EDA软件设计-调度算法

2024-2025年春季学期

主讲人：孙静翎

有约束的调度
Constrained Scheduling

- 约束调度
  - 一般情况下是NP完全问题
  - 在面积或资源的约束下最小化延迟（ML-RCS）
  - 使受到延迟约束的资源最小化（MR-LCS）
- 确切解决方法
  - ILP：整数线性规划（Integer linear program）
  - Hu算法：适用于只有一种资源类型的问题
- 启发式算法
  - 列表调度（List scheduling）
  - 力导向调度（Force-directed scheduling）

条件:

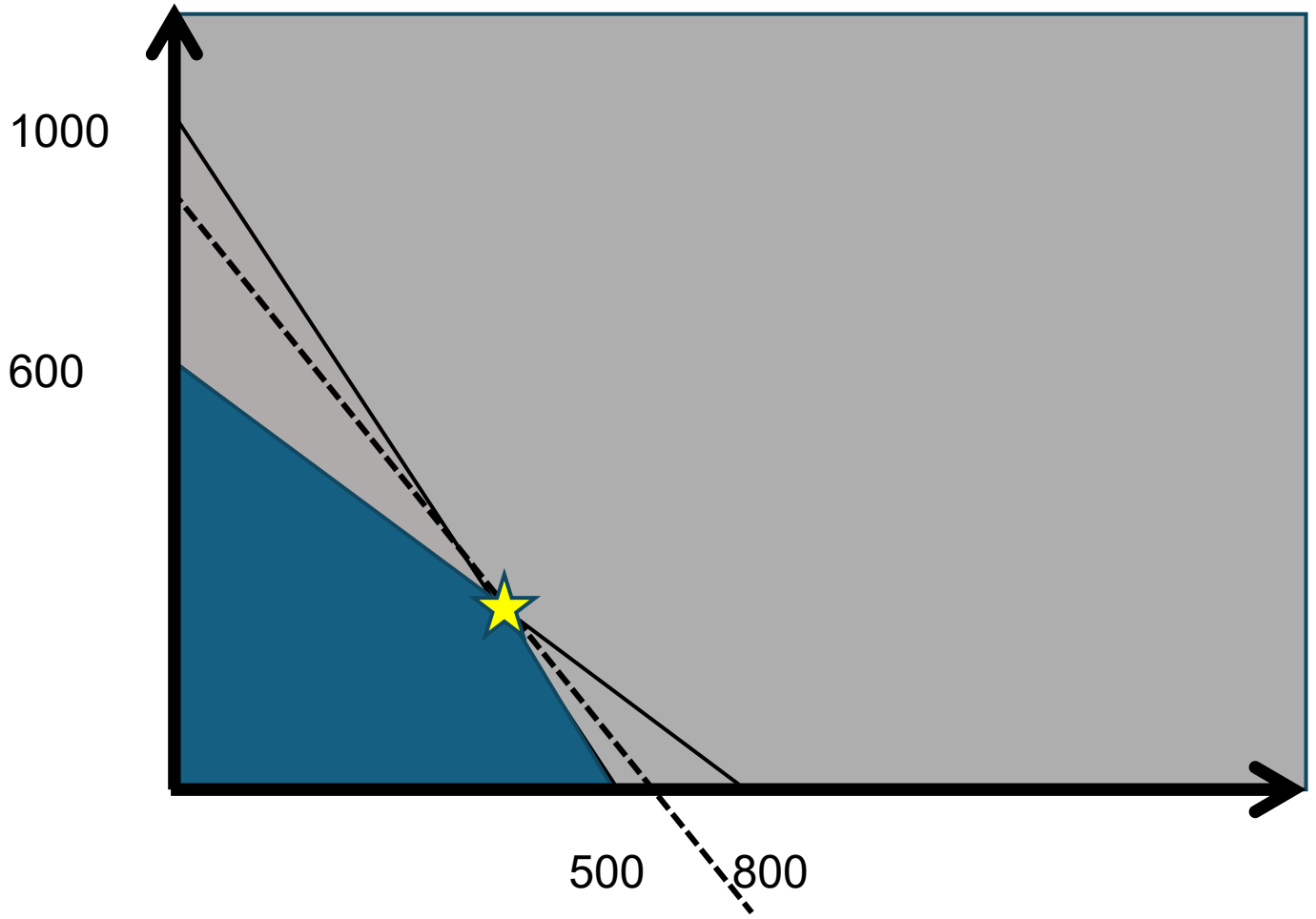
$X_1, X_2 >= 0$

$2X_1 + X_2 <= 1000$

$3X_1 + 4X_2 <= 2400$

最大化:

$8X_1 + 5X_2$



线性整数规划问题的求解器

商业求解器：*COPT, LEOPT, MindOpt, Gurobi, Matlab*

开源求解器：*HiGHS, SCIPC, SCIP, CBC, PULP*

The MIPLIB2017 Benchmark Instances - 8 threads (15 Sep 2024)

Choose base solver for comparison:

solver	score (as reported)	solved of 240
virtual best	0.73	95%
COPT	1.00 (1.00)	92%
LEOPT	2.90 (2.90)	75%
MindOpt	2.90 (2.90)	82%
XOPT	4.41 (4.41)	78%
HiGHS	6.94 (6.94)	66%
SCIPC	7.36 (7.36)	63%
SCIP	8.61 (8.61)	58%
CBC	12.80 (12.80)	45%

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

- 1. 确定待求解的变量
- 条件:
  - 2. 唯一约束（每个操作只会在一个周期开工）
  - 3. 顺序约束（当两个结点间有单向边时，代表要完成前一个操作后一个操作才能开工）
  - 4. 资源约束（资源是有限的）
- 目标:
  - 5. 最小化周期时间

用ILP来形式化ML-RCS问题

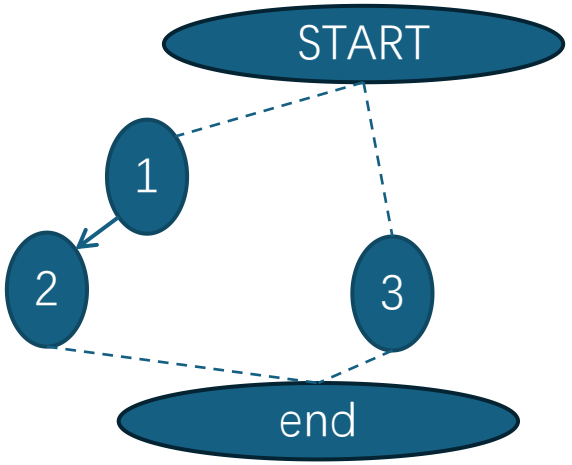
ILP Formulation of ML-RCS

已知：有n=4个待调度操作，需要考虑的周期上限是λ + 1=4

λ=3

n=4

- 当1，3是乘法操作，延迟为1。
- 2是加法操作，延迟为1。
- 且有两个乘法器和一个加法器时

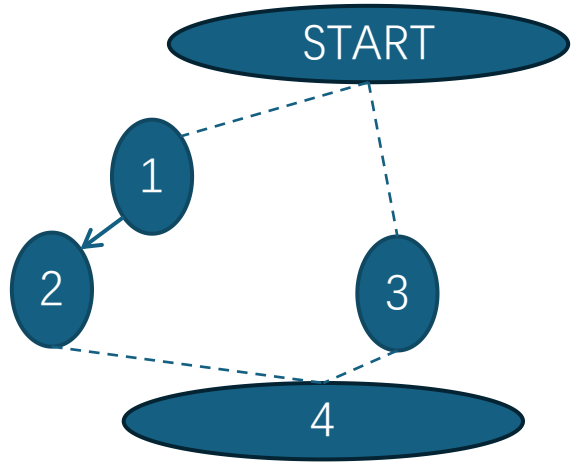


用ILP来形式化ML-RCS问题 - 1. 待求解的变量

ILP Formulation of ML-RCS - 1. Variables to be Solved

分别用一个变量x表示每个操作在每个周期开工的可能性

即：$X = \{ x_{il},\ i = 1,2,\cdots, n;\ l = 1,2,\cdots, \lambda + 1\}$

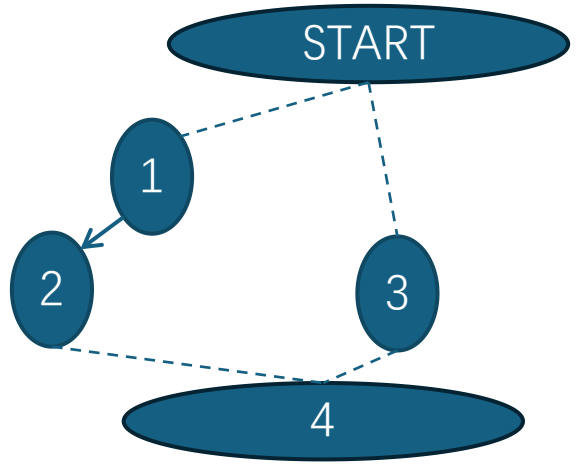


用ILP来形式化ML-RCS问题 - 1. 待求解的变量

ILP Formulation of ML-RCS - 1. Variables to be Solved

分别用一个变量x表示每个操作在每个周期开工的可能性

即：$X = \{ x_{il},\ i = 1,2,\cdots, n;\ l = 1,2,\cdots, \lambda + 1\}$



用ILP来形式化ML-RCS问题 - 1. 待求解的变量

ILP Formulation of ML-RCS - 1. Variables to be Solved

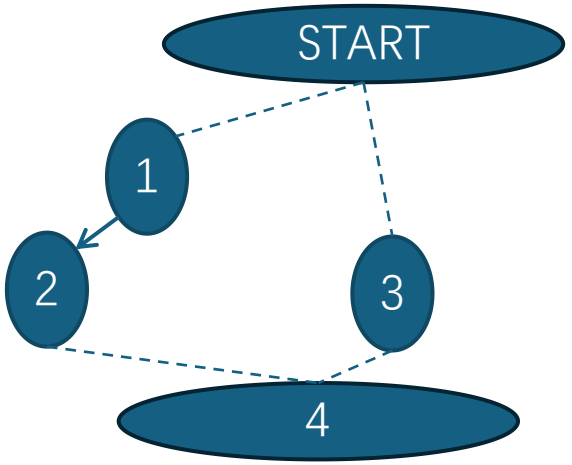
已知：有n个操作，最差情况是λ + 1个周期

分别用一个变量x表示每个操作在每个周期开工的可能性

即：$X = \{ x_{il},\ i = 1,2,\cdots, n;\ l = 1,2,\cdots, \lambda + 1\}$

x11, x12, x13, x14
x21, x22, x23, x24
x31, x32, x33, x34
x41, x42, x43, x44

λ =3



用ILP来形式化ML-RCS问题 - 1. 待求解的变量

ILP Formulation of ML-RCS - 1. Variables to be Solved

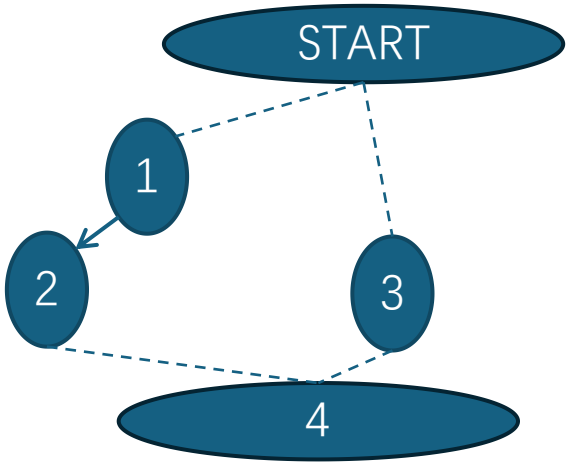
已知：有n个操作，最差情况是λ + 1个周期

分别用一个变量x表示每个操作在每个周期开工的可能性

即：$X = \{ x_{il},\ i = 1,2,\cdots n;\ l = 1,2,\cdots, \lambda + 1\}$

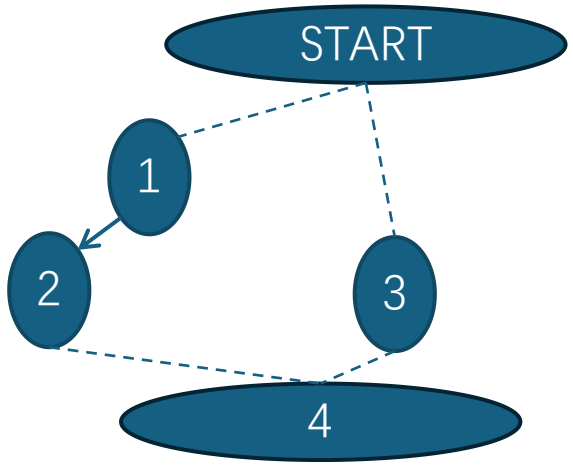
x11, x12, x13, x14, x15, x16
x21, x22, x23, x24, x25, x26
x31, x32, x33, x34, x35, x36
x41, x42, x43, x44, x45, x46

λ =5



用ILP来形式化ML-RCS问题 - 2. 唯一约束

ILP Formulation of ML-RCS - 2. Unique constraints



用ILP来形式化ML-RCS问题 - 2. 唯一约束

ILP Formulation of ML-RCS - 2. Unique constraints

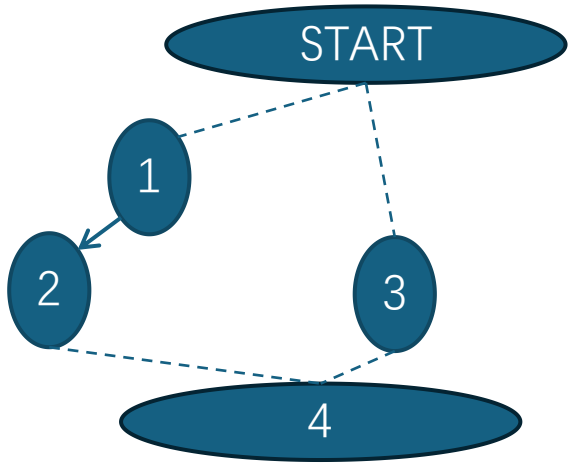
对每个操作i，都有：$\sum_{l=1}^{\lambda+1} x_{il} = 1$

$$x_{1,1} + x_{1,2} + \cdots + x_{1,\lambda+1} = 1$$

$$x_{2,1} + x_{2,2} + \cdots + x_{2,\lambda+1} = 1$$

$$x_{3,1} + x_{3,2} + \cdots + x_{3,\lambda+1} = 1$$

$$x_{4,1} + x_{4,2} + \cdots + x_{4,\lambda+1} = 1$$



用ILP来形式化ML-RCS问题 - 2. 唯一约束

ILP Formulation of ML-RCS - 2. Unique constraints

对每个操作i，都有：$\sum_{l=1}^{\lambda+1} x_{il} = 1$

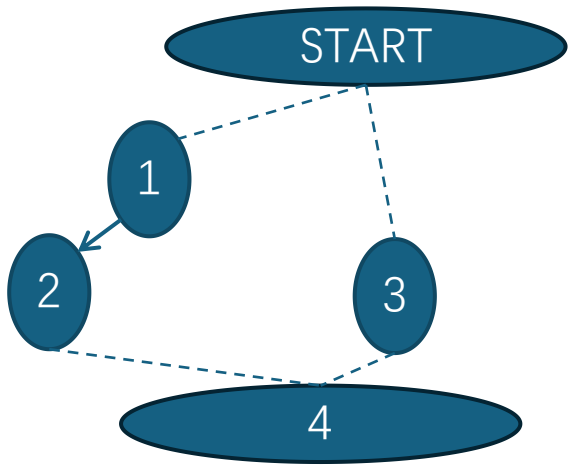
$x_{11}+x_{12}+x_{13}+x_{14}=1$

$x_{21}+x_{22}+x_{23}+x_{24}=1$

$x_{31}+x_{32}+x_{33}+x_{34}=1$

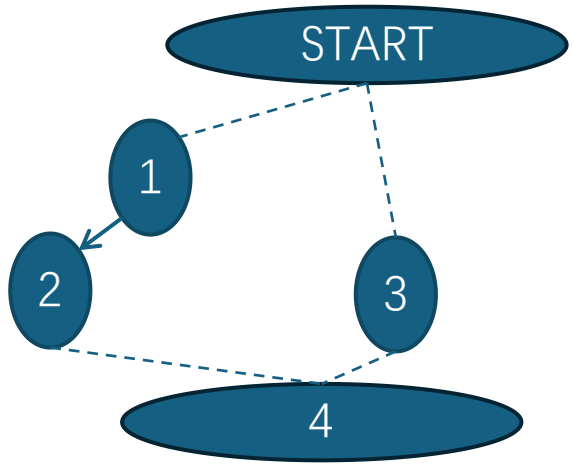
$x_{41}+x_{42}+x_{43}+x_{44}=1$

$\lambda = 3$



用ILP来形式化ML-RCS问题 – 3. 顺序约束

ILP Formulation of ML-RCS - 3. Sequential constraints



用ILP来形式化ML-RCS问题 – 3. 顺序约束

ILP Formulation of ML-RCS - 3. Sequential constraints

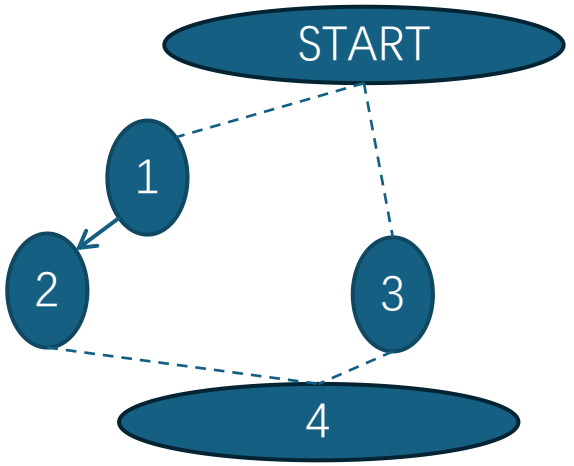
每个操作i的开工周期ti的计算公式为：$\sum_{l=1}^{\lambda+1} l * x_{il}$

