致。
    - 即：父代1中的“部分调度”的起始工件4，在父代1中代表第一个工序，所以父代2中“部分调度”的起始工件4，在父代2中也应该代表第一个工序。

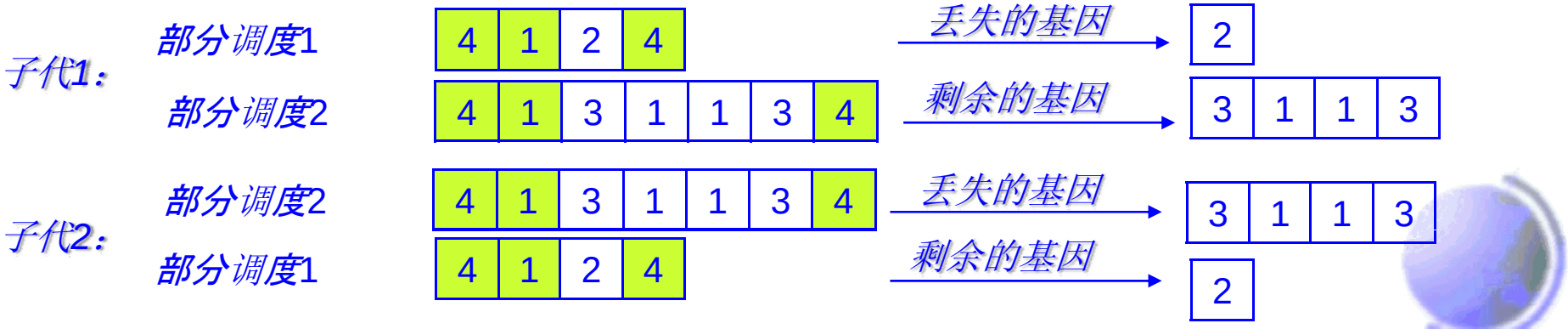


Gen-Tsujimura-Kubota方法

- 交换部分调度后，形成如下两个子代：

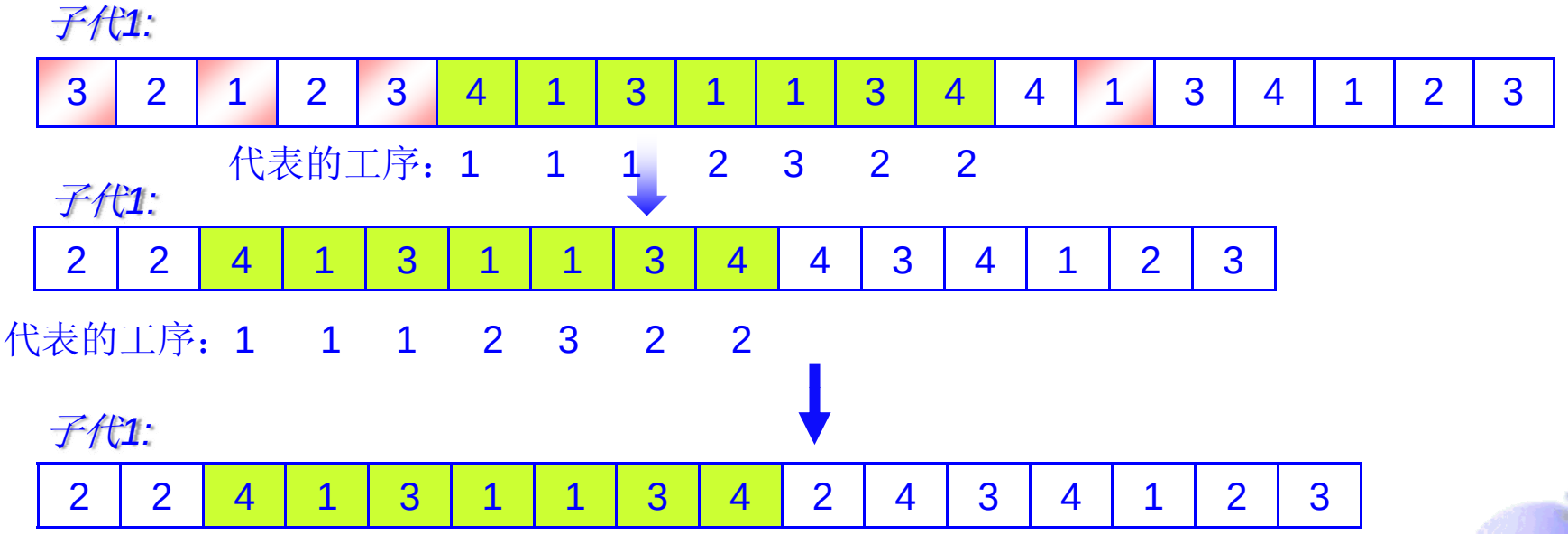


  - 子代继承了原始父代的部分调度，两个子代中的部分调度和父代中的基因具有同样的顺序。
- 通常部分调度包含不同的基因个数，因此通过交换得到的后代可能是非法的，过剩和丢失的基因如下确定：