$t_1=(1*x_{1,1}+\cdots+(\lambda+1)*x_{1,\lambda+1})$

$t_2=(1*x_{2,1}+\cdots+(\lambda+1)*x_{2,\lambda+1})$

$t_3=(1*x_{3,1}+\cdots+(\lambda+1)*x_{3,\lambda+1})$

$t_4=(1*x_{4,1}+\cdots+(\lambda+1)*x_{4,\lambda+1})$



用ILP来形式化ML-RCS问题 – 3. 顺序约束

ILP Formulation of ML-RCS - 3. Sequential constraints

每个操作i的开工周期ti的计算公式为：$\sum_{l=1}^{\lambda+1} l * x_{il}$

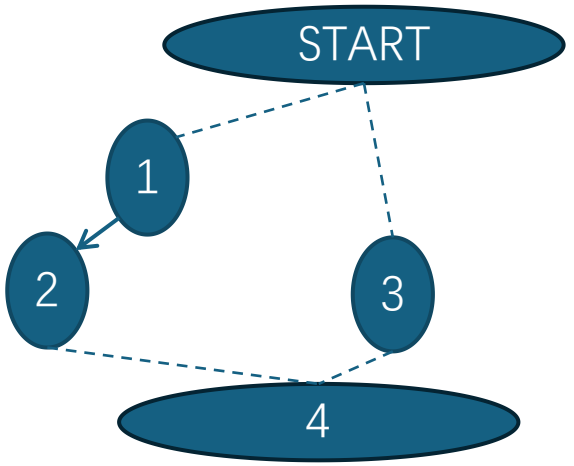
$t_1=1*x11+2*x12+3*x13+4*x14$

$t_2=1*x21+2*x22+3*x23+4*x24$

$t_3=1*x31+2*x32+3*x33+4*x34$

$t_4=1*x41+2*x42+3*x43+4*x44$

$\lambda = 3$



用ILP来形式化ML-RCS问题 – 3. 顺序约束

ILP Formulation of ML-RCS - 3. Sequential constraints

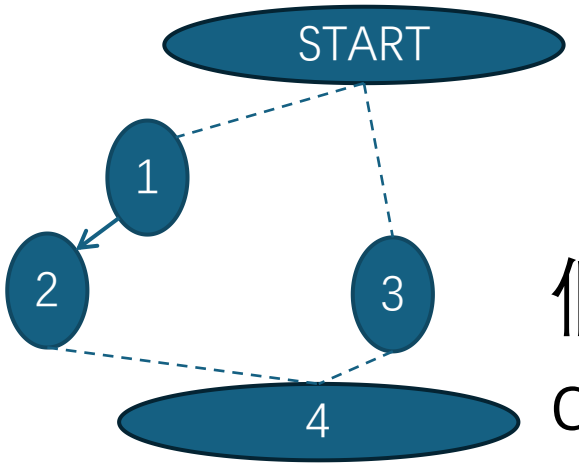
对每条边(vj, vi) ϵ E，都有ti ⩾ tj + dj，这里的t表示每个操作的开工周期

注意，不用考虑所有和虚拟起始结点start相关的边，dj是给定值不是变量

对于从1指向2的边有： t2 >= t1+1

对于从2指向4的边有： t4 >= t2+1

对于从3指向4的边有： t4 >= t3+1



假设
d1=1
d2=1
d3=1

用ILP来形式化ML-RCS问题 – 4. 资源约束

ILP Formulation of ML-RCS - 4. Resource constraints

对每种资源k，都在每个周期l列出约束：

$$\sum_{i:T(v_i)=k} \left(\sum_{m=l-(d_i-1)}^{l} x_{i,m} \right) \leq a_k$$

即对于每一个使用资源k的操作

用ILP来形式化ML-RCS问题 – 4. 资源约束

ILP Formulation of ML-RCS - 4. Resource constraints

对每种资源k，都在每个周期l列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \le a_k$$

即对于每一个使用资源k的操作都有这样一个累加式：

$$\sum_{m=l-(d_i-1)}^{l} x_{i,m}$$

m从$l-(d_i-1)$一直取值到l

把这些累加式加起来的结果小于等于资源k的数量a_k

用ILP来形式化ML-RCS问题 – 4. 资源约束

ILP Formulation of ML-RCS - 4. Resource constraints

如果l=6，di=3

则m从6-(3-1)=4开始一直到6

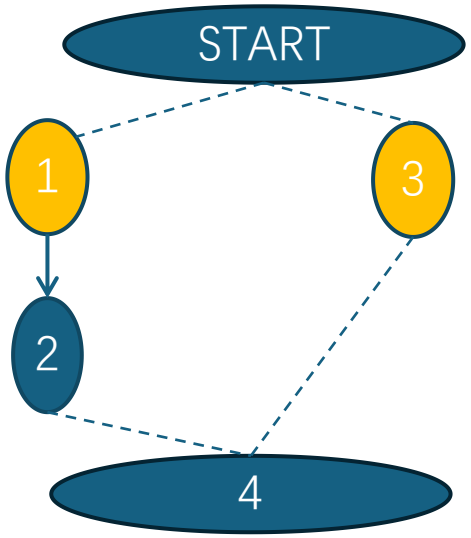
l-(di-1)　　l

$$\sum_{m=l-(d_i-1)}^{l} x_{i,m}$$

展开为：xi4+xi5+xi6

用ILP来形式化ML-RCS问题 – 4. 资源约束

ILP Formulation of ML-RCS - 4. Resource constraints



$\lambda = 3$

对每种资源k，都在每个周期l列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \le a_k$$

- 当1，3是乘法操作，延迟为2。
- 2和4是加法操作，延迟为1。
- 且有两个乘法器和一个加法器时

对周期1：$(x_{1,0}+x_{1,1})+(x_{3,0}+x_{3,1})<=2$ 且 $x_{2,1}+x_{4,1}<=1$

对周期2：$(x_{1,1}+x_{1,2})+(x_{3,1}+x_{3,2})<=2$ 且 $x_{2,2}+x_{4,2}<=1$

对周期3：$(x_{1,2}+x_{1,3})+(x_{3,2}+x_{3,3})<=2$ 且 $x_{2,3}+x_{4,3}<=1$

对周期4：$(x_{1,3}+x_{1,4})+(x_{3,3}+x_{3,4})<=2$ 且 $x_{2,3}+x_{4,3}<=1$

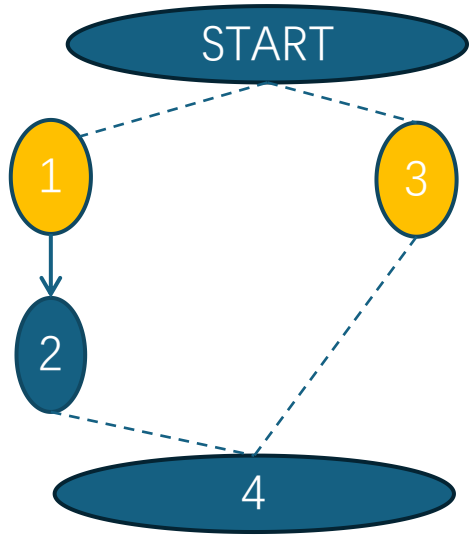
用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

最小化最后一个结点的开工时间，即：

$$\text{最小化：} t_{end} = \sum_{l=1}^{\lambda+1} l \cdot x_{end,l}$$

$$\text{最小化：} (1*x_{4,1} + \cdots + (\lambda+1)*x_{4,\lambda+1})$$



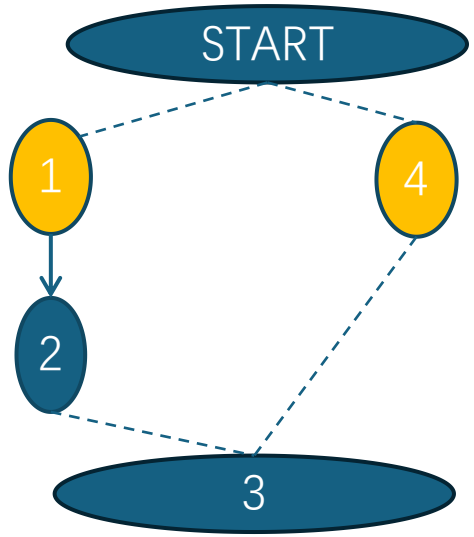
用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

最小化最后一个结点的开工时间，即：

$$\text{最小化：} t_{end} = \sum_{l=1}^{\lambda+1} l \cdot x_{end,l}$$

$$\text{最小化：} (1*x_{3,1} + \cdots + (\lambda+1)*x_{3,\lambda+1})$$



用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

- 1. 确定待求解的变量
- 条件:
  - 2. 唯一约束
  - 3. 顺序约束
  - 4. 资源约束
- 目标:
  - 5. 最小化周期时间

对每个操作i，都有：$\sum_{l=1}^{\lambda+1} x_{il} = 1$

对每条边$(v_j, v_i) \in E$，都有$t_i \geqslant t_j + d_j$

每个操作i的开工周期ti的计算公式为：$\sum_{l=1}^{\lambda+1} l * x_{il}$

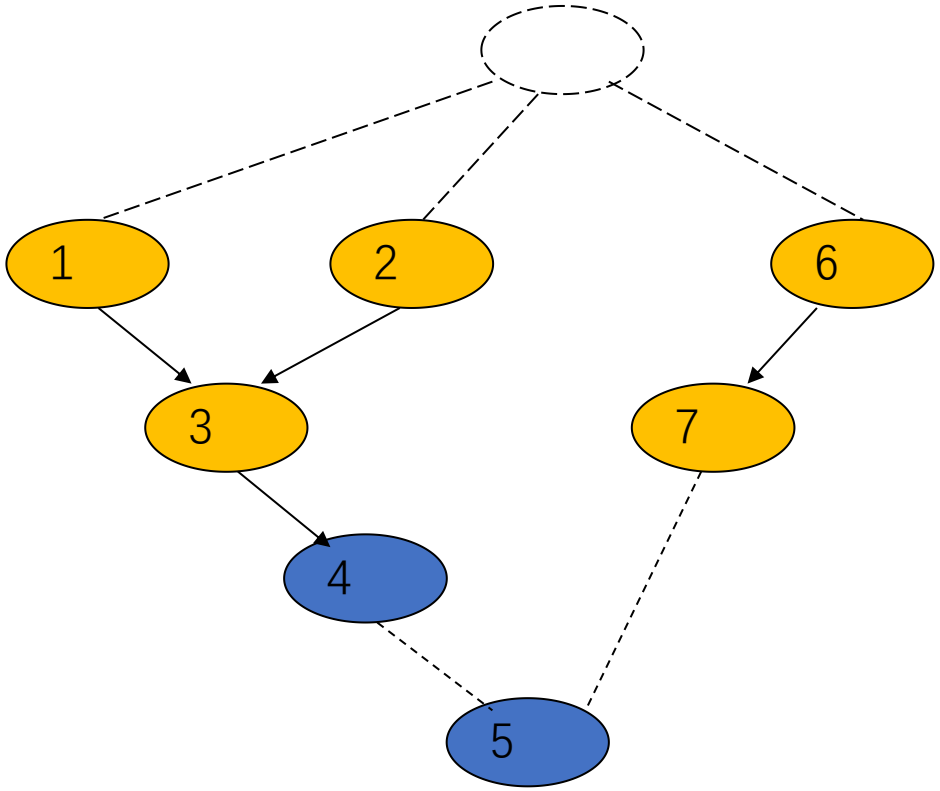
对每种资源k，都在每个周期l列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \leq a_k$$

最小化：$t_{end} = \sum_{l=1}^{\lambda+1} l \cdot X_{end,l}$

随堂作业

in-class assignment



黄色是乘法操作，蓝色是加法操作，

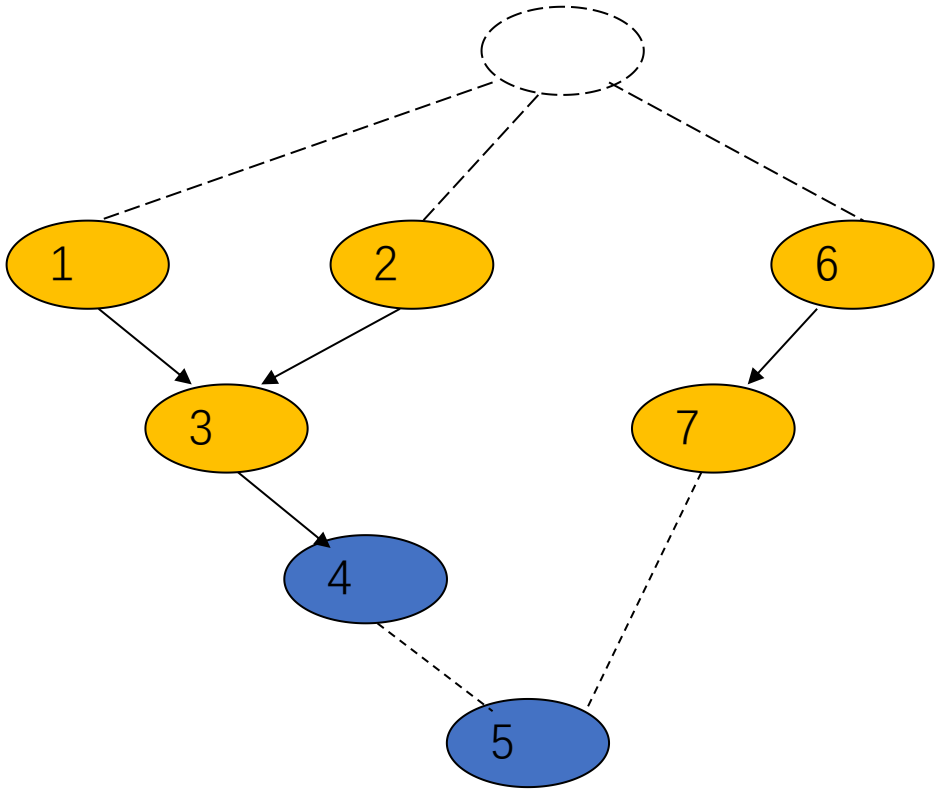
假设有两个乘法器和两个加法器，

λ是6，且乘法和加法的延迟都是2，

请列出用ILP表示的方法

随堂作业-唯一约束

λ是6，所以λ+1是7



对每个操作i，都有：$\sum_{l=1}^{\lambda+1} x_{il} = 1$

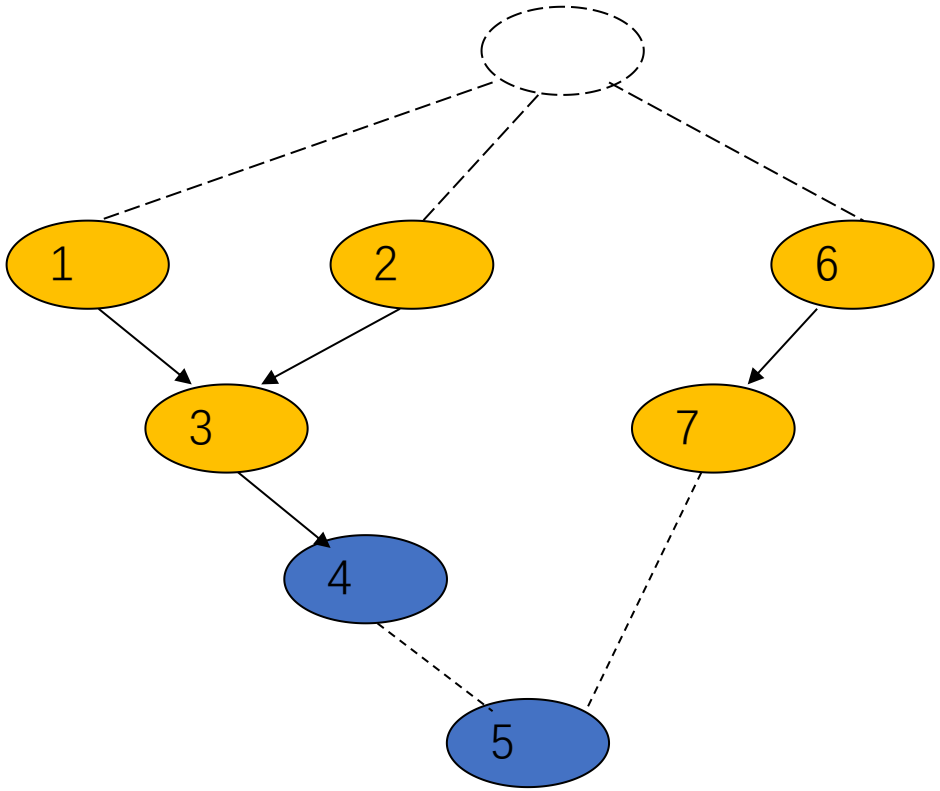
x11+x12+x13+x14+x15+x16+x17=1

…

x71+x72+x73+x74+x75+x76+x77=1

随堂作业-顺序约束

λ是6，所以λ+1是7

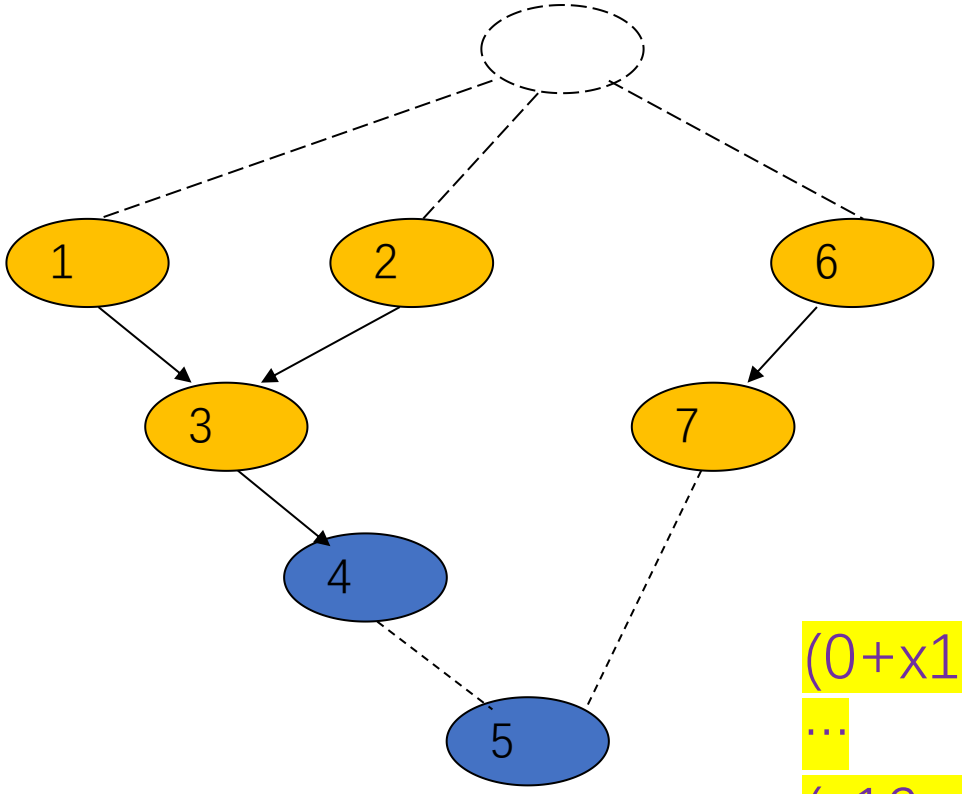


对每条边(vj, vi) ϵ E，都有ti ⩾ tj + dj

每个操作i的开工周期ti的计算公式为：$\sum_{l=1}^{\lambda+1} l * xil$

t3-t1-2>=0
t3-t2-2>=0
t4-t3-2>=0
t5-t4-2>=0
t7-t6-2>=0
t5-t7-2>=0
t1=1*x11+2*x12+3*x13+4*x14+5*x15+6*x16+7*x17
…
t7=1*x71+2*x72+3*x73+4*x74+5*x75+6*x76+7*x77