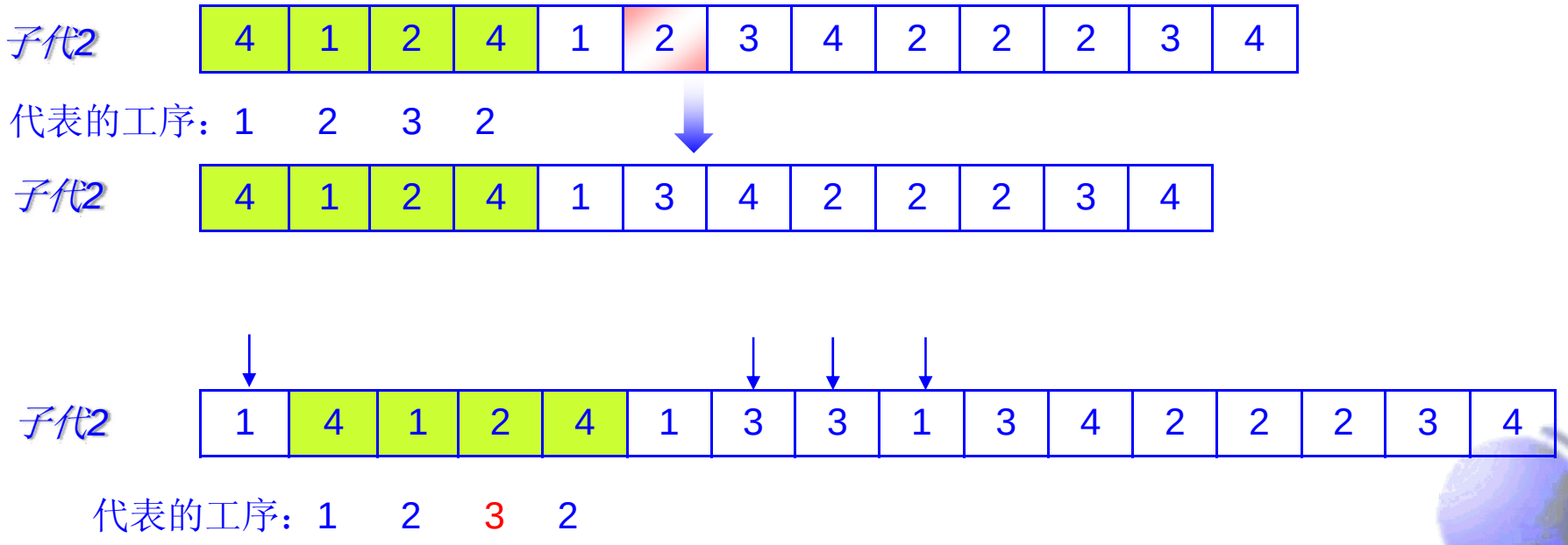
Gen-Tsujimura-Kubota方法

- 通过删除过剩基因和插入丢失基因使子代合法化，尽可能保持部分调度所代表的工序不变：
  - 对于子代1：
    - 过剩的基因是【3 1 1 3】，在部分调度之外，删除过剩的基因；
    - 丢失的基因是【2】，在部分调度之外，随机插入丢失的基因。



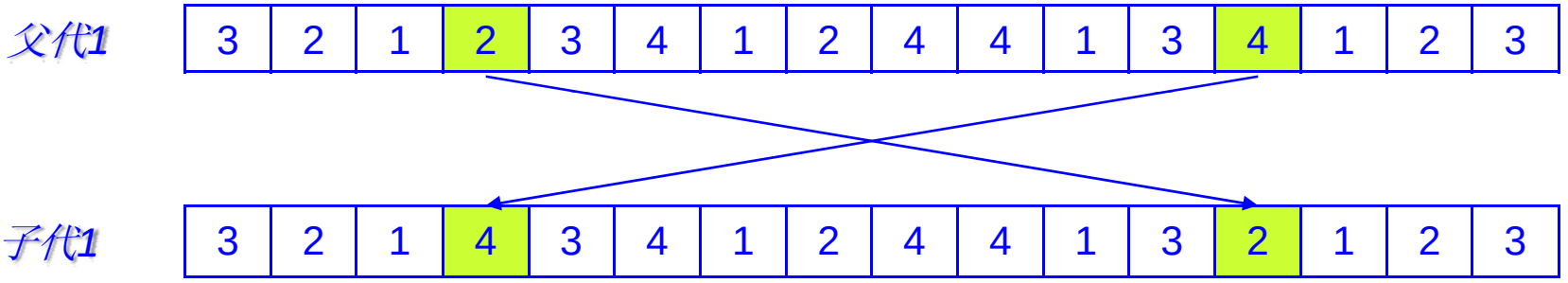
Gen-Tsujimura-Kubota方法

- 通过删除过剩基因和插入丢失基因使子代合法化，尽可能保持部分调度所代表的工序不变：
  - 对于子代2：
    - 过剩的基因是【2】，在部分调度之外，删除过剩的基因；
    - 丢失的基因是【3 1 1 3】，在部分调度前，插入丢失的基因1，部分调度之后，插入一个丢失的基因1，两个丢失的基因3插入到部分调度之外的任何位置。



Gen-Tsujimura-Kubota方法

- 工件对交换变异
  - 即随机选取两个不完全一致的工件交换位置。
  - 对于基于工序的表达法，染色体和调度之间的映像关系是多对一的，通过交换两个相邻的工件得到的后代可能产生与双亲相同的调度，
  - 因此两个不完全相同的工件之间的间距越大，效果越好。



Cheng-Gen-Tsujimura方法

- 为提高有效性，Cheng、Gen、Tsujimura进一步改进了GTK方法
- 主要改进包括三个方面：
  - 设计一个新的解码步骤确保产生活动调度；
  - 提出一种简化的交叉算子；
  - 用邻域搜索技术设计变异，以便进一步地搜索改进的后代。
- 在GTK方法中，染色体解码仅考虑2件事情
  - 给定染色体中工序的顺序
  - 每个工件的工序先后约束
- 解码只能确保产生半活动调度，而不能确保产生活动调度。就是说，在调度中可能存在允许左移（permissible left-shift）或全局左移（global left-shift)。
  - 因为最优调度是一个活动调度，解码步骤影响了本算法对大规模作业车间问题的有效性。