随堂作业-资源约束



对每种资源k，都在每个周期l列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \le a_k$$

(0+x11)+(0+x21)+(0+x31)+(0+x61)+(0+x71)<=2

…

(x16+x17)+(x26+x27)+(x36+x37)+(x66+x67)+(x76+x77)<=2

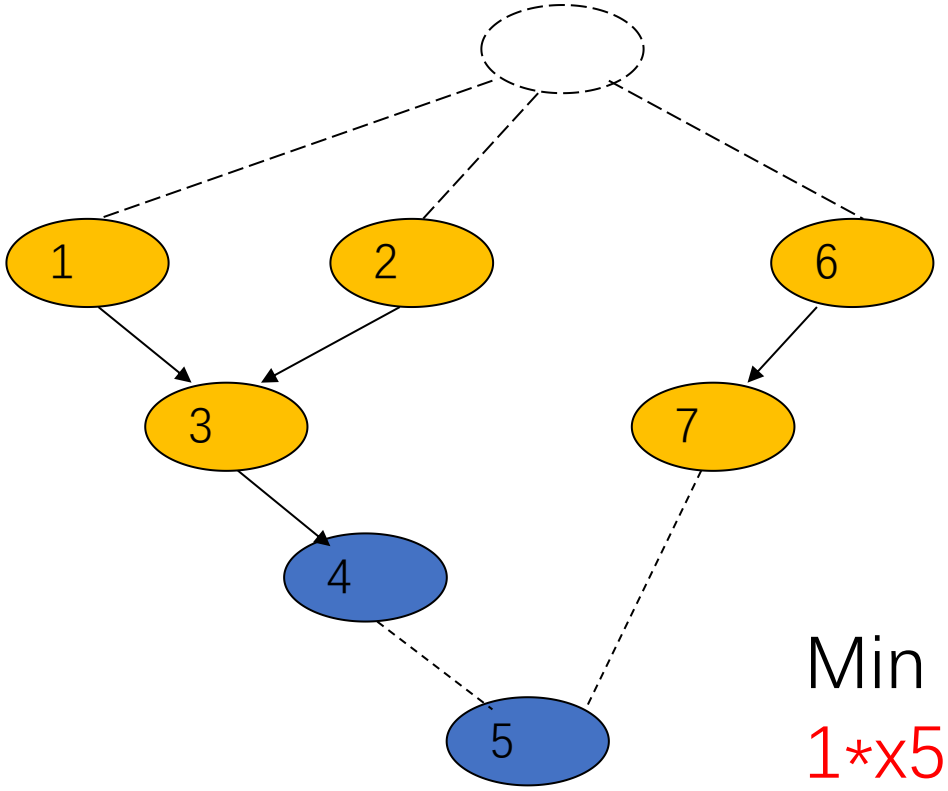
λ是6，所以λ+1是7

(0+x41)+(0+x51)<=2

…

(x46+x47)+(x56+x57)<=2

随堂作业-目标



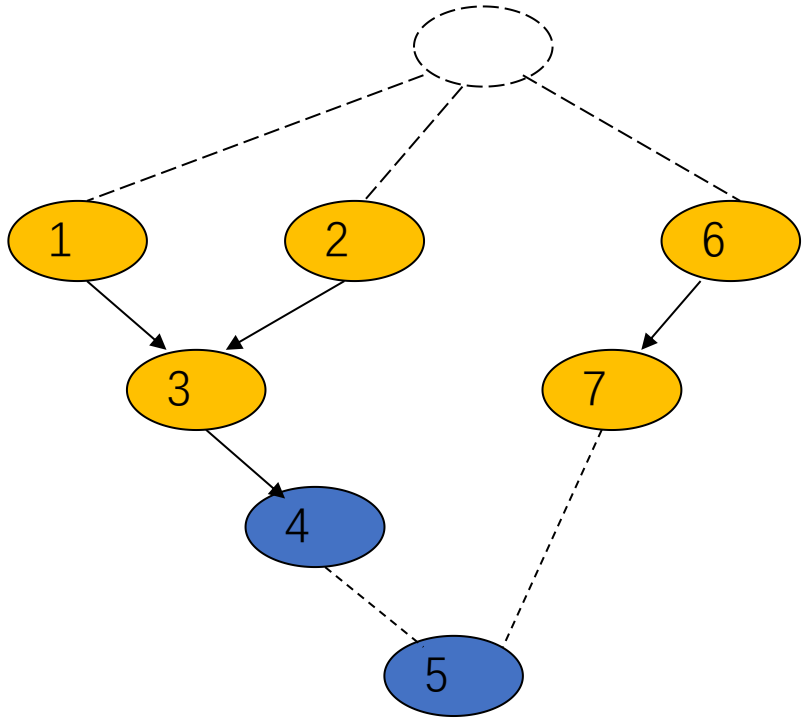
$$\text{最小化: } t_{end} = \sum_{l=1}^{\lambda+1} l \cdot X_{end,l}$$

Min

1*x51+2*x52+3*x53+4*x54+5*x55+6*x56+7*x57

λ是6，所以λ+1是7

- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化最后一个结点的开工时间



Min

1*x51+2*x52+3*x53+4*x54+5*x55+6*x56+7*x57

Subject to

x11+x12+x13+x14+x15+x16+x17=1 …

x71+x72+x73+x74+x75+x76+x77=1

t1=1*x11+2*x12+3*x13+4*x14+5*x15+6*x16+7*x17 …

t7=1*x71+2*x72+3*x73+4*x74+5*x75+6*x76+7*x77

t3-t1-2>=0

t3-t2-2>=0

t4-t3-2>=0

t5-t4-2>=0

t7-t6-2>=0

t5-t7-2>=0

(0+x11)+(0+x21)+(0+x31)+(0+x61)+(0+x71)<=2 …

(x16+x17)+(x26+x27)+(x36+x37)+(x66+x67)+(x76+x77)<=2

(0+x41)+(0+x51)<=2 …

(x46+x47)+(x56+x57)<=2

Binary

x11 x12 x13 x14 x15 x16 x17…

x71 x72 x73 x74 x75 x76 x77

λ是6，且乘法和加法的延迟都是2

用ILP来形式化MR-LCS问题

用ILP来形式化MR-LCS问题

ILP Formulation of ML-RCS

- 1. 确定待求解的变量
- 条件:
  - 2. 唯一约束（每个操作只会在一个周期开工）
  - 3. 顺序约束（当两个结点间有单向边时，代表要完成前一个操作后一个操作才能开工）
  - 4. 时间约束（必须在规定的时间内完成调度）
- 目标:
  - 5. 最小化资源面积

用ILP来形式化MR-LCS问题

ILP Formulation of ML-RCS

- 4. 时间约束（必须在规定的时间内完成调度）

$$t_{end} <= \lambda + 1$$

用ILP来形式化MR-LCS问题

ILP Formulation of ML-RCS

- 目标：
  - 5. 最小化资源面积

对每个周期l，都对每种资源k列出约束：

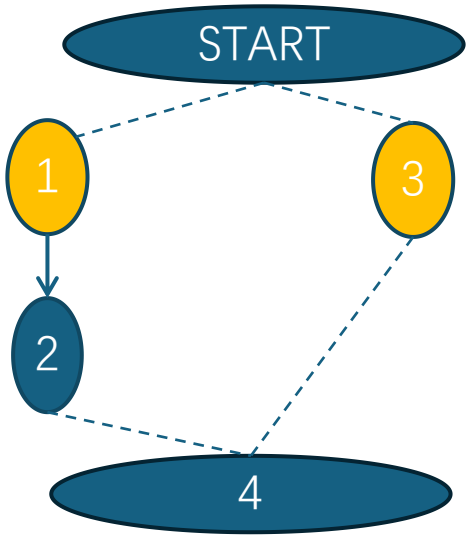
$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \leq a_k$$

  - 最小化：$c^T a$

c^T → 权重

用ILP来形式化MR-LCS问题

ILP Formulation of ML-RCS



- 目标：最小化资源面积

对每个周期l，都对每种资源k列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \le a_k$$

最小化：$c^T a$

假设：c = [5 , 1]

对周期1：$(x_{1,0}+x_{1,1})+(x_{3,0}+x_{3,1}) <= a_1$ 且 $x_{2,1}+x_{4,1} <= a_2$

…

对周期λ + 1：$(x_{1,\lambda}+x_{1,\lambda+1})+(x_{1,\lambda}+x_{3,\lambda+1}) <= a_1$ 且 $x_{2,\lambda+1}+x_{4,\lambda+1} <= a_2$

最小化：$5*a_1 + 1*a_2$

用ILP来形式化MR-LCS问题

ILP Formulation of ML-RCS

- 1. 确定待求解的变量
- 条件:
  - 2. 唯一约束
  - 3. 顺序约束
  - 4. 时间约束
- 目标:
  - 5. 最小化资源面积

对每个操作i，都有：$\sum_{l=1}^{\lambda+1} x_{il} = 1$

对每条边$(v_j, v_i) \in E$，都有$t_i \geqslant t_j + d_j$

每个操作i的开工周期ti的计算公式为：$\sum_{l=1}^{\lambda+1} l * x_{il}$

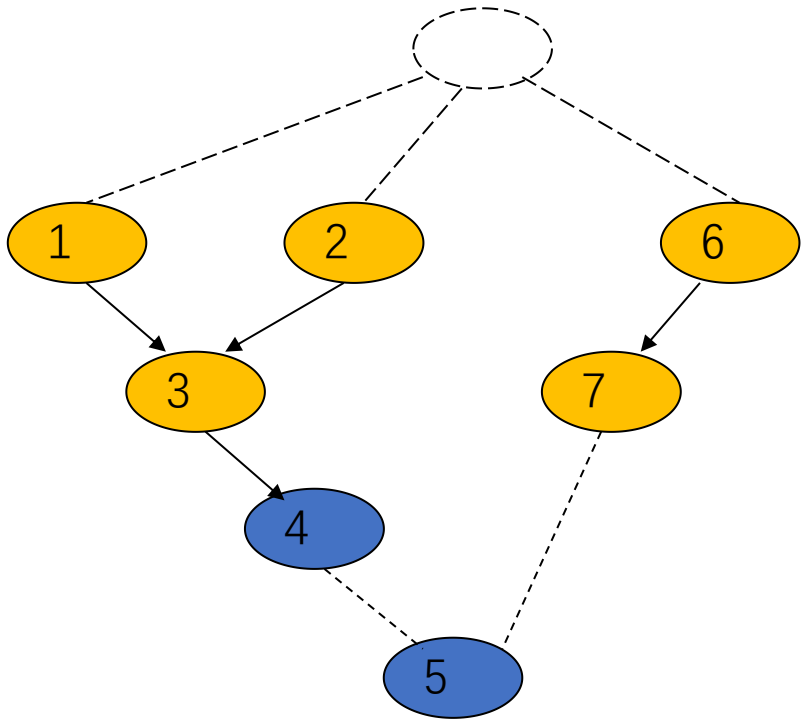
$t_{end} <= \lambda + 1$

对每个周期l，都对每种资源k列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \leq a_k$$

最小化：$c^T a$

随堂作业



λ是6，有两种资源，
延迟都为2，分别是：
乘法器，所占面积为3；
加法器，所占面积为2。

唯一约束

对每个操作i，都有：$\sum_{l=1}^{\lambda+1} x_{il} = 1$

顺序约束

对每条边$(v_j, v_i) \in E$，都有$t_i \geqslant t_j + d_j$

每个操作i的开工周期ti的计算公式为：$\sum_{l=1}^{\lambda+1} l * x_{il}$

时间约束

$t_{end} <= \lambda + 1$

对每个周期l，都对每种资源k列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \leq a_k$$

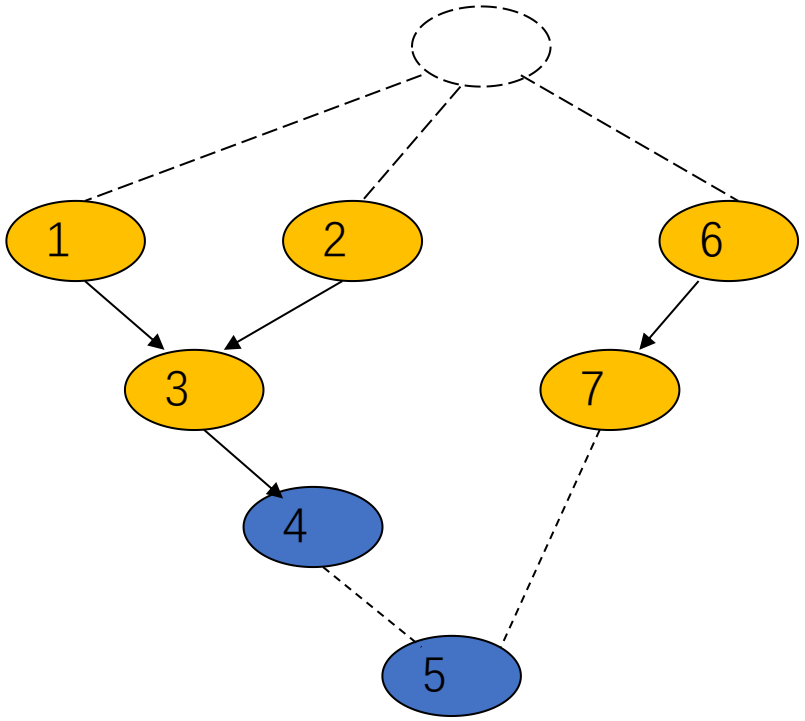
最小化：$c^T a$

最小化资源面积

随堂作业

- 用二进制变量集合来表示最终周期调度情况
- 条件：
  - 唯一约束
  - 顺序约束
  - 时间约束
- 目标：
  - 最小化资源面积

λ是6，有两种资源，分别：是乘法器，所占面积为3；加法器：所占面积为2



Min

3*a1+2*a2

Subject to

x11+x12+x13+x14+x15+x16+x17=1 …

x71+x72+x73+x74+x75+x76+x77=1

t1=1*x11+2*x12+3*x13+4*x14+5*x15+6*x16+7*x17 …

t7=1*x71+2*x72+3*x73+4*x74+5*x75+6*x76+7*x77

t3-t1-2>=0

t3-t2-2>=0

t4-t3-2>=0

t5-t4-2>=0

t7-t6-2>=0

t5-t7-2>=0

t5<=7

(0+x11)+(0+x21)+(0+x31)+(0+x61)+(0+x71)<=a1 …

(x16+x17)+(x26+x27)+(x36+x37)+(x66+x67)+(x76+x77)<=a1

(0+x41)+(0+x51)<=a2…

(x46+x47)+(x56+x57)<=a2

Binary

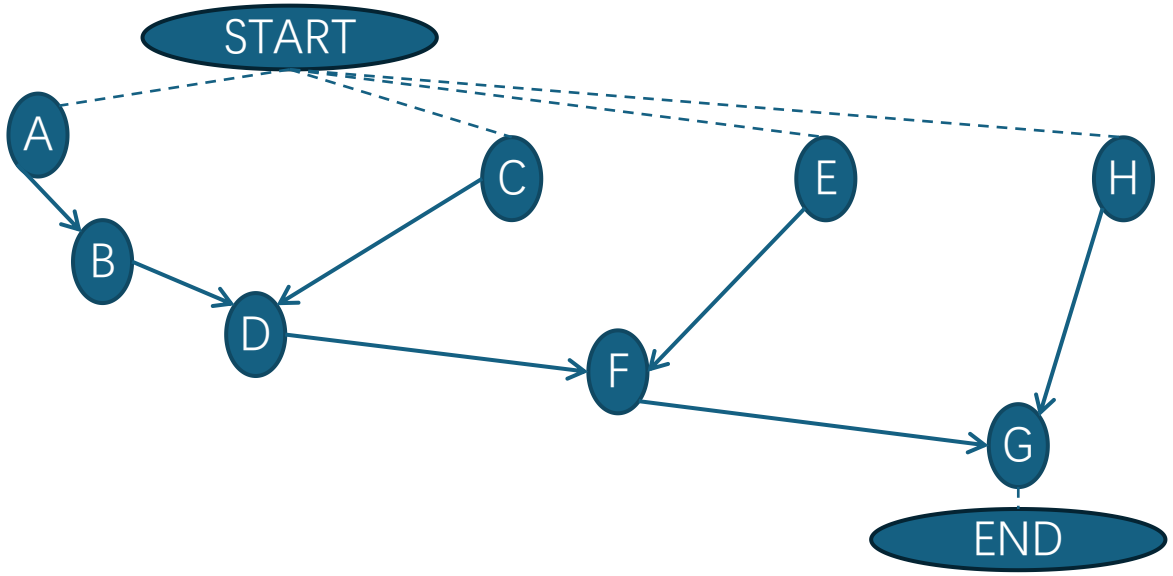
x10 x11 x12 x13 x14 x15 x16 x17…

x70 x71 x72 x73 x74 x75 x76 x77

用代码将调度问题转换为ILP

ASAP调度算法

ASAP scheduling algorithm

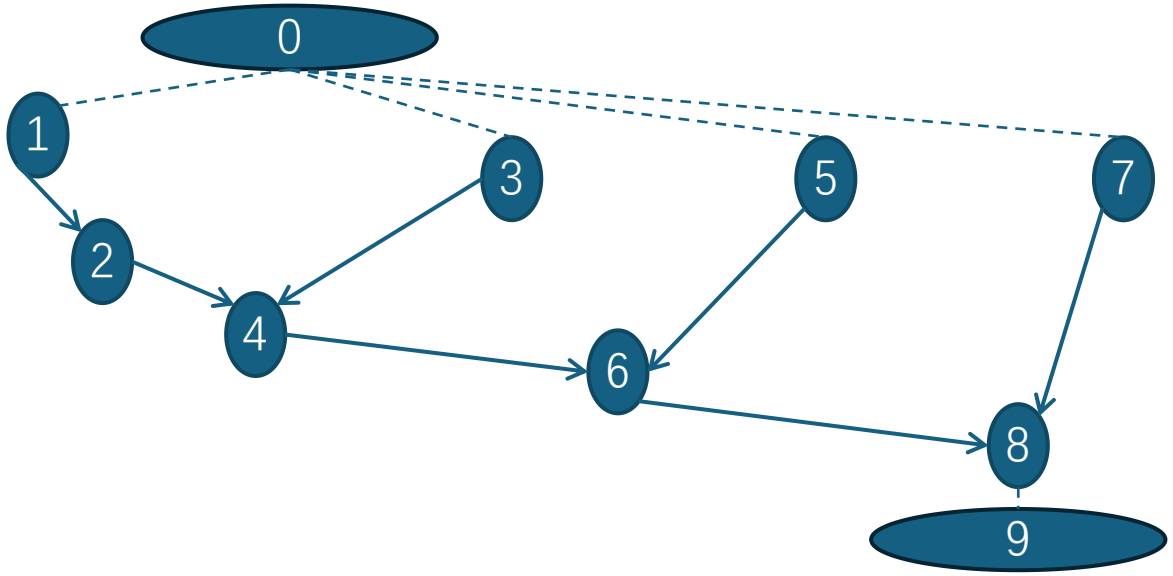


10
START 1
A 1
B 1
C 1
D 1
E 1
F 1
G 1
H 1
END 1

START A
START C
START E
START G
A B
C D
B D
E F
D F
H G
F G
G END

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

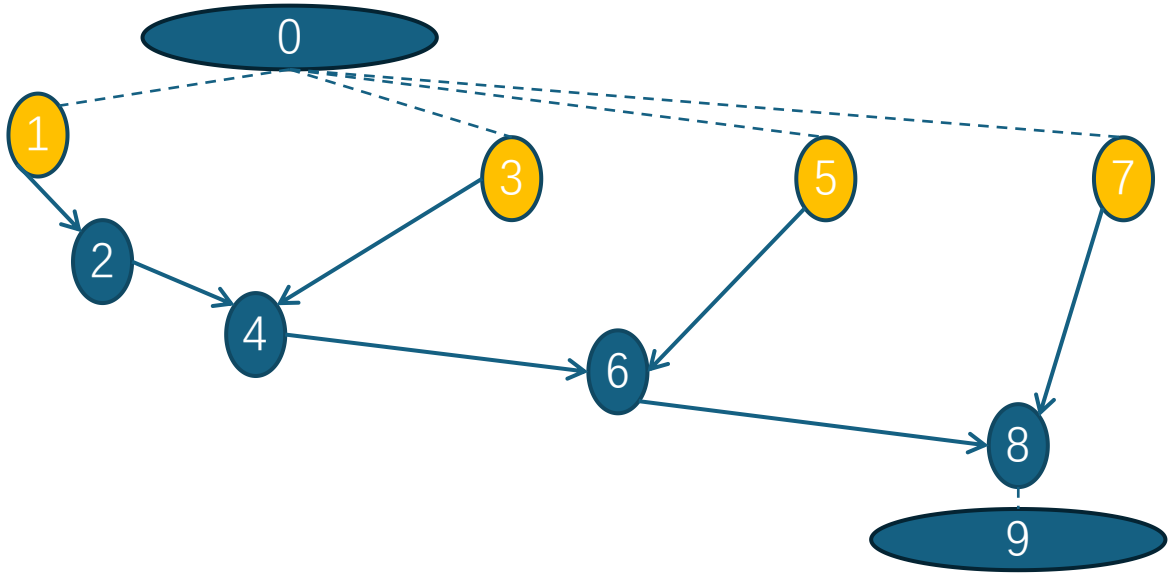


10
0 1
1 1
2 1
3 1
4 1
5 1
6 1
8 1
7 1
9 1

0 1
0 3
0 5
0 7
1 2
3 4
2 4
5 6
4 6
7 8
6 8
8 9

用ILP来形式化ML-RCS问题

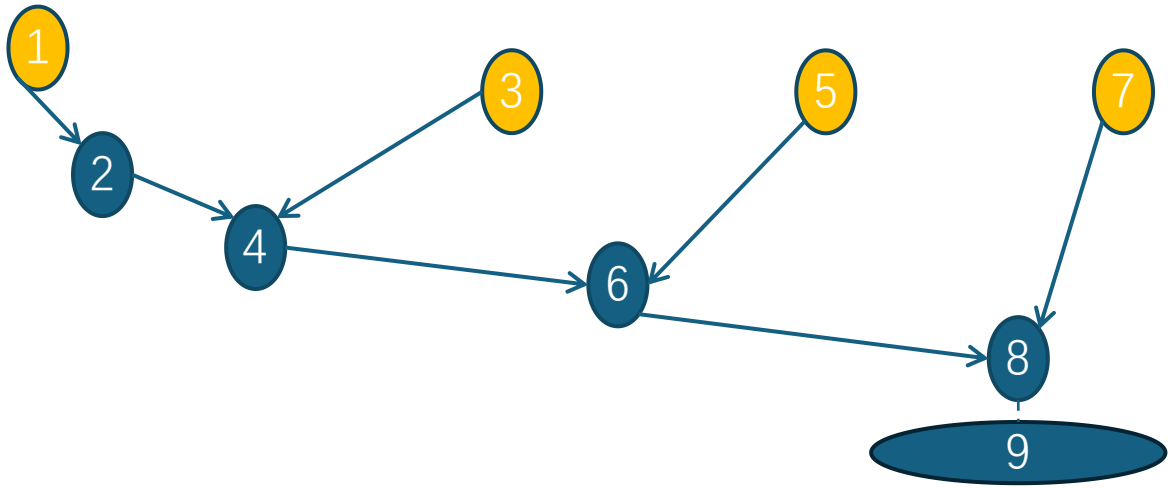
ILP Formulation of ML-RCS




```
10                  0 1
0 1 Adder           0 3
1 2 Multiplier      0 5
2 1 Adder           0 7
3 2 Multiplier      1 2
4 1 Adder           3 4
5 2 Multiplier      2 4
6 1 Adder           5 6
8 1 Adder           4 6
7 2 Multiplier      7 8
9 1 Adder           6 8
                    8 9
```

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS




```
9                    1 2
1 2 Multiplier       3 4
2 1 Adder            2 4
3 2 Multiplier       5 6
4 1 Adder            4 6
5 2 Multiplier       7 8
6 1 Adder            6 8
8 1 Adder            8 9
7 2 Multiplier
9 1 Adder
```

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

```
def read_graph_from_file(input_file):
    with open(input_file, "r", encoding="utf-8") as f:
        N = int(f.readline().strip())  # 读取节点数
        nodes = {}
        # 读取节点信息
        for _ in range(N):
            parts = f.readline().strip().split()
            node_name = parts[0]
            duration = int(parts[1])
            resource = parts[2]
            nodes[node_name] = Node(node_name, duration, resource)
        # 读取前驱关系
        for line in f:
            src, dst = line.strip().split()
            nodes[dst].predecessors.append(nodes[src])
    return list(nodes.values())
```

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

```
class Node:
    def __init__(self, name, duration,resource):
        self.name = name
        self.duration = duration
        self.resource = resource
        self.predecessors = []  # 存储前驱任务
```

用ILP来形式化ML-RCS问题

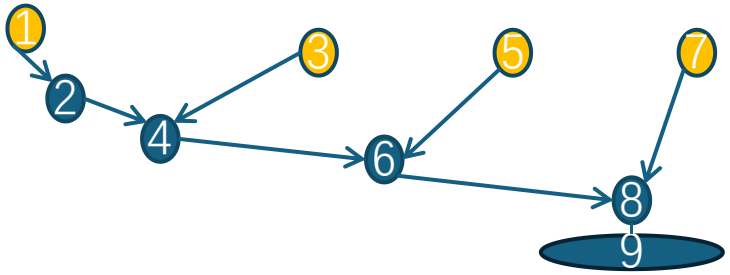
ILP Formulation of ML-RCS

```
if __name__ == "__main__":
    input_file = "input.txt"
    lambda_max = 6  # 预设的最大可能调度时间
    resource_constraints = {"Multiplier": 2, "Adder": 1}  # 资源约束
    # 读取 DAG 任务图
    nodes = read_graph_from_file(input_file)
    # 生成 ILP 约束
    ilp_output = generate_ilp_constraints(nodes, lambda_max, resource_constraints)
    # 输出 ILP 公式
    print(ilp_output)
```

用ILP来形式化ML-RCS问题

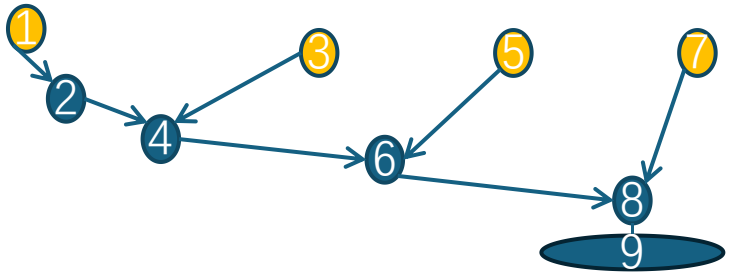
ILP Formulation of ML-RCS

```
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    constraints.append("\nBinary")
    binary_vars=""
    for node in nodes:
        for l in range(1, lambda_max + 2):
            binary_vars = binary_vars+"x"+node.name+str(l)+" "
    constraints.append(binary_vars)
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```



x11 x12 x13 x14 x15 x16 x27 x21 x22 x23 x24 x25 x26 x27…

```python
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    constraints.append("\nBinary")
    binary_vars=""
    for node in nodes:
        for l in range(1, lambda_max + 2):
            binary_vars = binary_vars+"x"+node.name+str(l)+" "
    constraints.append(binary_vars)
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```



用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

```python
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    # 目标函数：最小化 end 任务的调度时间
    end_node = nodes[-1].name # 假设最后一个节点是 `end`
    objective = "Min\n" + " + ".join([f"{l}*x{end_node}{l}" for l in range(1, lambda_max + 2)])
    constraints.append(objective)
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```

最小化: $t_{end} = \sum_{l=1}^{\lambda+1} l \cdot X_{end,l}$

Min
1*x91 + 2*x92 + 3*x93 + 4*x94 + 5*x95 + 6*x96 + 7*x97

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

```
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    constraints.append("\nSubject to")
    for node in nodes:
        constraint = " + ".join([f"x{node.name}{l}" for l in range(1, lambda_max + 2)]) + "=1"
        constraints.append(constraint)
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```

对每个操作i，都有：$\sum_{l=1}^{\lambda+1} x_{il} = 1$

```python
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    for node in nodes:
        for pred in node.predecessors:
            constraint = f"t{node.name} >= t{pred.name} + {pred.duration}"
            constraints.append(constraint)
    for node in nodes:
        t_constraint = f"t{node.name} ="+"+".join([f"{l}*x{node.name}{l}" for l in range(1, lambda_max+ 2)])
        constraints.append(t_constraint)
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```

对每条边$(v_j, v_i) \in E$，都有$t_i \geqslant t_j + d_j$

每个操作i的开工周期ti的计算公式为：$\sum_{l=1}^{\lambda+1} l * x_{il}$

```python
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    for resource, max_available in resource_constraints.items():
        for l in range(1, lambda_max + 2):
            resource_usage = []
            for node in nodes:
                if node.resource == resource:  # 假设资源信息在约束表里
                    for m in range(max(1, l - node.duration + 1), l + 1):
                        resource_usage.append(f"x{node.name}{m}")
            if resource_usage:
                constraint = " + ".join(resource_usage) + f" <= {max_available}"
                constraints.append(constraint)
    return "\n".join(constraints)
```

对每种资源k，都在每个周期l列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \le a_k$$

Binary

x11 x12 x13 x14 x15 x16 x17 x21 x22 x23 x24 x25 x26 x27 x31 x32 x33 x34
x35 x36 x37 x41 x42 x43 x44 x45 x46 x47 x51 x52 x53 x54 x55 x56 x57 x61
x62 x63 x64 x65 x66 x67 x81 x82 x83 x84 x85 x86 x87 x71 x72 x73 x74 x75
x76 x77 x91 x92 x93 x94 x95 x96 x97

Min

1*x91 + 2*x92 + 3*x93 + 4*x94 + 5*x95 + 6*x96 + 7*x97

Subject to

x11 + x12 + x13 + x14 + x15 + x16 + x17 = 1

x21 + x22 + x23 + x24 + x25 + x26 + x27 = 1

x31 + x32 + x33 + x34 + x35 + x36 + x37 = 1

x41 + x42 + x43 + x44 + x45 + x46 + x47 = 1

x51 + x52 + x53 + x54 + x55 + x56 + x57 = 1

x61 + x62 + x63 + x64 + x65 + x66 + x67 = 1

x81 + x82 + x83 + x84 + x85 + x86 + x87 = 1

x71 + x72 + x73 + x74 + x75 + x76 + x77 = 1

x91 + x92 + x93 + x94 + x95 + x96 + x97 = 1

t1 = 1*x11 + 2*x12 + 3*x13 + 4*x14 + 5*x15 + 6*x16 + 7*x17
t2 = 1*x21 + 2*x22 + 3*x23 + 4*x24 + 5*x25 + 6*x26 + 7*x27
t3 = 1*x31 + 2*x32 + 3*x33 + 4*x34 + 5*x35 + 6*x36 + 7*x37
t4 = 1*x41 + 2*x42 + 3*x43 + 4*x44 + 5*x45 + 6*x46 + 7*x47
t5 = 1*x51 + 2*x52 + 3*x53 + 4*x54 + 5*x55 + 6*x56 + 7*x57
t6 = 1*x61 + 2*x62 + 3*x63 + 4*x64 + 5*x65 + 6*x66 + 7*x67
t8 = 1*x81 + 2*x82 + 3*x83 + 4*x84 + 5*x85 + 6*x86 + 7*x87
t7 = 1*x71 + 2*x72 + 3*x73 + 4*x74 + 5*x75 + 6*x76 + 7*x77
t9 = 1*x91 + 2*x92 + 3*x93 + 4*x94 + 5*x95 + 6*x96 + 7*x97
t2 - t1 - 2 >= 0
t4 - t3 - 2 >= 0
t4 - t2 - 1 >= 0
t6 - t5 - 2 >= 0
t6 - t4 - 1 >= 0
t8 - t7 - 2 >= 0
t8 - t6 - 1 >= 0
t9 - t8 - 1 >= 0
x11 + x31 + x51 + x71 <= 2
x11 + x12 + x31 + x32 + x51 + x52 + x71 + x72 <= 2
x12 + x13 + x32 + x33 + x52 + x53 + x72 + x73 <= 2
x13 + x14 + x33 + x34 + x53 + x54 + x73 + x74 <= 2
x14 + x15 + x34 + x35 + x54 + x55 + x74 + x75 <= 2
x15 + x16 + x35 + x36 + x55 + x56 + x75 + x76 <= 2
x16 + x17 + x36 + x37 + x56 + x57 + x76 + x77 <= 2
x21 + x41 + x61 + x81 + x91 <= 1
x22 + x42 + x62 + x82 + x92 <= 1
x23 + x43 + x63 + x83 + x93 <= 1
x24 + x44 + x64 + x84 + x94 <= 1
x25 + x45 + x65 + x85 + x95 <= 1
x26 + x46 + x66 + x86 + x96 <= 1
x27 + x47 + x67 + x87 + x97 <= 1

直接引入求解器的实现

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

```
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 生成ILP模型并定义变量
    model = pulp.LpProblem("ML-RCS_Scheduling", pulp.LpMinimize)
    x = {
        (node.name, l): pulp.LpVariable(f"x_{node.name}_{l}", cat="Binary")
        for node in nodes for l in range(1, lambda_max + 2)
    }
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

```
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    # 目标函数：最小化 end 任务的调度时间
    end_node = nodes[-1].name  # 假设最后一个节点是 `end`
    model += pulp.lpSum(l * x[end_node, l] for l in range(1, lambda_max + 2)), "Objective"
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```

$$\text{最小化: } t_{end} = \sum_{l=1}^{\lambda+1} l \cdot X_{end,l}$$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

```
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    for node in nodes:
        model += pulp.lpSum(x[node.name, l] for l in range(1, lambda_max + 2)) == 1
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```

对每个操作i，都有：$\sum_{l=1}^{\lambda+1} x_{il} = 1$

```python
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    for node in nodes:
        for pred in node.predecessors:
            model += t[node.name] >= t[pred.name] + pred.duration
    for node in nodes:
        model += t[node.name] == pulp.lpSum(l*x[node.name, l] for l in range(1, lambda_max + 2))
    # 资源约束：确保同一时刻资源不超限
    return "\n".join(constraints)
```

对每条边$(v_j, v_i) \in E$，都有$t_i \geqslant t_j + d_j$

每个操作i的开工周期ti的计算公式为：$\sum_{l=1}^{\lambda+1} l * x_{il}$

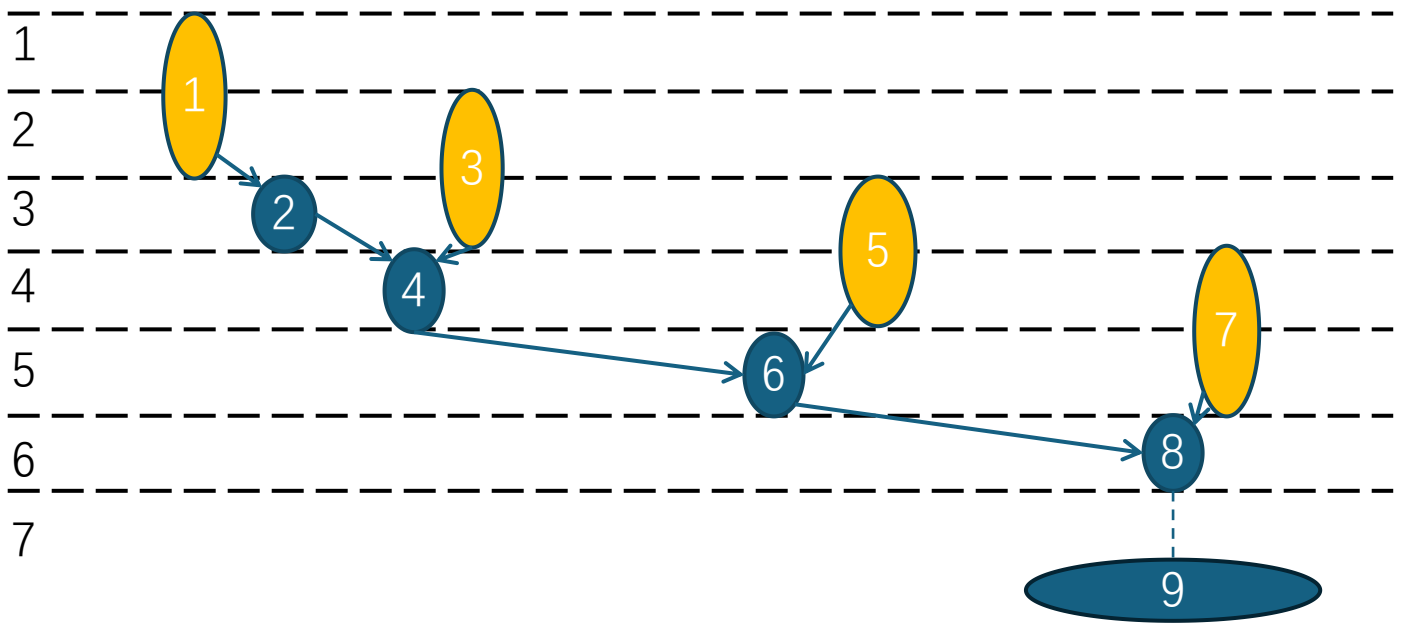
```python
def generate_ilp_constraints(nodes, lambda_max, resource_constraints):
    constraints = []
    # 变量类型
    # 目标函数：最小化 end 任务的调度时间
    # 唯一约束：每个任务只能在一个时间点开工
    # 顺序约束：确保前驱任务完成后才能开始
    # 资源约束：确保同一时刻资源不超限
    for resource, max_available in resource_constraints.items():
        for l in range(1, lambda_max + 2):
            resource_usage = [
                x[node.name, m]
                for node in nodes
                if node.resource == resource
                for m in range(max(1, l - node.duration + 1), l + 1)
            ]
            if resource_usage:
                model += pulp.lpSum(resource_usage) <= max_available
```