Cheng-Gen-Tsujimura方法

- 例如：3-Job、3-Machine的JSP问题

加工时间			
工件	工序		
	1	2	3
J_1	3	3	2
J_2	1	5	3
J_3	3	2	3

机器顺序			
工件	工序		
	1	2	3
J_1	M_1	M_2	M_3
J_2	M_1	M_3	M_2
J_3	M_2	M_1	M_3

机器	工件顺序		
M_1	J_2	J_1	J_3
M_2	J_1	J_2	J_3
M_3	J_1	J_2	J_3

染色体: v =

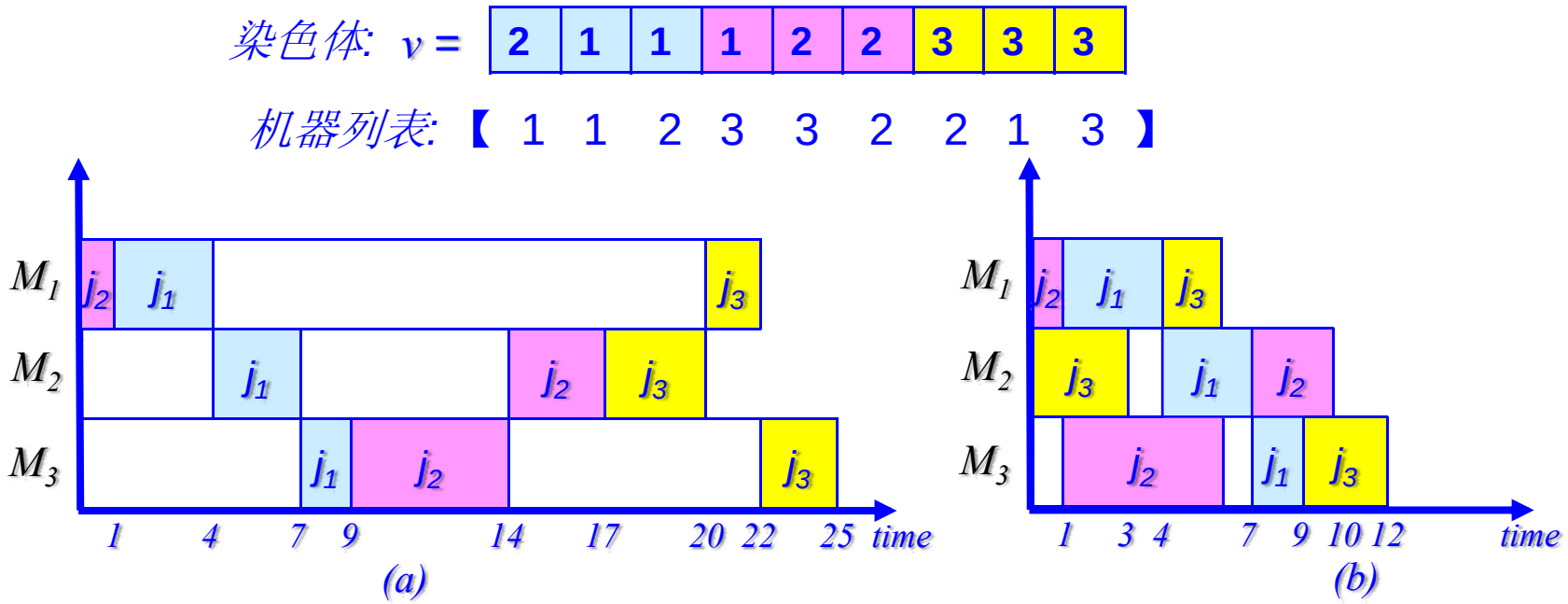
2	1	1	1	2	2	3	3	3

机器列表: 【 1 1 2 3 3 2 2 1 3 】

- 根据解码步骤得到相应的机器列表
- 根据机器列表可以构造一个调度，它确定了每台机器上工序的顺序。

Cheng-Gen-Tsujimura方法

- 对给定的染色体通过解码，得到半活动调度的甘特图，流程时间为：**makespan=25**



- 调度中有两个允许左移
  - 工件 j_3 在机器 M_2 上可以在时刻0开始加工
  - 工件 j_2 在机器 M_3 上可以再时刻1开始加工
- 通过实施这些左移，可以得到一个活动调度。其流程时间为：**makespan=12**

Cheng-Gen-Tsujimura方法

- 改进的解码步骤
  - CGT方法，改进了解码步骤，确保由一个给定的染色体产生一个活动调度。
- 步骤：
  - 首先把染色体解码为有序的工序列表；
  - 基于列表，采用一次通过启发式算法产生一个调度。首先调度列表中的第一道工序，然后考虑第二道工序，如此进行；
  - 每道处理中的工序的加工时间位于工序要求的机器的最可能的加工时间；
  - 重复这一过程，直到列表中的所有工序都被调度到适当的位置。

Cheng-Gen-Tsujimura方法

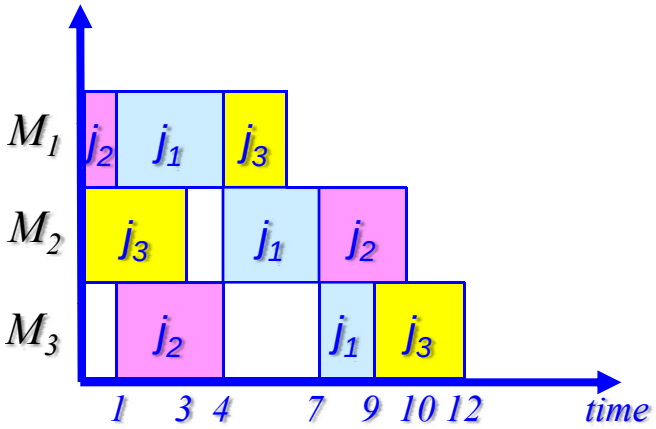
- 通过改进步骤得到的调度确保是一个活动调度，
- 染色体可以被视为分配了一个优先权给每道工序。
- 一个在列表中处于较低位置的工序有较高的优先权，可比其他有较高位置的工序优先调度。
- 染色体可被解码为有序工序的唯一列表：

染色体: $v =$

2	1	1	1	2	2	3	3	3

有序工序列表：【O_{211} O_{111} O_{122} O_{133} O_{223} O_{232} O_{312} O_{321} O_{333}】

- 工序 O_{211} 具有最高优先权，首先调度，然后调度O_{111}，依此类推，最后得到活动调度。



Cheng-Gen-Tsujimura方法

- 简化的部分调度交换交叉算子
  - GTK方法中的交叉算子相当复杂，因为它要求后代中部分调度的工序顺序与他们在双亲中的保持一致。
  - 改进的解码方法用染色体来确定每道工序的优先权，然后根据一次通过优先分配启发式产生调度，因此不要求双亲和后代中部分调度的工序顺序保持一致。
- 交叉过程：
  - 首先从双亲中分别选出包含相同个数基因的两个部分调度；
  - 交换部分调度产生后代；
  - 比较两个部分调度，找出丢失的基因和过程的基因；
  - 通过以随机的方式，删除过剩的基因和增加丢失的基因，使后代合法化。

Cheng-Gen-Tsujimura方法

- 说明：

机器调度问题

(Machine Scheduling Problems)

问题概述

- 机器调度问题既有丰富的研究内容，同时又是一个在机械制造、逻辑、计算机结构等方面有广泛应用前景的研究领域。
- 机器调度问题包括的主要方面有：
  - 机器配置
    - 单机问题
    - 多机问题
  - 工件特征（不限于）
    - 工件之间的先后关系
    - 工件下达时间
    - 工件交货期
    - 工件优先权
  - 目标函数
    - 单目标问题
    - 多目标问题
- 机器调度问题属于组合优化问题，其复杂性常常通过转换为已知复杂性的其他问题来确定

单机调度问题

- 基本单机调度问题的假设可描述为：
  - 一组由 n 个独立的单道工序工件在时间 0 等待加工；
  - 工件的装设时间与工件的顺序无关，可以包括在加工时间内；
  - 工件是可知的；
  - 机器连续可用，在工件等待加工时机器不能闲置；
  - 不允许工件预定机器；
- 问题的目标：
  - 确定工件的加工顺序，使得一些性能的度量最优。

基本定义

- $(j_1, j_1, \cdots j_n)$ 表示工件；
- p_i 表示工件 j_i 的加工时间；
- d_i 表示工件 j_i 的交货期；
- w_i 表示权重；
- c_i 表示完工时间；
- r_i 表示准备时间；
- 流程时间（**flowtime**） $F_i = c_i - r_i$，即工件在系统中的总逗留时间。用于度量系统对各个服务需求的反应，表示工件在到达和离开系统的时间长度。
- 延迟时间（**lateness**） $L_i = c_i - d_i$，即工件的完工时间超过其交货期的总时间。用于度量一个调度对于给定交货期的一致性。负值表示在交货期前完工；正值通常伴有惩罚和费用。
- 拖期时间（**tardiness**） $T_i = \max(0, L_i)$，表示工件j_i不满足交货期的延迟时间，如果满足交货期，其值为**0**。
- 令$\prod$表示工件之间无闲置时间的可行调度的集合。对于一个给定的调度$\sigma \in \prod$，令 $f(\sigma)$ 表示相应的目标函数值，且$r_i = 0$。

各种问题

① 平均流程时间问题（mean flowtime problem）

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \frac{1}{n}\sum_{i=1}^{n} F_i = \frac{1}{n}\sum_{i=1}^{n} c_i$$

- 可以用工件加工时间的非减顺序来调度工件，即：$p_{i1} \le p_{i2} \ldots \le p_{in}$。这就是著名的最短加工时间调度规则(SPT规则)。

② 加权流程时间问题（weighted flowtime problem）

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n} w_i F_i = \sum_{i=1}^{n} w_i c_i$$

- 可以用 p_i/w_i 的非减顺序来调度工件，即：$p_{i1}/w_{i1} \le p_{i2}/w_{i2} \ldots \le p_{in}/w_{in}$。这就是加权最短加工时间规则(WSPT)。

③ 平均拖期时间问题（mean tardiness problem）

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \frac{1}{n}\sum_{i=1}^{n} T_i$$

各种问题

④ 最大流程时间问题（maximum flowtime problem），也称为制造周期(makespan)问题

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \max_{1\le i\le n}\{F_i\} = \max_{1\le i\le n}\{c_i\}$$

⑤ 最大延迟时间问题（maximum lateness problem）

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \max_{1\le i\le n}\{L_i\}$$

- 可以用最早交货期规则(EDD规则)来调度工件，即: $d_{i1} \le d_{i2} \ldots \le d_{in}$ 。

⑥ 最大拖期时间问题（maximum tardiness problem）

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \max_{1\le i\le n}\{T_i\}$$

- 可以用EDD规则求解。

各种问题

⑦ 最少拖期工件数问题（minimum tardy job problem）

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n}\delta(T_i) \quad , \quad \delta(T_i)=\begin{cases} 1 & \text{if } T_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

- 以上讨论的所有指标都是工件完工时间的函数，所以有如下一般形式：

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \quad f(c_1, c_2, \ldots, c_n)$$

- 它们都属于一类重要的性能度量—规则度量。
  - 如果满足如下条件，则一个性能度量是规则的：
    - 调度的目标是极小化 $f(\sigma)$；
    - $f(\sigma)$ 增加，当且仅当至少有一个完工时间增加

各种问题

⑧ 完工时间方差问题（completion time variance problem - CTV）

$$\min_{\sigma\in\Pi} f(\sigma) = \frac{1}{n}\sum_{j=1}^{n}(c_j - \bar{c})^2, \quad \bar{c} = \frac{1}{n}\sum_{j=1}^{n} c_j$$

- 在最优的CTV调度中，最长时间的工件首先加工
- Cheng和Cai已经证明CTV问题是NP难题，不存在求解CTV问题最优解的多项式或伪多项式算法。

⑨ 等待时间方差问题（waiting time variance problem - WTV）

$$\min_{\sigma\in\Pi} f(\sigma) = \frac{1}{n}\sum_{j=1}^{n}(w_j - \bar{w})^2, \quad \bar{w} = \frac{1}{n}\sum_{j=1}^{n} w_j$$

- 其中w_i表示工件j_i的等待时间：$w_i = c_i - p_i$
- 最优WTV调度是V型调度，也就是说最短加工时间工件的紧前和后继工件分别按LPT和SPT顺序调度。

各种问题

⑩ 完工时间绝对偏差和问题（total absolute differences in completion time- TADC）

$$\min_{\sigma\in\Pi} f(\sigma) = \sum_{i=1}^{n}\sum_{j=i}^{n} | c_j - c_i | = \sum_{i=1}^{n}(i-1)(n-i+1)p_{[i]}$$

- 其中[*i*]表示第 *i* 个加工的工件。
- 因为CTV问题的复杂性．Kanet提出了TADC作为方差的度量，并且提出了一个求解该问题的*O*(n*Log*n)算法。
- TADC和CTV问题的关键区别是前者为绝对偏差和后者为方差和。
- 通过匹配位置权重(*i*-1)(*n*-*i*+1)的非增顺序与加工时间 $p_{[i]}$ 的非减顺序来求解。

⑪ 等待时间绝对偏差和问题（total absolute differences in waiting time- TADW）

$$\min_{\sigma\in\Pi} f(\sigma) = \sum_{i=1}^{n}\sum_{j=i}^{n} | w_j - w_i | = \sum_{i=1}^{n} i(n-i)p_{[i]}$$

- TADW和WTV之间的关系类似于TADC和CTV之间的关系。
- 可通过匹配位置权重 *i*(*n*-*i*) 的非增顺序与加工时间 $p_{[i]}$的非减顺序来求解。

- 以上给出的这些度量都是非规则度量。
  - 非规则性能度量可以随工件完成时间的减少而增加。

提前/拖期(E/T)调度问题

- 问题的提出
  - 调度模型中**Earliness and Tardiness**惩罚的研究是最近兴起的一个研究领域。
  - 多年来，调度研究主要侧重于对工件完工时间的非减规则度量
    - 如平均流程时间，
    - 平均延迟时间，
    - 工件拖期率，
    - 以及及平均拖期时间这样的度量。
  - 特别地，平均拖期时间准则尽管忽略了工件可能提前完工的结果，但它仍是度量交货期一致性的标准方式。
  - 然而，人们对准时生产**(JIT)**的兴趣改变了这种状态。
    - 准时生产（**Just-In-Time**）源于提前和拖期都不应该鼓励的思想；
    - 提前和拖期惩罚概念的引进，已经成为调度领域中一个新的迅速发展的研究分支。
  - 由于提前和拖期惩罚的使用，引出了一个非规则性能度量，导致了新的求解方法的研究。

E/T模型描述

- 令E_i和 T_i分别代表工件 j_i 的提前和拖期时间，并定义为：

$$E_i = \max\{0, d_i - c_i\} = (d_i - c_i)^+$$
$$T_i = \max\{0, c_i - d_i\} = (c_i - d_i)^+$$