对每种资源k，都在每个周期l列出约束：

$$\sum_{i:T(v_i)=k} \sum_{m=l-(d_i-1)}^{l} x_{i,m} \le a_k$$

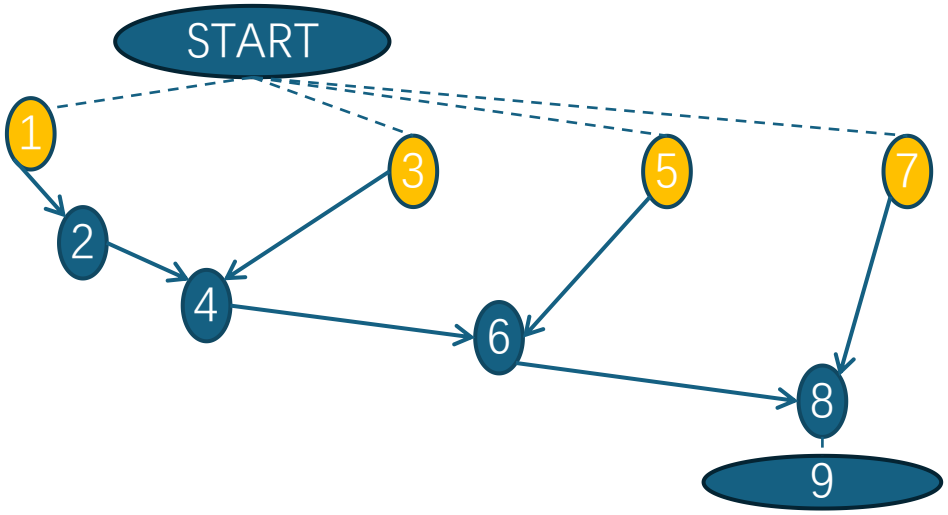
```python
def solve_ilp_model(model, x, t):
    model.solve()
    print("\n=== 任务调度结果 ===")
    for var in model.variables():
        if var.name.startswith("t_"):
            print(f"{var.name}: {int(pulp.value(var))}")
    print("\n=== 最优调度时间 ===")
    print(f"Optimal Makespan: {pulp.value(model.objective)}")
```

```
Result - Optimal solution found

Objective value:                7.00000000
Enumerated nodes:               0
Total iterations:               2
Time (CPU seconds):             0.02
Time (Wallclock seconds):       0.02

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.05   (Wallclock seconds):       0.05


=== 任务调度结果 ===
t_1: 1
t_2: 3
t_3: 2
t_4: 4
t_5: 3
t_6: 5
t_7: 4
t_8: 6
t_9: 7

=== 最优调度时间 ===
Optimal Makespan: 7.0
```



用ILP来形式化 - 优化

- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间



Binary
x11 x12 x13 x14 x15 x16 x17 x21 x22 x23 x24 x25 x26 x27 x31 x32 x33 x34 x35 x36 x37 x41
x42 x43 x44 x45 x46 x47 x51 x52 x53 x54 x55 x56 x57 x61 x62 x63 x64 x65 x66 x67 x81 x82
x83 x84 x85 x86 x87 x71 x72 x73 x74 x75 x76 x77 x91 x92 x93 x94 x95 x96 x97
Min
5*a1+a2
Subject to
x11 + x12 + x13 + x14 + x15 + x16 + x17 = 1
x21 + x22 + x23 + x24 + x25 + x26 + x27 = 1
x31 + x32 + x33 + x34 + x35 + x36 + x37 = 1
x41 + x42 + x43 + x44 + x45 + x46 + x47 = 1
x51 + x52 + x53 + x54 + x55 + x56 + x57 = 1
x61 + x62 + x63 + x64 + x65 + x66 + x67 = 1
x81 + x82 + x83 + x84 + x85 + x86 + x87 = 1
x71 + x72 + x73 + x74 + x75 + x76 + x77 = 1
x91 + x92 + x93 + x94 + x95 + x96 + x97 = 1
t1 = 1*x11 + 2*x12 + 3*x13 + 4*x14 + 5*x15 + 6*x16 + 7*x17
t2 = 1*x21 + 2*x22 + 3*x23 + 4*x24 + 5*x25 + 6*x26 + 7*x27
t3 = 1*x31 + 2*x32 + 3*x33 + 4*x34 + 5*x35 + 6*x36 + 7*x37
t4 = 1*x41 + 2*x42 + 3*x43 + 4*x44 + 5*x45 + 6*x46 + 7*x47
t5 = 1*x51 + 2*x52 + 3*x53 + 4*x54 + 5*x55 + 6*x56 + 7*x57
t6 = 1*x61 + 2*x62 + 3*x63 + 4*x64 + 5*x65 + 6*x66 + 7*x67
t8 = 1*x81 + 2*x82 + 3*x83 + 4*x84 + 5*x85 + 6*x86 + 7*x87
t7 = 1*x71 + 2*x72 + 3*x73 + 4*x74 + 5*x75 + 6*x76 + 7*x77
t9 = 1*x91 + 2*x92 + 3*x93 + 4*x94 + 5*x95 + 6*x96 + 7*x97
t2 - t1 - 2 >= 0
t4 - t3 - 2 >= 0
t4 - t2 - 1 >= 0
t6 - t5 - 2 >= 0
t6 - t4 - 1 >= 0
t8 - t7 - 2 >= 0
t8 - t6 - 1 >= 0
t9 - t8 - 1 >= 0
t9=7
x11 + x31 + x51 + x71 <= a1
(x11+x12)+(x31+x32)+(x51+x52)+(x71+x72)<= a1
(x12+x13)+(x32+x33)+(x52+x53)+(x72+x73)<= a1
(x13+x14)+(x33+x34)+(x53+x54)+(x73+x74)<= a1
(x14+x15)+(x34+x35)+(x54+x55)+(x74+x75)<= a1
(x15+x16)+(x35+x36)+(x55+x56)+(x75+x76)<= a1
(x16+x17)+(x36+x37)+(x56+x57)+(x76+x77)<= a1
x21+x41+x61+x81+x91 <= a2
x22+x42+x62+x82+x92 <= a2
x23+x43+x63+x83+x93 <= a2
x24+x44+x64+x84+x94 <= a2
x25+x45+x65+x85+x95 <= a2
x26+x46+x66+x86+x96 <= a2
x27+x47+x67+x87+x97 <= a2

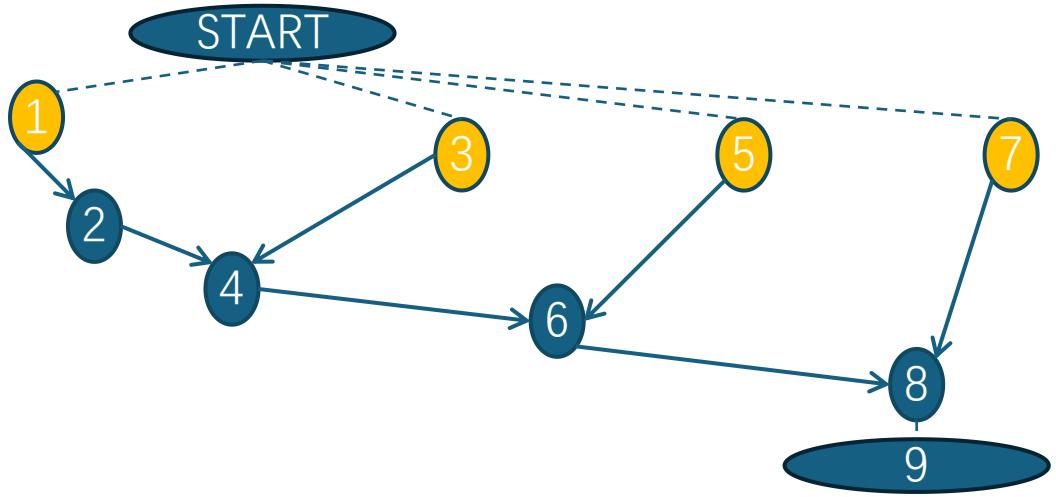
- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

Subject to

x11+x12+x13+x14+x15+x16+x17=1
x21+x22+x23+x24+x25+x26+x27=1
x31+x32+x33+x34+x35+x36+x37=1
x41+x42+x43+x44+x45+x46+x47=1
x51+x52+x53+x54+x55+x56+x57=1
x61+x62+x63+x64+x65+x66+x67=1
x71+x72+x73+x74+x75+x76+x77=1
x81+x82+x83+x84+x85+x86+x87=1
x91+x92+x93+x94+x95+x96+x97=1

Binary

x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77
x81 x82 x83 x84 x85 x86 x87
x91 x92 x93 x94 x95 x96 x97



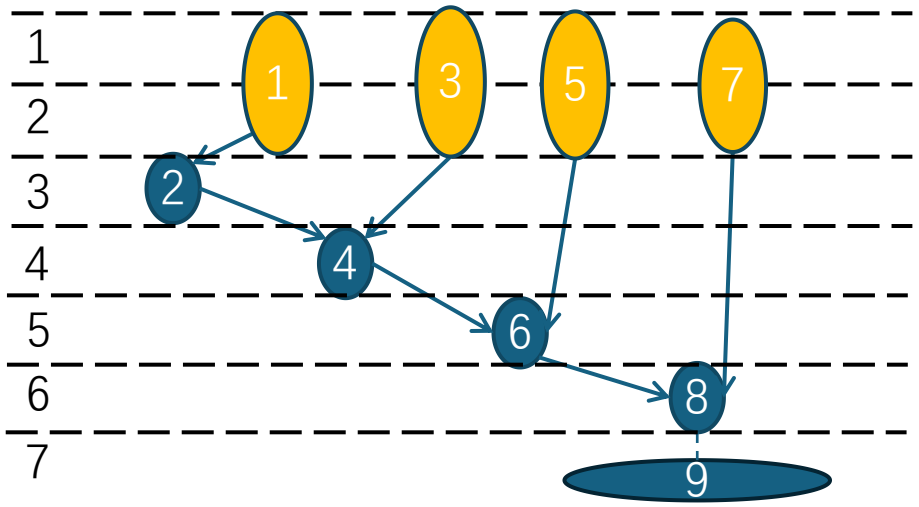
- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

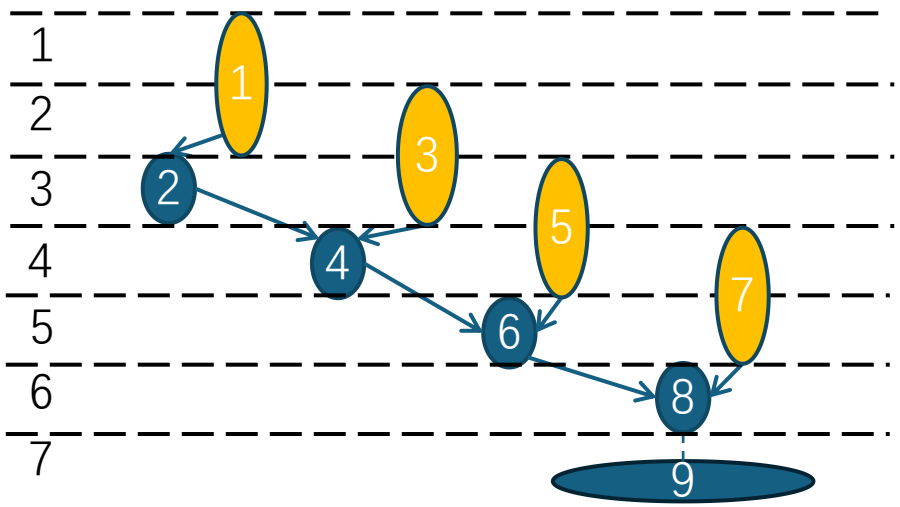
Subject to

x11+x12+x13+x14+x15+x16+x17=1
x21+x22+x23+x24+x25+x26+x27=1
x31+x32+x33+x34+x35+x36+x37=1
x41+x42+x43+x44+x45+x46+x47=1
x51+x52+x53+x54+x55+x56+x57=1
x61+x62+x63+x64+x65+x66+x67=1
x71+x72+x73+x74+x75+x76+x77=1
x81+x82+x83+x84+x85+x86+x87=1
x91+x92+x93+x94+x95+x96+x97=1

Binary

x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77
x81 x82 x83 x84 x85 x86 x87
x91 x92 x93 x94 x95 x96 x97





X1: 1-1
X2: 3-3
X3: 1-2
X4: 4-4
X5: 1-3
X6: 5-5
X7: 1-4
X8: 6-6
X9: 7-7

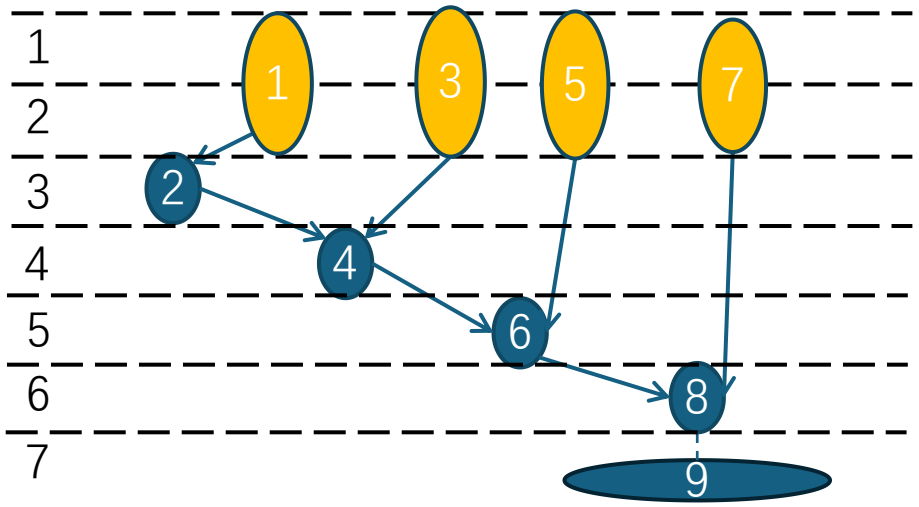
- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

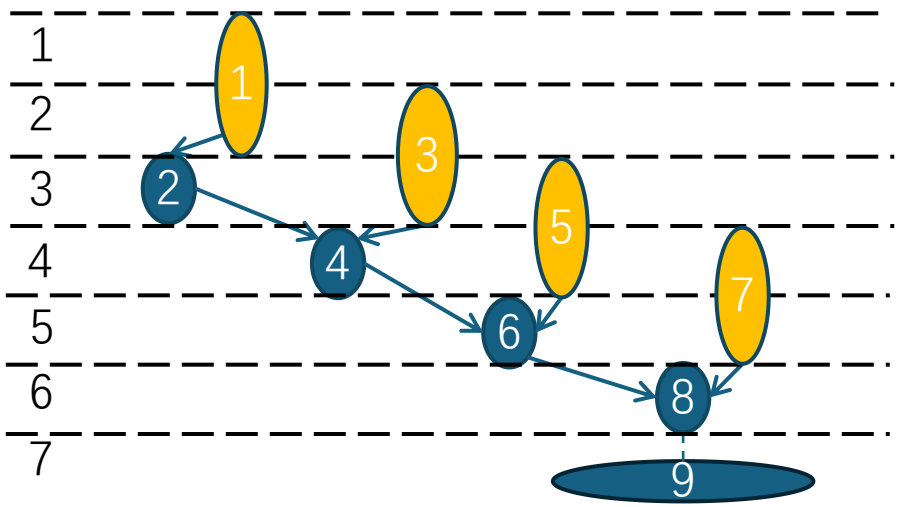
Subject to

x11+x12+x13+x14+x15+x16+x17=1
x21+x22+x23+x24+x25+x26+x27=1
x31+x32+x33+x34+x35+x36+x37=1
x41+x42+x43+x44+x45+x46+x47=1
x51+x52+x53+x54+x55+x56+x57=1
x61+x62+x63+x64+x65+x66+x67=1
x71+x72+x73+x74+x75+x76+x77=1
x81+x82+x83+x84+x85+x86+x87=1
x91+x92+x93+x94+x95+x96+x97=1

Binary

x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77
x81 x82 x83 x84 x85 x86 x87
x91 x92 x93 x94 x95 x96 x97





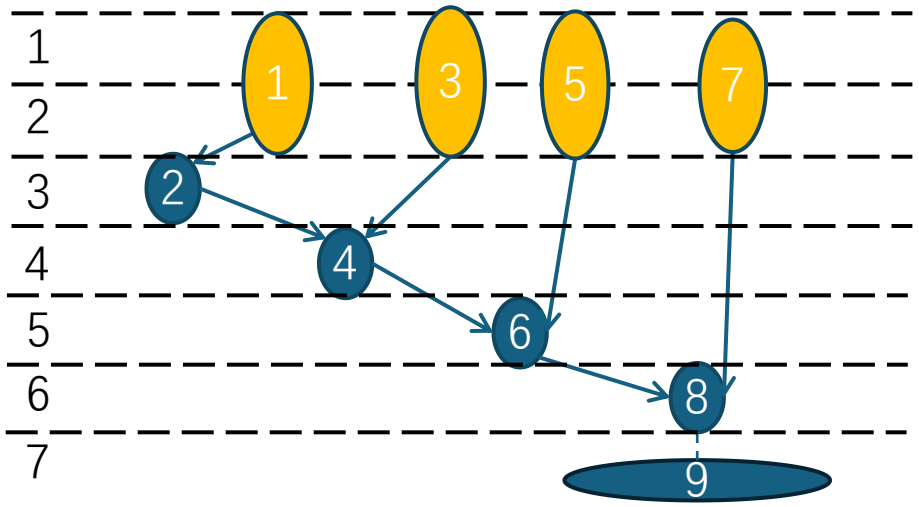
- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

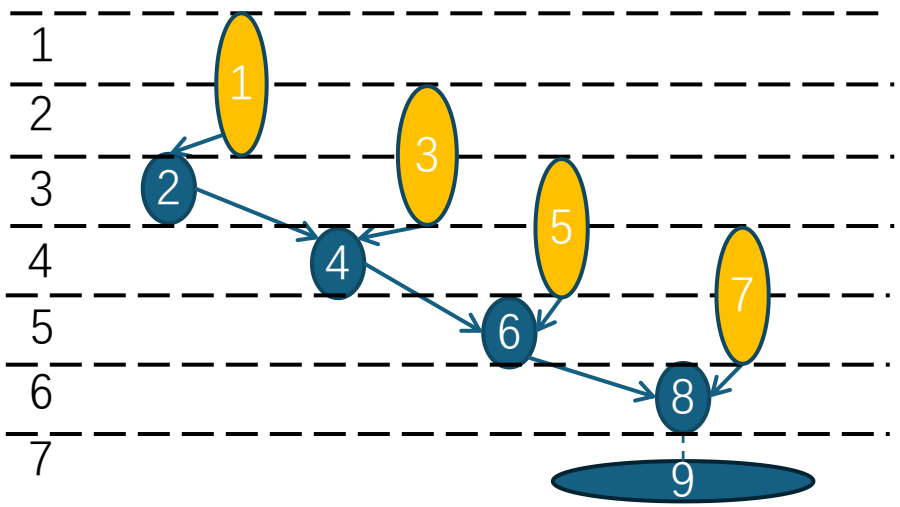
Subject to

x11+x12+x13+x14+x15+x16+x17=1
x21+x22+x23+x24+x25+x26+x27=1
x31+x32+x33+x34+x35+x36+x37=1
x41+x42+x43+x44+x45+x46+x47=1
x51+x52+x53+x54+x55+x56+x57=1
x61+x62+x63+x64+x65+x66+x67=1
x71+x72+x73+x74+x75+x76+x77=1
x81+x82+x83+x84+x85+x86+x87=1
x91+x92+x93+x94+x95+x96+x97=1

Binary

x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77
x81 x82 x83 x84 x85 x86 x87
x91 x92 x93 x94 x95 x96 x97





- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

Subject to

x31+x32=1

x51+x52+x53=1

x71+x72+x73+x74=1

Binary

x11 x12 x13 x14 x15 x16 x17

x21 x22 x23 x24 x25 x26 x27

x31 x32 x33 x34 x35 x36 x37

x41 x42 x43 x44 x45 x46 x47

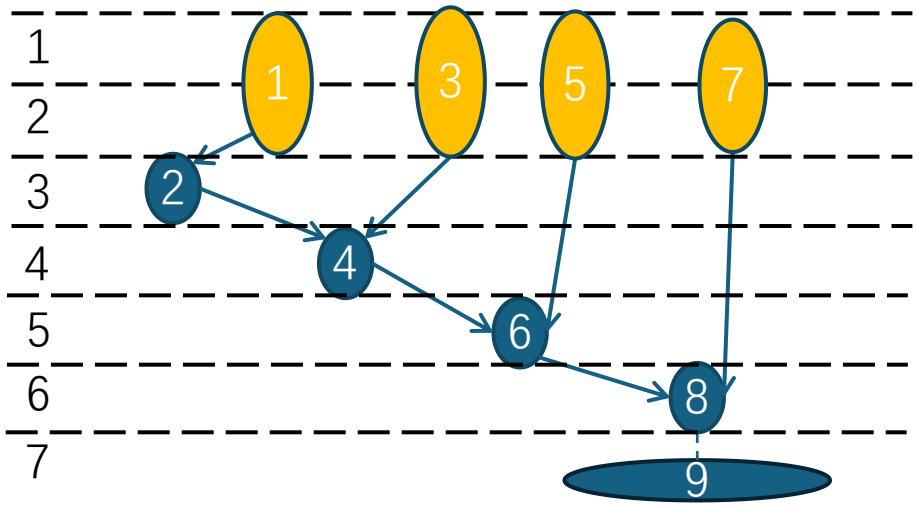
x51 x52 x53 x54 x55 x56 x57

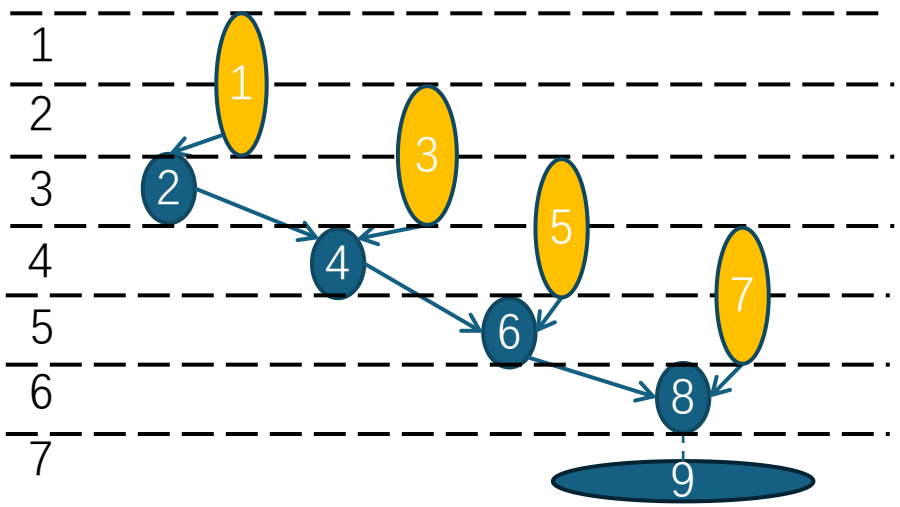
x61 x62 x63 x64 x65 x66 x67

x71 x72 x73 x74 x75 x76 x77

x81 x82 x83 x84 x85 x86 x87

x91 x92 x93 x94 x95 x96 x97





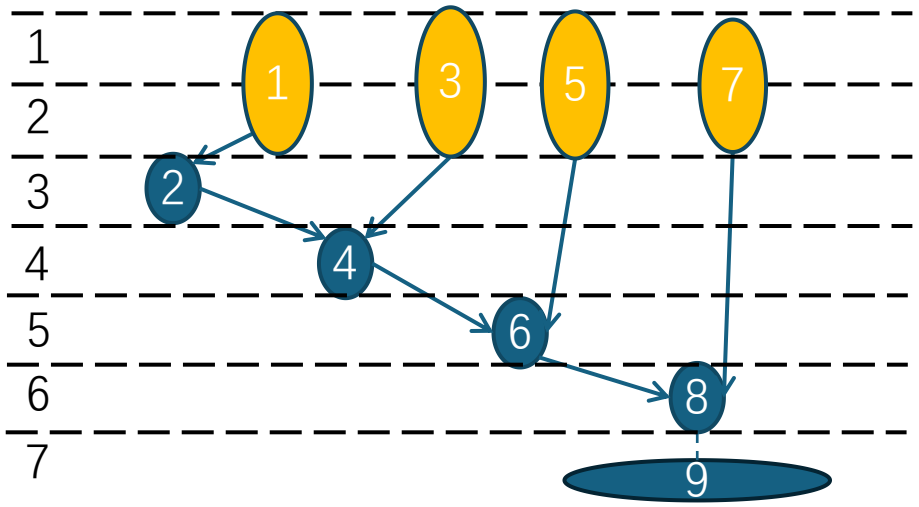
- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

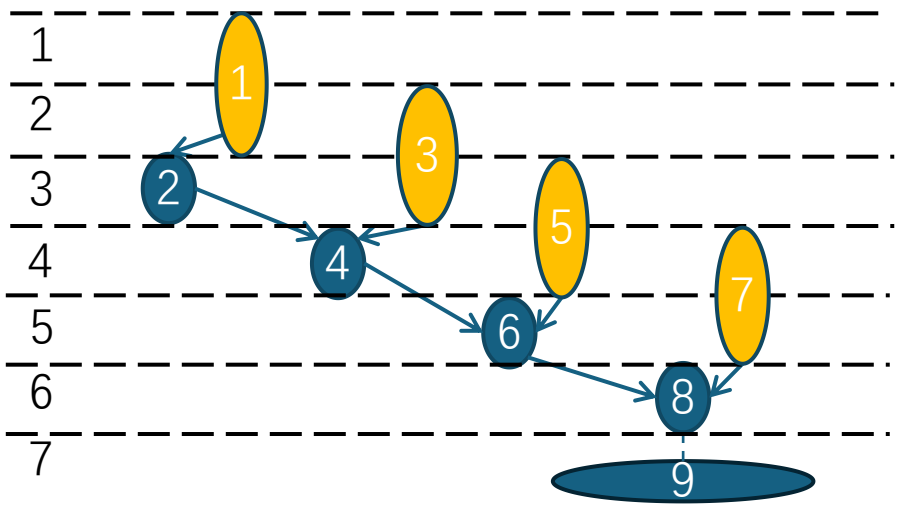
Subject to

t1=1*x11+2*x12+3*x13+4*x14
+5*x15+6*x16+7*x17
t2=1*x21+2*x22+3*x23+4*x24
+5*x25+6*x26+7*x27
t3=1*x31+2*x32+3*x33+4*x34
+5*x35+6*x36+7*x37
…
t9=1*x91+2*x92+3*x93+4*x94
+5*x95+6*x96+7*x97

Binary

x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77
x81 x82 x83 x84 x85 x86 x87
x91 x92 x93 x94 x95 x96 x97





- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

Subject to

t1=1*1+2*0+3*0+4*0+5*0+6*0+7*0

t2=1*x21+2*x22+3*x23+4*x24+5*x25+6*x26+7*x27

t3=1*x31+2*x32+3*x33+4*x34+5*x35+6*x36+7*x37

...

t9=1*x91+2*x92+3*x93+4*x94+5*x95+6*x96+7*x97

Binary

x11 x12 x13 x14 x15 x16 x17

x21 x22 x23 x24 x25 x26 x27

x31 x32 x33 x34 x35 x36 x37

x41 x42 x43 x44 x45 x46 x47

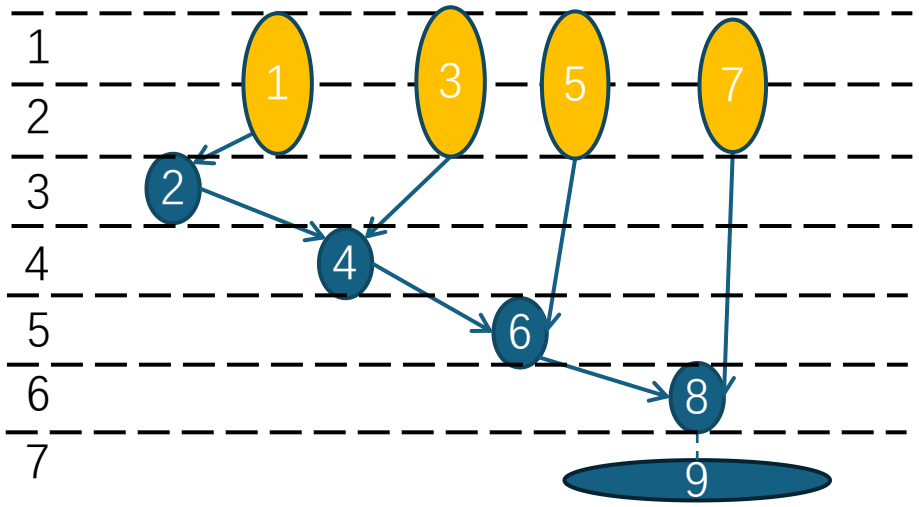
x51 x52 x53 x54 x55 x56 x57

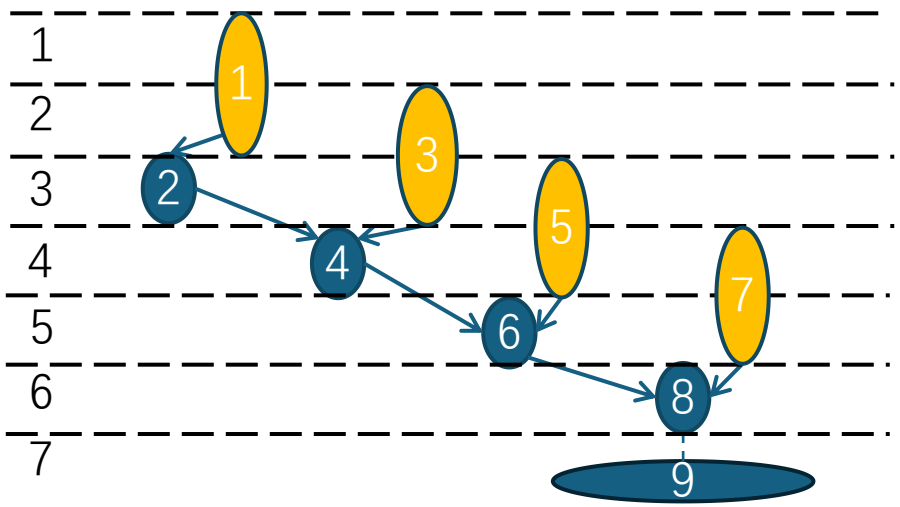
x61 x62 x63 x64 x65 x66 x67

x71 x72 x73 x74 x75 x76 x77

x81 x82 x83 x84 x85 x86 x87

x91 x92 x93 x94 x95 x96 x97





- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

Subject to

t1=1

t2=1*x21+2*x22+3*x23+4*x24
+5*x25+6*x26+7*x27

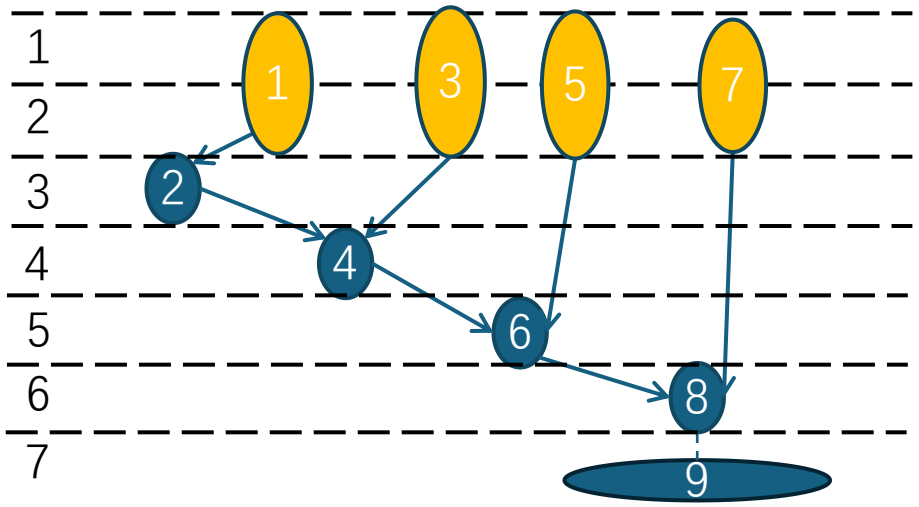
t3=1*x31+2*x32+3*x33+4*x34
+5*x35+6*x36+7*x37

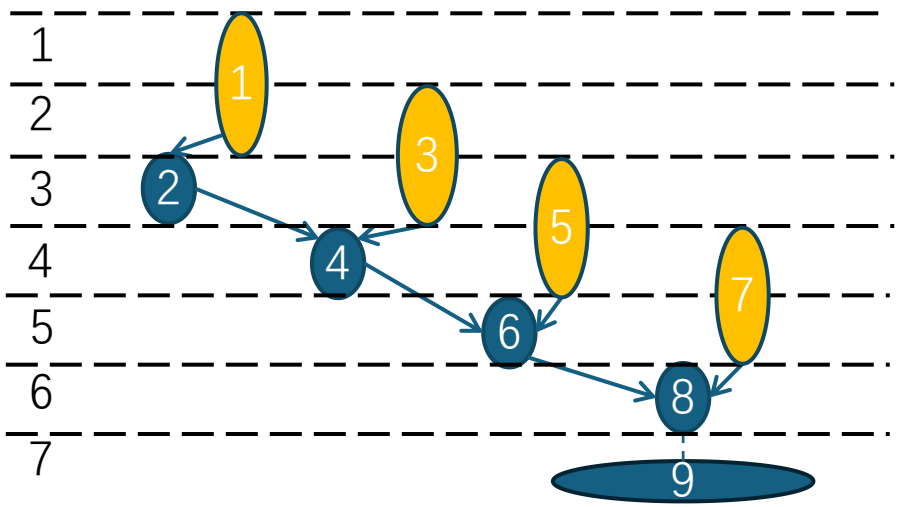
...

t9=1*x91+2*x92+3*x93+4*x94
+5*x95+6*x96+7*x97

Binary

x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77
x81 x82 x83 x84 x85 x86 x87
x91 x92 x93 x94 x95 x96 x97



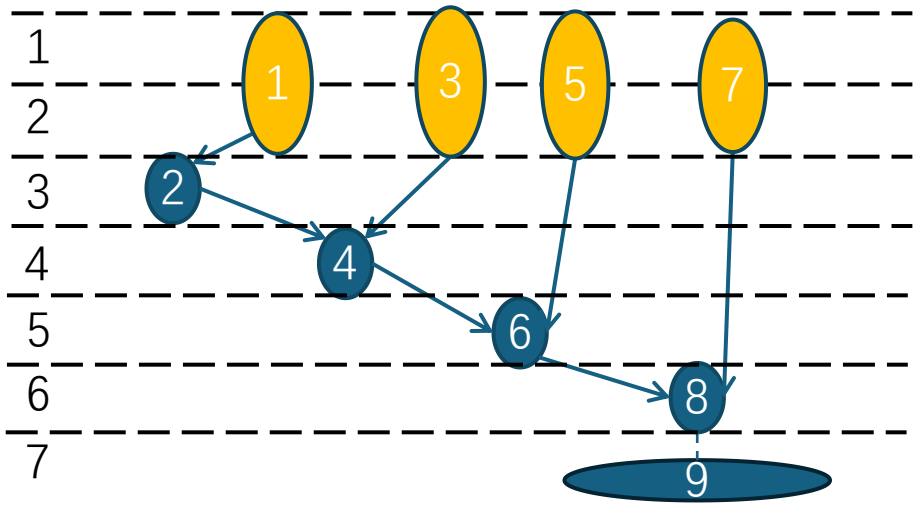


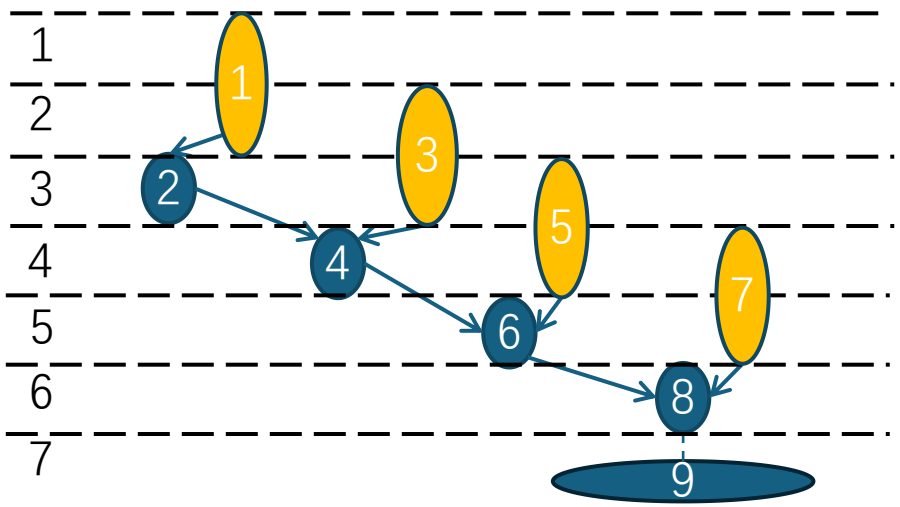
- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

Subject to

t1 =1
t2 =3
t3 =1*x31+2*x32
t4 =4
t5 =1*x51+2*x52+3*x53
t6 =5
t7 =1*x71+2*x72+3*x73+4*x74
t8 =6
t9 =7

t2-t1-1>=0
t4-t2-1>=0
t4-t3-1>=0
t6-t4-1>=0
t6-t5-1>=0
t8-t6-1>=0
t8-t7-1>=0
t9-t8-1>=0





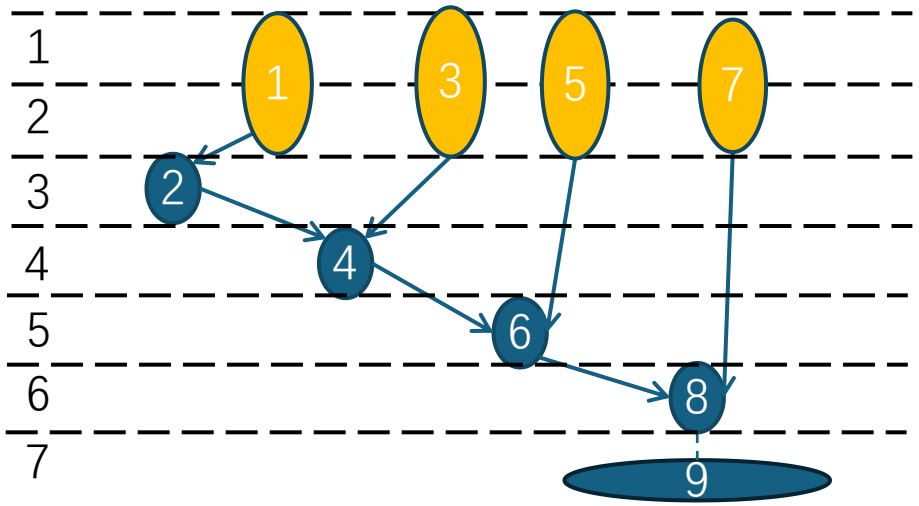
- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