- 令α_i>0, β_i>0 是与每个工件相对应的单位提前和单位拖期惩罚。
- 对于一个给定的调度$\sigma \in \prod$，基本E/T调度问题可以写成：

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n}[\alpha_i(d_i - c_i)^+ + \beta_i(c_i - d_i)^+]$$
$$= \sum_{i=1}^{n}(\alpha_i E_i + \beta_i T_i)$$

  - 惩罚可以采用不同的度量方法，可以对工件给出相同或不同的惩罚
  - 交货期可以是给定的，或者与工件顺序同时优化；
  - 最简单的模型是考虑所有工件有共同的交货期，更一般的模型是允许有不同的交货期。
  - 即使最简单模型的优化问题也是一个NP难题。

目标函数的几种形式

① 绝对偏差问题（absolute deviation problem）：

- E/T问题中的一个重要特例是工件完工时间与公共交货期的绝对偏差和的最小化问题，目标函数可以写成：

$$\min_{\sigma \in \Pi} \quad f(\sigma) = \sum_{i=1}^{n} | c_i - d |$$

- 其中$d_i=d$。模型中所有工件有相同的E/T惩罚系数
- 希望构造一个调度使得交货期 d 处于中间工件。
- 如果交货期太紧，则不可能在交货期前放置足够的工件，因此对于结定的问题，公共交货期太紧，就会成为限制问题，否则就是非限制问题。
- 非限制问题存在最优解，且最优解满足如下性质。
  - 调度中没有可插入的空闲时间；
  - 最优调度是V型调度；
  - 一个工件恰好在交货期完工；
  - 在最优调度中，第k个工件在时间d完工。其中k是大于或等于$n/2$的最小整数

目标函数的几种形式

② 加权绝对偏差问题（weighted absolute deviation problem）：

- Hall等研究的加权E/T问题定义为：

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n} w_i \left| c_i - d \right|$$

- 其中 w_i 是对应第 i 个工件的整数权重
- 模型的基本思想是不同的工件应该得到系数不同的惩罚。
- 从一般意义上来说，加权绝对偏差问题是NP完全问题。
- 令提前工件集 $E=\{j_i| c_i\le d\}$，拖期工件集 $T=\{j_i| c_i>d\}$，最优性条件如下
  - 存在一个最优解，其中某些工件恰好在交货期 d 完工；
  - 最优调度是V型调度，即：E中的工件按 w_i/p_i 非减排列，T中的工件按w_i/p_i非增排列；
  - 给定一个最优调度 σ^*，有：

$$\sum_{i\in E} w_i \ge \sum_{i\in T} w_i \quad , \quad \sum_{i\in E} p_i \ge \sum_{i\in T} p_i$$

目标函数的几种形式

③ 平方偏差问题（squared deviation problem）：

- 在某些情况下，不希望出现交货期的太大偏差，这时用与公共交货期的平方偏差作为性能度量可能更为合适：

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n}(c_i - d)^2 = \sum_{i=1}^{n}(E_i^2 + T_i^2)$$

- 这是总绝对偏差的二次形式。
- **Elion和Chowdhury曾经证明其最优解是V型调度**
- **Bagchi, Chang and Sullivan还研究了不同E/T惩罚的情况，即：**

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n}(\alpha E_i^2 + \beta T_i^2)$$

并行机调度问题

- 多机调度问题（Multiple machine scheduling）
  - 是研究机器对于一组工件的加工顺序，以确保所有工件在合理的总加工时间内加工完成。
  - 主要涉及以下三个问题：
    - 哪台机器分配给哪些工件?
    - 如何确定工件适当的加工顺序?
    - 如何评价调度的合理性?
  - 换句话说，多机调度理论主要关心的是如何提供一个机器与工件的完美匹配，或者是近似完美的匹配，然后确定每台机器上工件的加工顺序以达到预先规定的一些目标。
- 并行机器调度问题（Parallel machine scheduling）
  - 可以视为一类由多机调度问题松弛而得到的问题。
  - 在并行机器系统中，所有机器都相同并且每个工件可以在任何一个空闲机器上加工；
  - 每个加工完的工件释放一台机器并且离开系统。

工件描述及最优性准则

- 一个工件可从以下方面来描述：
  - 工件加工时间：单位加工时间或者各种加工时间；
  - 交货期要求：无交货期，公共交货期或者一般交货期；
  - 优先的工件：是否允许优先；
  - 先后约束：工件是独立的、非独立的或者有树形先后约束；
  - 工件准备时间：所有工件有相同准备时间或者任意准备时间。
- 最优性准则可以分为三个不同组：
  - 基于完成时间的度量；
  - 基于交货期的度量；
  - 基于闲置惩罚的度量。

假设条件

- 并行机器调度问题一般作如下假设：
  - 每个工件只有一道工序；
  - 一个工件同时只能在一台机器上加工；
  - 任何机器可以加工任意工件；
  - 一台机器同时只能加工一个工件；
  - 机器在调度期间内不会中断，且始终可用；
  - 机器加工时间与调度无关；
  - 机器装设时间可忽略；
  - 机器之间的运输时间可忽略；
  - 允许加工中的库存存在，其费用可以忽略；
  - 工件数可预先指定；
  - 机器数可预先指定；
  - 对于所有的 i 和 k，机器 m_k 上加工工件 j_i 的时间是预先给定的；
  - 对所有的 i，工件 j_i 的准备时间已知。

目标函数的几种形式

① 最小流程时间问题（minimum makespan problem）

- 考虑同时可用且相互独立的非优先工件的集合（*n*个工件）在完全一致、互不关联的并行机器集合(*m*台机器)(*m*<*n*)上加工的调度问题。
- 最小流程时间问题定义为:

$$\min_{\sigma \in \Pi} \quad f(\sigma) \;=\; \max\{c_j | j = 1, \ldots, n\}$$

- **Garey和Johnson**已经证明该问题在机器数不定的情况下从严格意义上来说是**NP**难题。
- 当机器数预先给出时可以在伪多项式时间内求解，因此只在一般的意义上是**NP**难题。

目标函数的几种形式

② 最小化加权流程时间问题（minimum weighted flowtime problem）

- 令w_i，r_i分别表示工件j_i的权重和准备时间。
- 问题是寻找一个最优的无预先指定的工件调度，使得加权流程时间最小：

$$\min_{\sigma \in \Pi} \quad f(\sigma) = \sum_{j=1}^{n} w_j (c_j - r_j)$$

- **Bruno**等已经证明，即使是双机系统，在不同工件具有不同权重的情况下，也是一个**NP**难题。

目标函数的几种形式

③ 最小化最大加权绝对延迟时间问题（minmax weighted absolute lateness problem）

- 给定一个非限制的公共交货期 d，

$$d \geq \sum_{j=1}^{n} p_j$$

- 问题是寻找一个最优的无优先工件的调度，使得最大加权绝对延迟时间最小，即：

$$\min_{\sigma \in \Pi} \quad f(\sigma) = \max\{w_j|c_j - d|; j = 1, \ldots, n\}$$