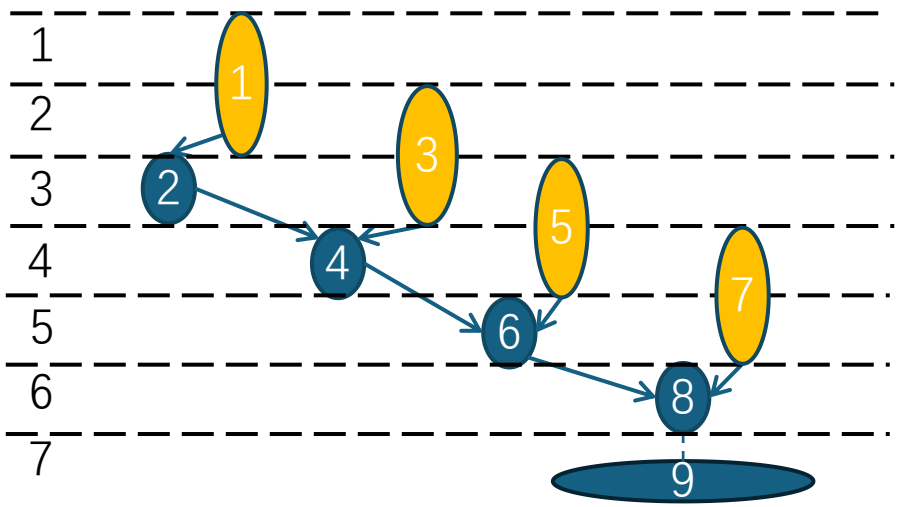
Subject to

t1 =1
t2 =3
t3 =1*x31+2*x32
t4 =4
t5 =1*x51+2*x52+3*x53
t6 =5
t7 =1*x71+2*x72+3*x73+4*x74
t8 =6
t9 =7
4-t3-1>=0
5-t5-1>=0
6-t7-1>=0

Binary

x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77
x81 x82 x83 x84 x85 x86 x87
x91 x92 x93 x94 x95 x96 x97





- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

Subject to

t3 =1*x31+2*x32

t5 =1*x51+2*x52+3*x53

t7 =1*x71+2*x72+3*x73+4*x74

4-t3-1>=0

5-t5-1>=0

6-t7-1>=0

Binary

x11 x12 x13 x14 x15 x16 x17

x21 x22 x23 x24 x25 x26 x27

x31 x32 x33 x34 x35 x36 x37

x41 x42 x43 x44 x45 x46 x47

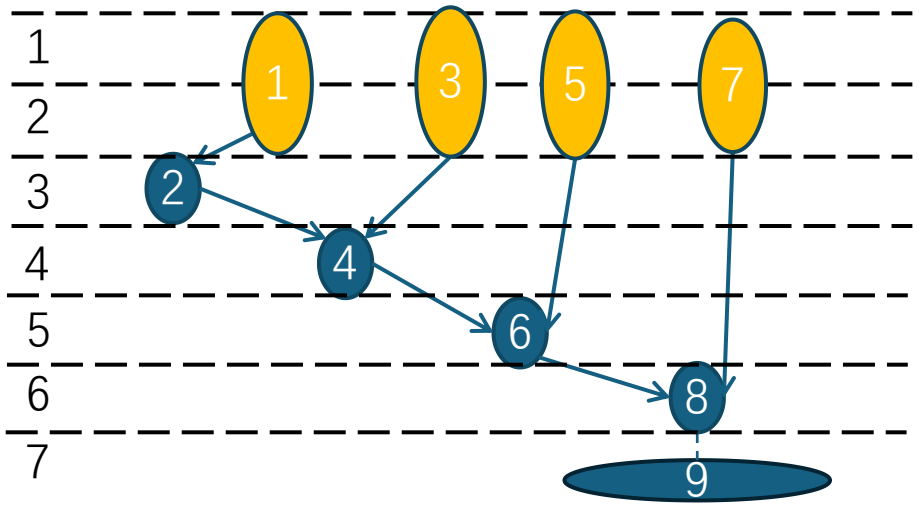
x51 x52 x53 x54 x55 x56 x57

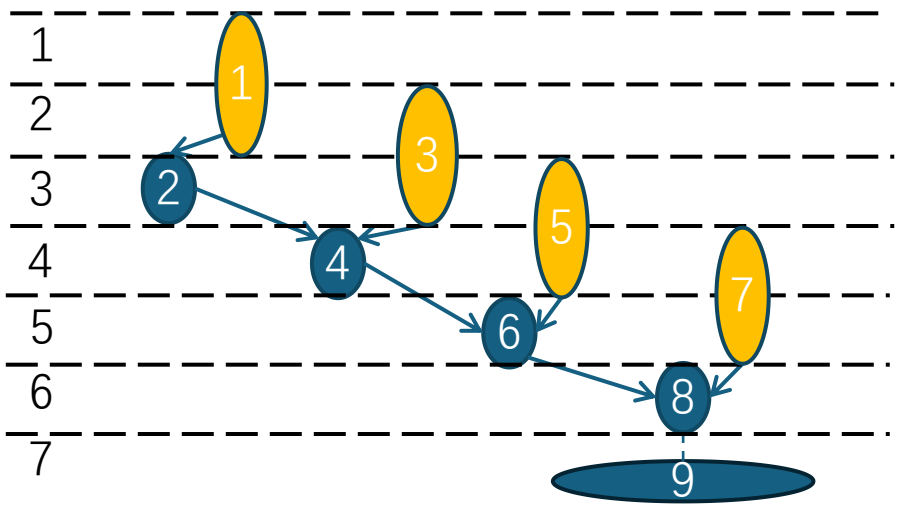
x61 x62 x63 x64 x65 x66 x67

x71 x72 x73 x74 x75 x76 x77

x81 x82 x83 x84 x85 x86 x87

x91 x92 x93 x94 x95 x96 x97





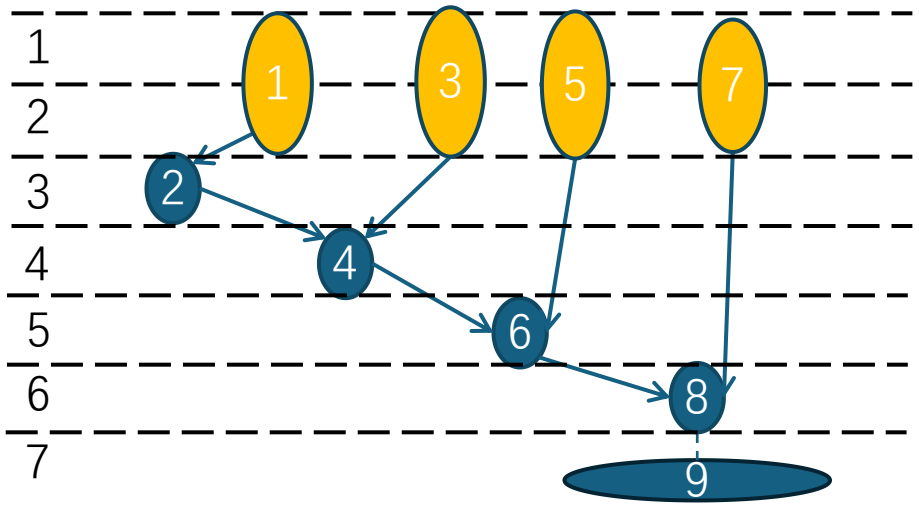
- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

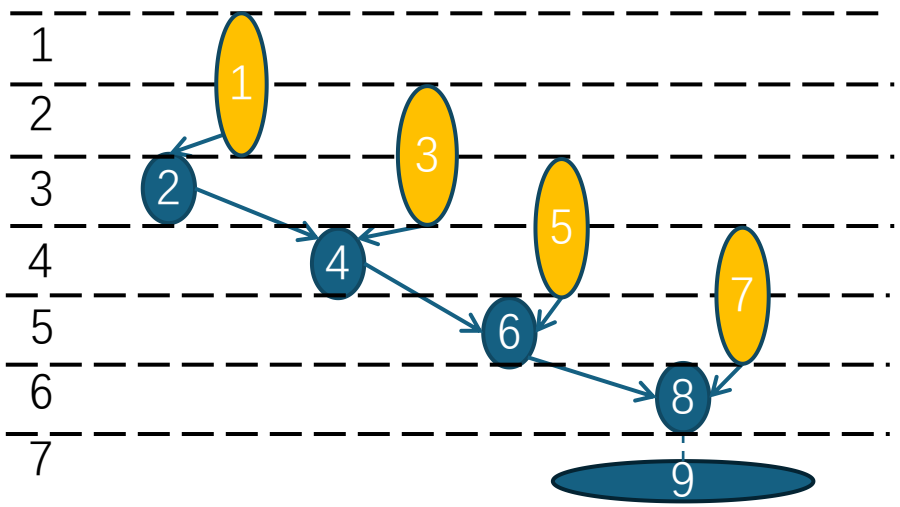
Subject to

1*x91+2*x92+3*x93+4*x94+5*x95+6*x96+7*x97=7

Binary

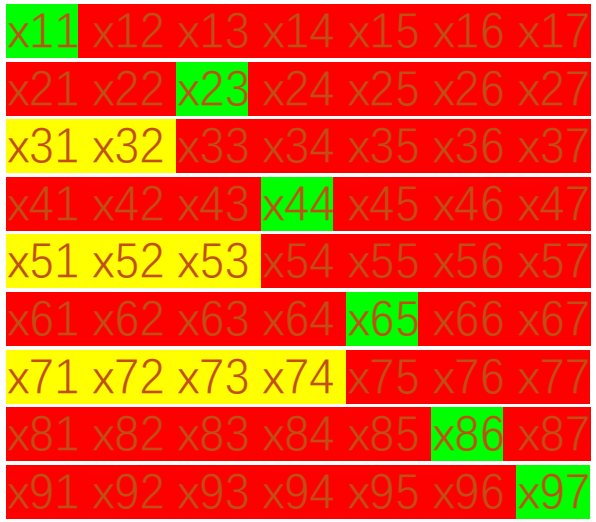
x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77
x81 x82 x83 x84 x85 x86 x87
x91 x92 x93 x94 x95 x96 x97





- 用二进制变量集合来表示最终周期调度情况
- 条件：
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标：
  - 最小化结束时间

Binary



Subject to

(0+x11)+(0+x31)+(0+x51)+(0+x71)<=2

(x11+x12)+(x31+x32)+(x51+x52)+(x71+x72)<=2

(x12+x13)+(x32+x33)+(x52+x53)+(x72+x73)<=2

(x13+x14)+(x33+x34)+(x53+x54)+(x73+x74)<=2

(x14+x15)+(x34+x35)+(x54+x55)+(x74+x75)<=2

(x15+x16)+(x35+x36)+(x55+x56)+(x75+x76)<=2

(x16+x17)+(x36+x37)+(x56+x57)+(x76+x77)<=2

x21+x41+x61+x81+x91<=1

x22+x42+x62+x82+x92<=1

x23+x43+x63+x83+x93<=1

x24+x44+x64+x84+x94<=1

x25+x45+x65+x85+x95<=1

x26+x46+x66+x86+x96<=1

x27+x47+x67+x87+x97<=1

- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

Subject to

1+x31+x51+x71<= 2

1+(x31+x32)+(x51+x52)+(x71+x72)<= 2

(x32)+(x52+x53)+(x72+x73)<= 2

(x53)+(x73+x74)<= 2

Binary

x11 x12 x13 x14 x15 x16 x17

x21 x22 x23 x24 x25 x26 x27

x31 x32 x33 x34 x35 x36 x37

x41 x42 x43 x44 x45 x46 x47

x51 x52 x53 x54 x55 x56 x57

x61 x62 x63 x64 x65 x66 x67

x71 x72 x73 x74 x75 x76 x77

x81 x82 x83 x84 x85 x86 x87

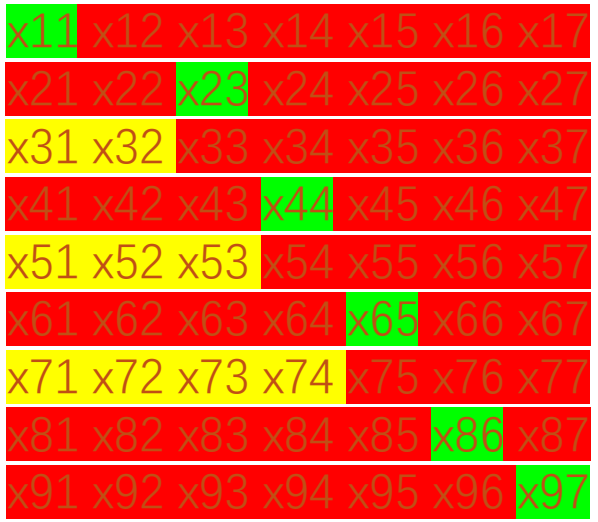
x91 x92 x93 x94 x95 x96 x97

- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

Min

1*x91+2*x92+3*x93+4*x94+5*x95+6*x96+7*x97

Binary



- 用二进制变量集合来表示最终周期调度情况
- 条件:
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标:
  - 最小化结束时间

```
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.02   (Wallclock seconds):       0.02


=== 最优调度方案 ===
t1 = 1.0
t2 = 3.0
t3 = 1.0
t4 = 4.0
t5 = 3.0
t6 = 5.0
t7 = 4.0
t8 = 6.0
t9 = 7.0
a1 = 2.0
a2 = 1.0

最小化目标: 11.0
```

Subject to
x31+x32=1
x51+x52+x53=1
x71+x72+x73+x74=1
t1 =1
t2 =3
t3 =1*x31+2*x32
t4 =4
t5 =1*x51+2*x52+3*x53
t6 =5
t7 =1*x71+2*x72+3*x73+4*x74
t8 =6
t9 =7
4-t3-1>=0
5-t5-1>=0
6-t7-1>=0
1+x31+x51+x71<= 2
1+(x31+x32)+(x51+x52)+(x71+x72)<= 2
(x32)+(x52+x53)+(x72+x73)<= 2
(x53)+(x73+x74)<= 2
Binary
x31 x32 x51 x52 x53 x71 x72 x73 x74
Min
1*x91+2*x92+3*x93+4*x94+5*x95+6*x96+7*x97

- 用二进制变量集合来表示最终周期调度情况
- 条件：
  - 唯一约束
  - 顺序约束
  - 资源约束
- 目标：
  - 最小化结束时间

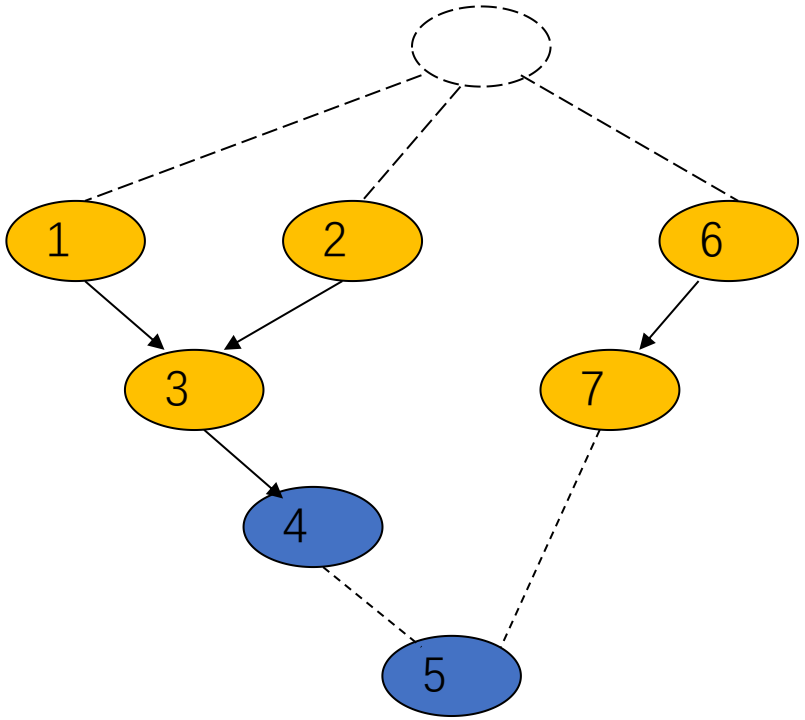
```
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.05   (Wallclock seconds):       0.05


=== 最优调度方案 ===
t1 = 1.0
t2 = 3.0
t3 = 1.0
t4 = 4.0
t5 = 3.0
t6 = 5.0
t7 = 3.0
t8 = 6.0
t9 = 7.0
a1 = 2.0
a2 = 1.0

最小化目标: 11.0
```

```
Binary
x11 x12 x13 x14 x15 x16 x17 x21 x22 x23 x24 x25 x26 x27 x31 x32 x33 x34 x35 x36 x37 x41
x42 x43 x44 x45 x46 x47 x51 x52 x53 x54 x55 x56 x57 x61 x62 x63 x64 x65 x66 x67 x81 x82
x83 x84 x85 x86 x87 x71 x72 x73 x74 x75 x76 x77 x91 x92 x93 x94 x95 x96 x97
Min
5*a1+a2
Subject to
x11 + x12 + x13 + x14 + x15 + x16 + x17 = 1
x21 + x22 + x23 + x24 + x25 + x26 + x27 = 1
x31 + x32 + x33 + x34 + x35 + x36 + x37 = 1
x41 + x42 + x43 + x44 + x45 + x46 + x47 = 1
x51 + x52 + x53 + x54 + x55 + x56 + x57 = 1
x61 + x62 + x63 + x64 + x65 + x66 + x67 = 1
x81 + x82 + x83 + x84 + x85 + x86 + x87 = 1
x71 + x72 + x73 + x74 + x75 + x76 + x77 = 1
x91 + x92 + x93 + x94 + x95 + x96 + x97 = 1
t1 = 1*x11 + 2*x12 + 3*x13 + 4*x14 + 5*x15 + 6*x16 + 7*x17
t2 = 1*x21 + 2*x22 + 3*x23 + 4*x24 + 5*x25 + 6*x26 + 7*x27
t3 = 1*x31 + 2*x32 + 3*x33 + 4*x34 + 5*x35 + 6*x36 + 7*x37
t4 = 1*x41 + 2*x42 + 3*x43 + 4*x44 + 5*x45 + 6*x46 + 7*x47
t5 = 1*x51 + 2*x52 + 3*x53 + 4*x54 + 5*x55 + 6*x56 + 7*x57
t6 = 1*x61 + 2*x62 + 3*x63 + 4*x64 + 5*x65 + 6*x66 + 7*x67
t8 = 1*x81 + 2*x82 + 3*x83 + 4*x84 + 5*x85 + 6*x86 + 7*x87
t7 = 1*x71 + 2*x72 + 3*x73 + 4*x74 + 5*x75 + 6*x76 + 7*x77
t9 = 1*x91 + 2*x92 + 3*x93 + 4*x94 + 5*x95 + 6*x96 + 7*x97
t2 - t1 - 2 >= 0
t4 - t3 - 2 >= 0
t4 - t2 - 1 >= 0
t6 - t5 - 2 >= 0
t6 - t4 - 1 >= 0
t8 - t7 - 2 >= 0
t8 - t6 - 1 >= 0
t9 - t8 - 1 >= 0
t9=7
x11 + x31 + x51 + x71 <= a1
(x11+x12)+(x31+x32)+(x51+x52)+(x71+x72)<= a1
(x12+x13)+(x32+x33)+(x52+x53)+(x72+x73)<= a1
(x13+x14)+(x33+x34)+(x53+x54)+(x73+x74)<= a1
(x14+x15)+(x34+x35)+(x54+x55)+(x74+x75)<= a1
(x15+x16)+(x35+x36)+(x55+x56)+(x75+x76)<= a1
(x16+x17)+(x36+x37)+(x56+x57)+(x76+x77)<= a1
x21+x41+x61+x81+x91 <= a2
x22+x42+x62+x82+x92 <= a2
x23+x43+x63+x83+x93 <= a2
x24+x44+x64+x84+x94 <= a2
x25+x45+x65+x85+x95 <= a2
x26+x46+x66+x86+x96 <= a2
x27+x47+x67+x87+x97 <= a2
```