- Li和Cheng的研究表明，即使对于一个单机系统，该问题也是一个NP难题，并且提出了一个启发式方法来求解这一问题。

目标函数的几种形式

④ 最小化加权绝对延迟时间和问题

（ **minsum weighted absolute lateness problem** ）

- 该问题是寻找一个最优的工件调度，使得加权绝对延迟时间和最小，即：

$$\min_{\sigma \in \Pi} \quad f(\sigma) = \sum_{j=1}^{n} w_j |c_j - d|$$

- 最小和问题试图极小化工件完工时间关于交货期的加权绝对偏差和，
- 而最小最大问题是最小化工件完工时间关于交货期的最大加权绝对偏差。

机器调度问题的应用

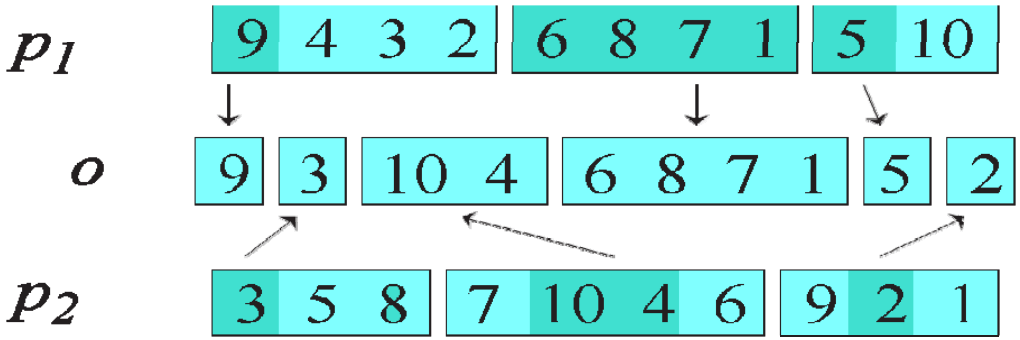
- 电网检修排序问题
- 分布式计算机系统中的任务调度
- 降落飞机的分组排序问题
- 制衣企业染整车间染缸优化调度问题
- ……

遗传算法求解

- **Cleveland-Smith方法**
  - 研究了用遗传算法求解单机调度的问题
- **Gupta-Gupta-Kumar方法**
  - 用遗传算法求解了一个以最小化流程时间方差为目标的 n 个工件单机调度问题
- **Lee-Kim方法**
  - 提出了一个并行遗传算法用于求解单机的工件调度问题
- **Cheng-Gen方法**
  - 应用遗传算法，求解了一个以最小化最大加权绝对延迟时间为目标的、并行机器系统的工件调度问题

Cleveland-Smith方法

- **Cleveland和Smith首先研究了用遗传算法求解单机调度的问题**
  - 他们考虑把遗传算法作为一种下达工件给制造资源(设备)的调度方法，使得加工这些工件的总体费用尽可能少。
  - 总体费用常常是库存、机器装设、提前拖期等费用的组合。
- 遗传算子
  - 这个问题与**TSP**有很多共同之处，可以借助很多求解**TSP**的**GA**方法来求解此问题。
  - 在他们的实验研究中，采用了一种“子巡回块算子”能够产生最好的结果。不再采用变异算子。
  - 子巡回块算子
    - 在两个双亲染色体中交互选择块，然后结合到后代中。后代中块的位置与其在双亲染色体中占据的位置几乎相同。一些冲突可以通过剪去块和向左向右滑动块来解决，以寻求到适当的位置



Gupta-Gupta-Kumar方法

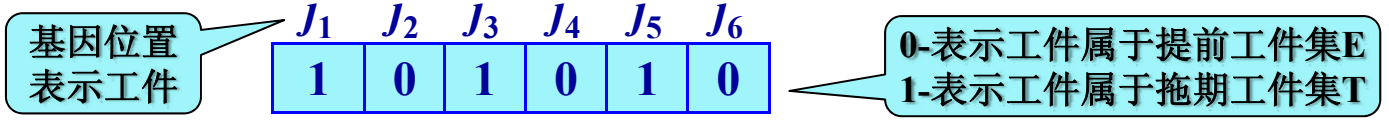
- **M.Gupta, Y.Gupta 和 A.Kumar曾用遗传算法求解了一个以最小化流程时间方差为目标的 *n* 个工件单机调度问题。**
  - 在他们的研究中，采用了顺序表达法，因为 *n* 个工件的顺序是单机调度问题的一种自然表达。
  - 遗传算法的实现中采用了**PMX**算子、交换变异、线性标定适值函数、以及聚集与精选的组合选择机制。
    - 聚集（**crowding**）与精选（**elitist**）的组合选择机制
      - 通过组合方式，产生一个后代池以构造新的种群。
      - 如果所有的后代都比前一代的每个染色体优越，那么在新的种群中，所有的后代代替原有的染色体。
      - 如果其中一部分相当好，那么它们替代前一代种群中相同数量的性能最差的染色体。
      - 对于剩余的后代，用预设置的概率值从前一代种群中选择后代。
      - 这种策略确保新种群继承前一代种群中最好的染色体。

Lee-Kim方法

- **Lee和Kim曾经提出了一个并行遗传算法，用于求解单机的工件调度问题**
  - 调度的目标是最小化公共交货期下的加权E/T惩罚和。
  - 一般的加权E/T问题（GWET）定义如下：

$$\min_{\sigma\in\Pi} \quad f(\sigma) = \sum_{i=1}^{n}\{\alpha_i(d-c_i)^+ + \beta_i(c_i-d)^+\}$$

    - 这是一个NP难题。
    - GWET问题的最优调度是以交货期为中心的V型调度，即：
    - 在交货期或交货期前完工的工件按照 p_i/α_i 非增的顺序加工；
    - 在交货期或交货期以后开始加工的工件按照 p_i/β_i 的非减顺序加工。
- 染色体编码
  - 采用二进制表达法把工件调度编码为染色体，



  - 给定一个染色体编码，就可以构造一个V型调度。

Lee-Kim方法

- 并行子种群
  - 并行遗传算法包含了一组子种群、每个子种群利用遗传算子独立地产生后代。
- 并行遗传算法的结构：

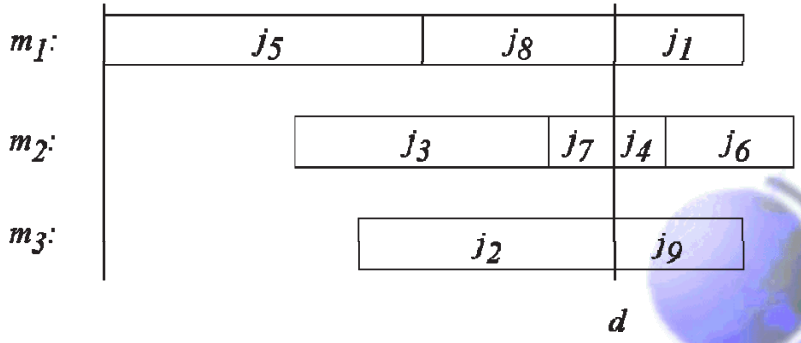
```
begin
   initialization
   evaluation
   while (not done of each subpop)
      begin
      reproduction
      crossover
      mutation
      evaluation
      communication
      deletion
   end
end
```

  - 交互是在并行遗传算法子种群之间交换信息的一种方法。每个子种群中最好的染色体被转换到其他的子种群，通过对特定组染色体的配对产生改进的调度。

Cheng-Gen方法

- **Cheng和Gen应用遗传算法，求解了一个以最小化最大加权绝对延迟时间为目标的、并行机器系统的工件调度问题**
  - 有一组已知加工时间和权重的工件集合，并有多台相同的并行机器和公共交货期。
  - 目标是寻找一个最优调度，使得最大加权绝对延迟时间最小，其目标函数是非规则性能度量。
- 染色体表达
  - 对于各种类型的多机调度问题主要是解决以下两个本质问题：
    - 分配工件结机器；
    - 对于每台机器调度工件。
  - **Cheng和Gen提出了一种扩展的顺序表达法，把这两个方面编码成染色体，其中整数代表所有可能的工件排列，*标志工件分配给机器。**
  - 例如，9个工件、3台机器的例子，假定有一个如图所示的调度，其染色体表示为：

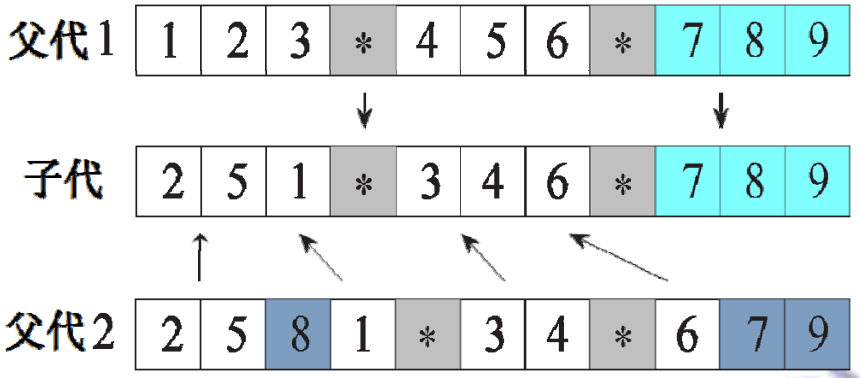
5	8	1	*	3	7	4	6	*	2	9



  - 对于*n*个工件、*m*台机器问题，一个
  - 合法的染色体应包括*n*个工件符号和
  - *m*-1个分配符号，总数为(*n*+*m*-1)。

Cheng-Gen方法

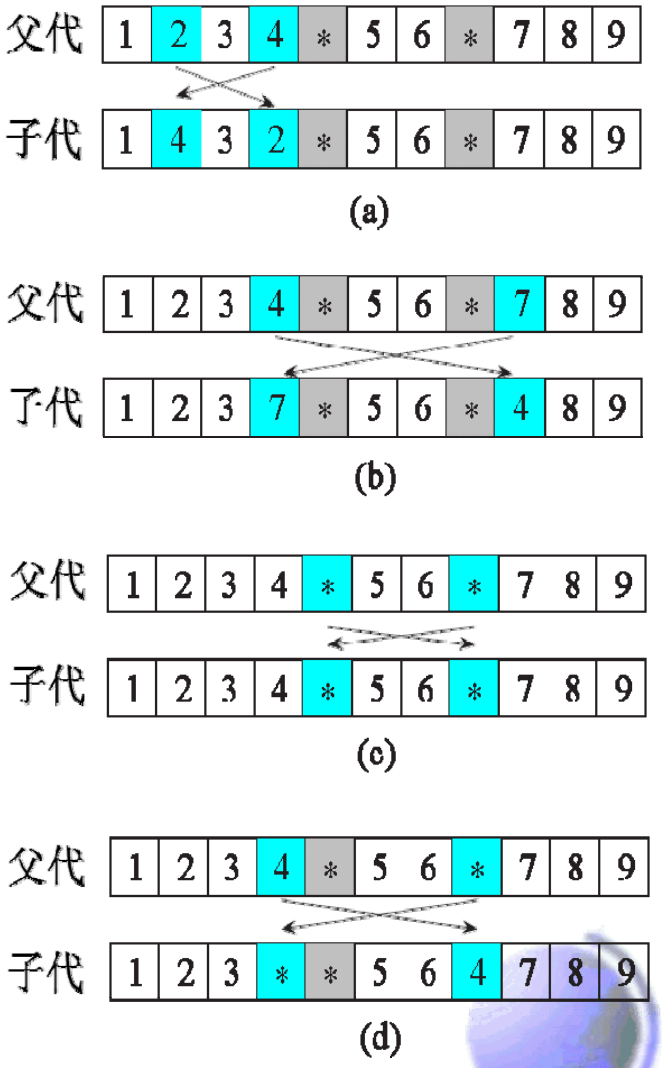
- 交叉
  - 多机调度问题的本质是工件和机器的组合和排列，可用交叉和变异算子来调整工件的分配和排列。
  - 多机系统下单台机器的一个调度称为子调度。子调度可以看作是遗传搜索的自然构成块，为了保持后代中这样的构成块，可以把一个双亲中的一个子调度扩展成一个后代，然后用另一个双亲中剩余的工件来完成该后代的构造。
  - 交叉过程如下：
    - 从一个双亲中获得*的位置，复制到子代；
    - 同时随机选择一个子调度，复制到子代；
    - 通过从左到右扫描，从另一双亲中得到剩余的工件