随堂作业

- 用二进制变量集合来表示最终周期调度情况
- 条件：
  - 唯一约束
  - 顺序约束
  - 时间约束
- 目标：
  - 最小化资源面积

λ是6，有两种资源，延迟都为2，分别是：乘法器，所占面积为3；加法器，所占面积为2。



Min

3*a1+2*a2

Subject to

x11+x12+x13+x14+x15+x16+x17=1 …

x71+x72+x73+x74+x75+x76+x77=1

t1=1*x11+2*x12+3*x13+4*x14+5*x15+6*x16+7*x17 …

t7=1*x71+2*x72+3*x73+4*x74+5*x75+6*x76+7*x77

t3-t1-2>=0

t3-t2-2>=0

t4-t3-2>=0

t5-t4-2>=0

t7-t6-2>=0

t5-t7-2>=0

t5<=7

(0+x11)+(0+x21)+(0+x31)+(0+x61)+(0+x71)<=a1 …

(x16+x17)+(x26+x27)+(x36+x37)+(x66+x67)+(x76+x77)<=a1

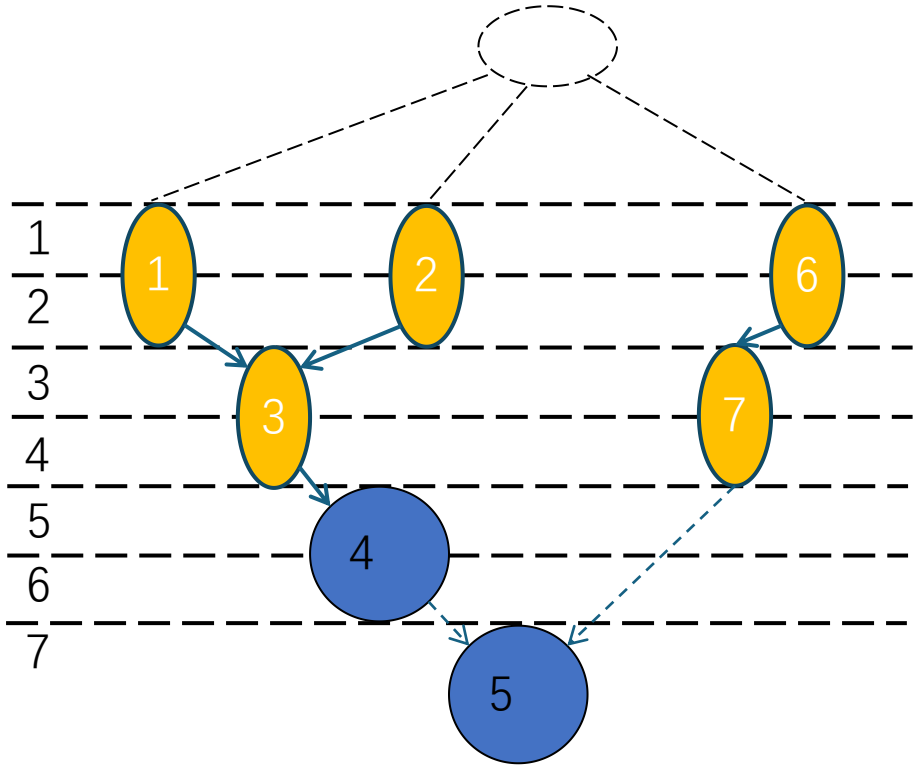
(0+x41)+(0+x51)<=a2…

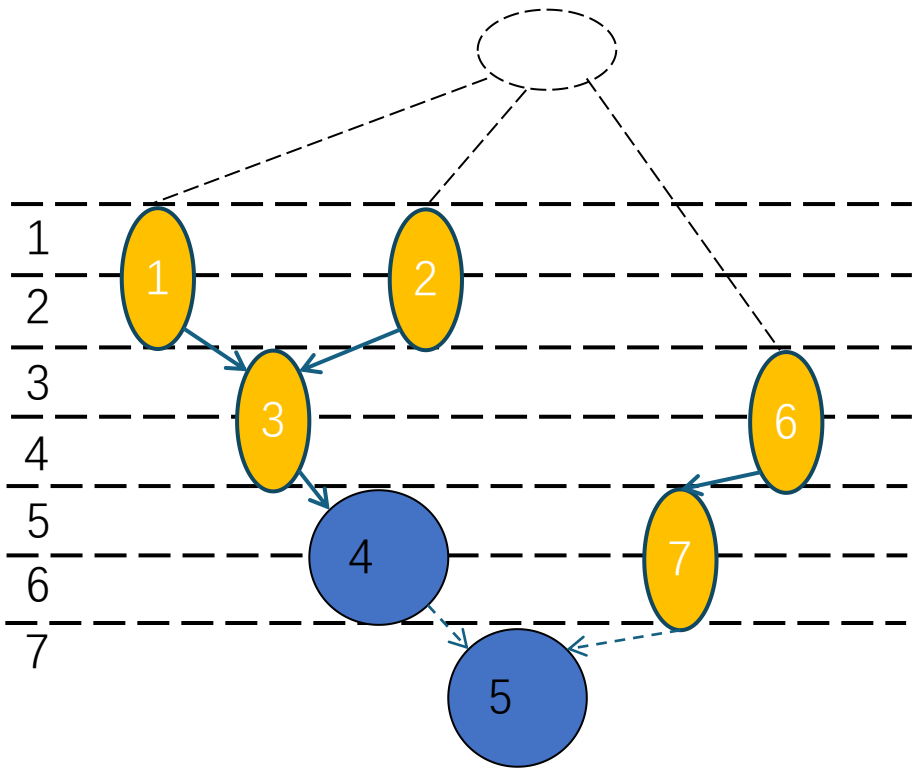
(x46+x47)+(x56+x57)<=a2

Binary

x10 x11 x12 x13 x14 x15 x16 x17…

x70 x71 x72 x73 x74 x75 x76 x77

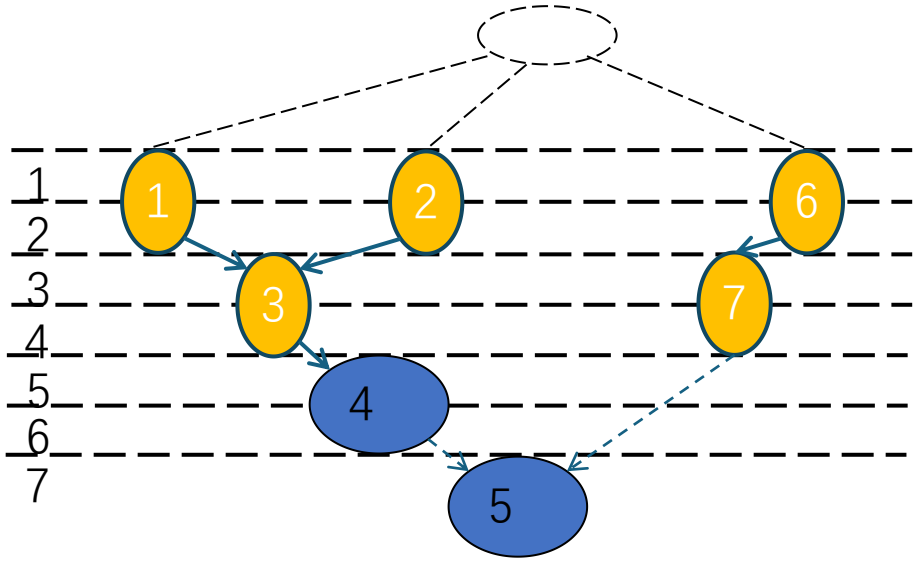


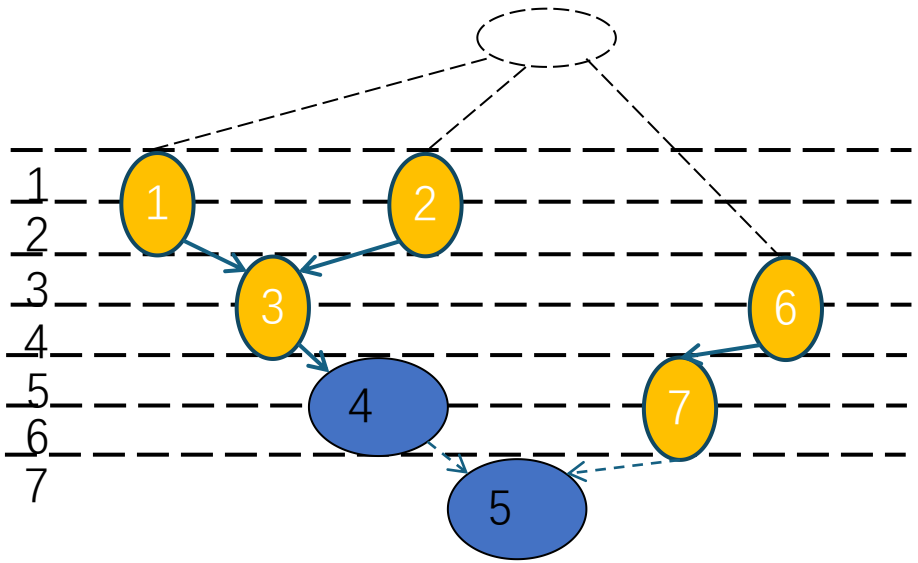


1的开工周期: 1-1
2的开工周期: 1-1
3的开工周期: 3-3
4的开工周期: 5-5
5的开工周期: 7-7
6的开工周期: 1-3
7的开工周期: 3-5

Binary

x11 x12 x13 x14 x15 x16 x17
x21 x22 x23 x24 x25 x26 x27
x31 x32 x33 x34 x35 x36 x37
x41 x42 x43 x44 x45 x46 x47
x51 x52 x53 x54 x55 x56 x57
x61 x62 x63 x64 x65 x66 x67
x71 x72 x73 x74 x75 x76 x77





1的开工周期: 1-1
2的开工周期: 1-1
3的开工周期: 3-3
4的开工周期: 5-5
5的开工周期: 7-7
6的开工周期: 1-3
7的开工周期: 3-5

Subject to

x11+x12+x13+x14+x15+x16+x17=1

x21+x22+x23+x24+x25+x26+x27=1

x31+x32+x33+x34+x35+x36+x37=1

x41+x42+x43+x44+x45+x46+x47=1

x51+x52+x53+x54+x55+x56+x57=1

x61+x62+x63+x64+x65+x66+x67=1

x71+x72+x73+x74+x75+x76+x77=1

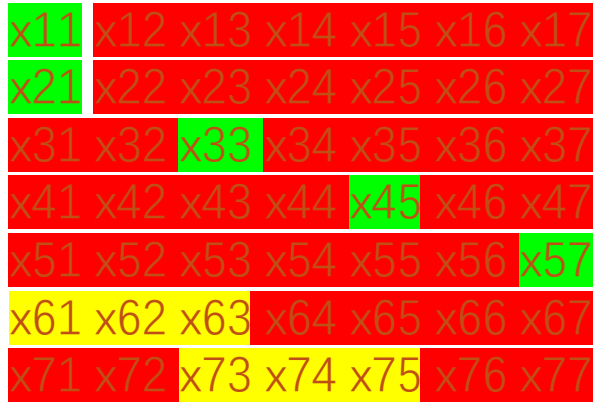
Binary

Subject to

x61+x62+x63=1

x73+x74+x75=1

Binary



Subject to

t1=1*x11+2*x12+3*x13+4*x14+5*x15+6*x16+7*x17

t2=1*x21+2*x22+3*x23+4*x24+5*x25+6*x26+7*x27

t3=1*x31+2*x32+3*x33+4*x34+5*x35+6*x36+7*x37

t4=1*x41+2*x42+3*x43+4*x44+5*x45+6*x46+7*x47

t5=1*x51+2*x52+3*x53+4*x54+5*x55+6*x56+7*x57

t6=1*x61+2*x62+3*x63+4*x64+5*x65+6*x66+7*x67

t7=1*x71+2*x72+3*x73+4*x74+5*x75+6*x76+7*x77

Binary

x11 x12 x13 x14 x15 x16 x17

x21 x22 x23 x24 x25 x26 x27

x31 x32 x33 x34 x35 x36 x37

x41 x42 x43 x44 x45 x46 x47

x51 x52 x53 x54 x55 x56 x57

x61 x62 x63 x64 x65 x66 x67

x71 x72 x73 x74 x75 x76 x77

Subject to

t1=1

t2=1

t3=3

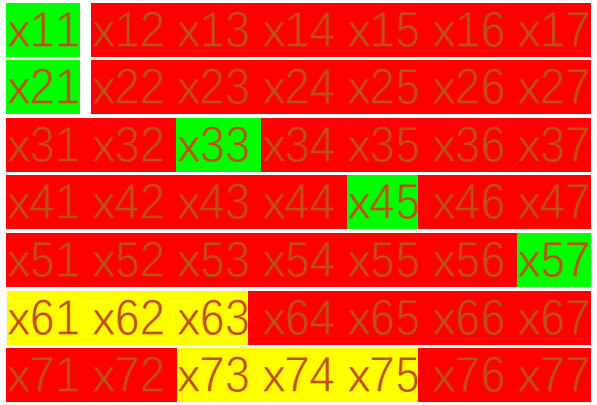
t4=5

t5=7

t6=1*x61+2*x62+3*x63

t7=3*x73+4*x74+5*x75

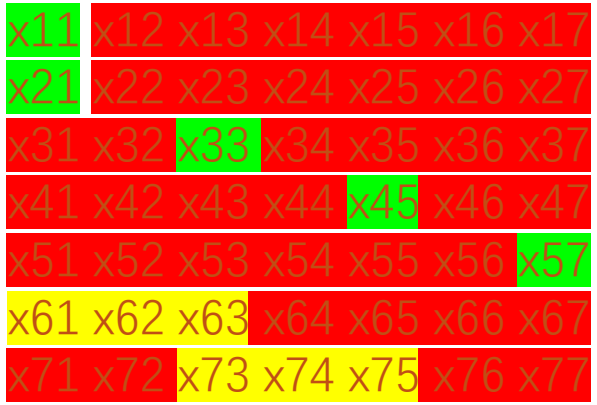
Binary



Subject to

t5<=7

Binary



Subject to

(0+x11)+(0+x21)+(0+x31)+(0+x61)+(0+x71)<=a1

(x11+x12)+(x21+x22)+(x31+x32)+(x61+x62)+(x71+x72)<=a1

(x12+x13)+(x22+x23)+(x32+x33)+(x62+x63)+(x72+x73)<=a1

(x13+x14)+(x23+x24)+(x33+x34)+(x63+x64)+(x73+x74)<=a1

(x14+x15)+(x24+x25)+(x34+x35)+(x64+x65)+(x74+x75)<=a1

(x15+x16)+(x25+x26)+(x35+x36)+(x65+x66)+(x75+x76)<=a1

(x16+x17)+(x26+x27)+(x36+x37)+(x66+x67)+(x76+x77)<=a1

(0+x41)+(0+x51)<=a2…

(x41+x42)+(x51+x52)<=a2

(x42+x43)+(x52+x53)<=a2

(x43+x44)+(x53+x54)<=a2

(x44+x45)+(x54+x55)<=a2

(x45+x46)+(x55+x56)<=a2

(x46+x47)+(x56+x57)<=a2

Min

3*a1+2*a2

Binary

Subject to

2+x61<=a1

2+(x61+x62)<=a1

1+(x62+x63)+x73<=a1

1+x63+(x73+x74)<=a1

(x74+x75)<=a1

x75<=a1

1<=a2

Min

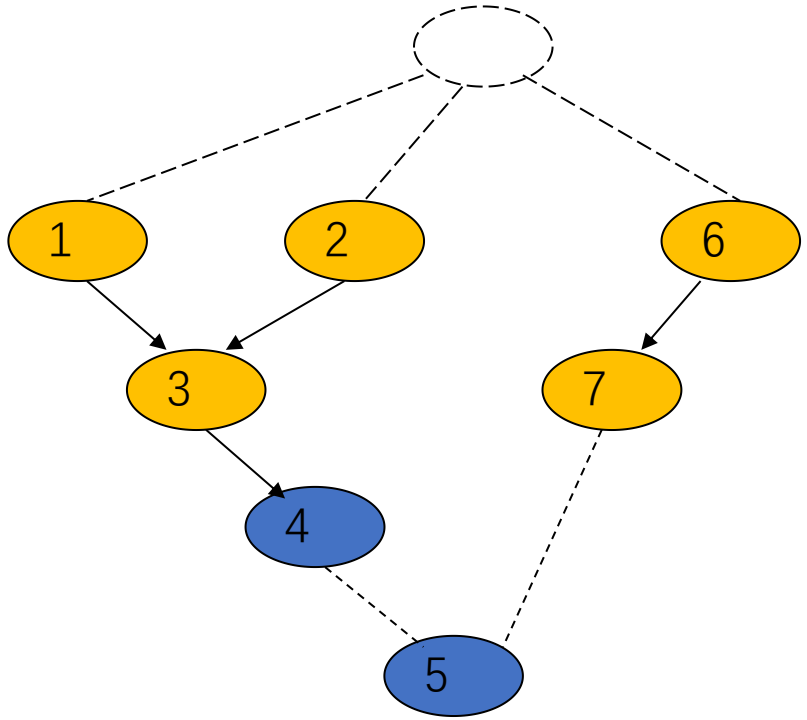
3*a1+2*a2

Binary

随堂作业

- 用二进制变量集合来表示最终周期调度情况
- 条件：
  - 唯一约束
  - 顺序约束
  - 时间约束
- 目标：
  - 最小化资源面积

λ是6，有两种资源，延迟都为2，分别是：乘法器，所占面积为3；加法器，所占面积为2。



Min
3*a1+2*a2
Subject to
x61+x62+x63=1
x73+x74+x75=1
t1=1
t2=1
t3=3
t4=5
t5=7
t6=1*x61+2*x62+3*x63
t7=3*x73+4*x74+5*x75
2+x61<=a1
2+(x61+x62)<=a1
1+(x62+x63)+x73<=a1
1+x63+(x73+x74)<=a1
(x74+x75)<=a1
x75<=a1
1<=a2
Min
3*a1+2*a2
Binary
x61 x62 x63 x73 x74 x75