  - 这种交叉算子能够同时调整工件的分配和顺序。

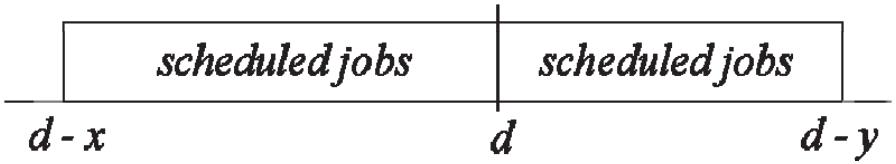
Cheng-Gen方法

- 交换变异
  - 随机交换的基因可以是工件，也可以是*号，工件和*号的不同组合导致了四种基本类型的变异：
    a) 如果两个基因都是工件，可能出现两种情况。一种情况是两个工件在同一台机器上加工，这时变异改变工件的顺序。
    b) 一种情况是两个工件在不同的机器上加工。这时变异改变染色体的工件顺序和工件对机器的分配
    c) 如果两个基因都是*号，变异执行无意义的操作，这种操作是禁止的。
    d) 如果一个基因是*，另一个基因是工件．变异同时改变染色体的工件顺序和工件对机器的分配。
  - 最后一种类型是改变*号位置的唯一遗传操作。没有这种操作，*的位置在整个进化过程中都不能改变，这一类的变异在遗传搜索中起到关键作用。



Cheng-Gen方法

- 启发式变异
  - 采用贪婪策略调整工件的顺序，从而形成对于每台机器的V型子调度
  - 也就是说，公共交货期前的所有工件按照 p_i/w_i 的非增顺序排列，而公共交货期后的工件按照 p_i/w_i 的非减顺序排列。
    - 令*d-x*表示最早工件的起始时间，*d+y*表示调度的最后一个工件的完工时间



    - 对于每台机器（由*界定），启发式变异的步骤如下：

step 1: Sort jobs in descending order of w_i/p_i. Let the sorted jobs be $(j_1, j_2, \ldots, j_n)$. Set $x \leftarrow 0$ and $y \leftarrow 0$.

step 2: Put job j_1 in the position before due date d and let $x \leftarrow x + p_1$.

step 3: for $(i = 2; i \leq n; ++i)$

```
    if (y + p_i < x) then
      add job j_i to the end of the scheduled job list;
      let y ← y + p_i;
    otherwise
      add job j_i to the beginning of the scheduled job list;
      let x ← x + p_i;
    endif
  endfor
```

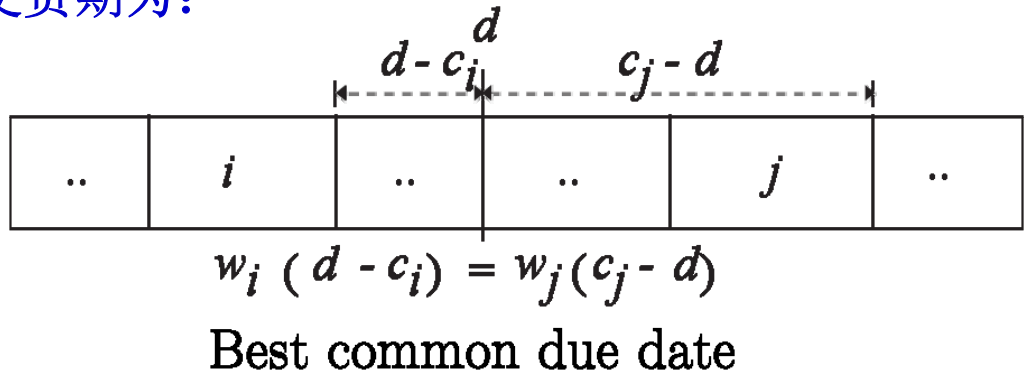
Cheng-Gen方法

- 确定最好交货期
  - 前面讨论的是如何用遗传算子处理工件分配和工件调度的问题，还没有讨论如何确定最好交货期问题。
  - 对于在一台机器上给定的一个工件顺序，能够最优地确定公共交货期。
    - 令交货期 *d* 是决策变量，对于任意两个工件 *i* 和 *j*, 最好交货期由如下公式确定：

$$w_i(d - c_i) = w_j(c_j - d)$$

    - 即，工件 *i* 和 *j* 的最好交货期为：

$$d = \frac{w_i c_i + w_j c_j}{c_i + c_j}$$



    - 依次可以计算所有可能工件对的交货期，给定工件调度的最好交货期可从方程(1)表示的具有最大绝对延迟的交货期中选取。

Cheng-Gen方法

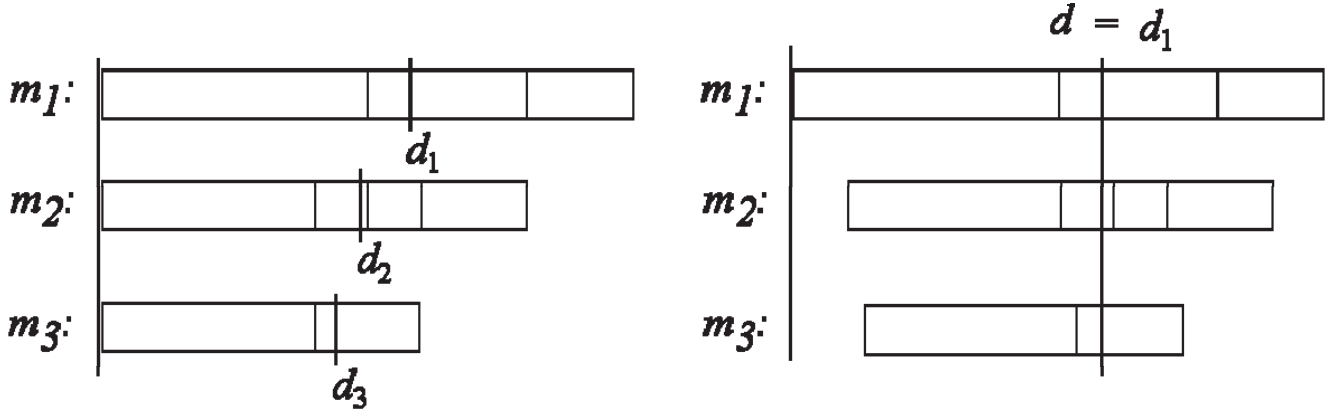
- 确定最好交货期
  - 对于多机情况，首先用以上方法计算机器 k 的最好交货期d_k，
  - m台机器的公共交货期由下式确定：

$$d = \max\{d_k \mid k = 1, 2, \ldots, m\}$$

  - 关于公共交货期 d，如果工件在机器 k上加工，需要推迟每个工件的起始时间：

$$\Delta_k = d - d_k$$

  - 起始时间的调整如图所示：



(a) before adjustment　　(b) after adjustment

End