EDA软件设计-调度算法2

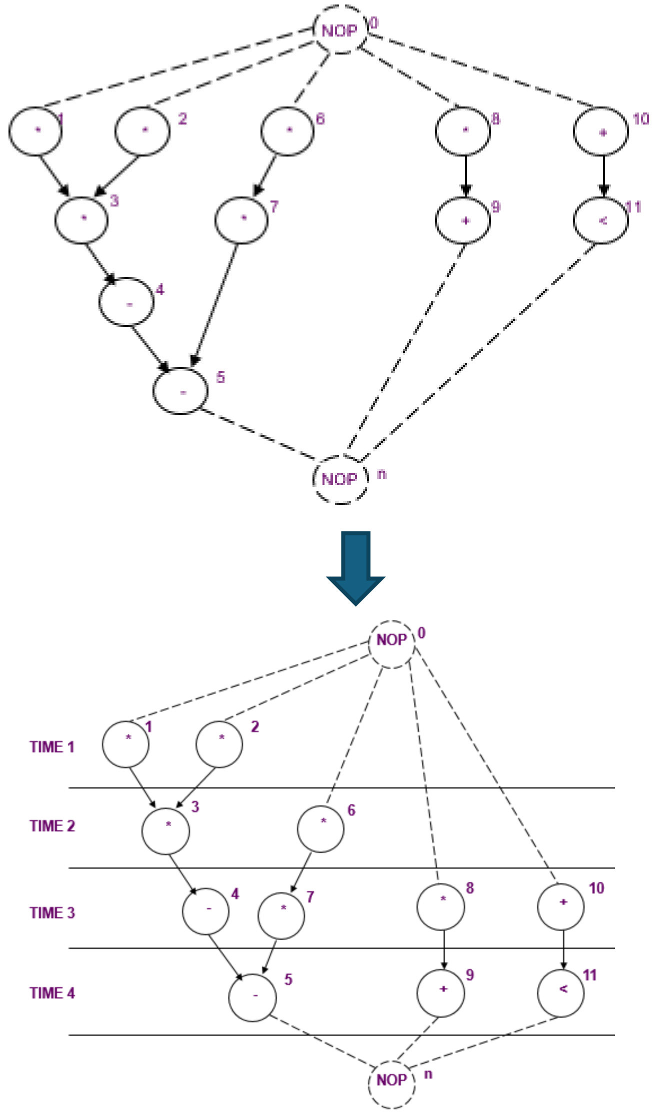
2024-2025年春季学期

在正式开始之前

操作调度

Operation Scheduling

- 输入：
  - 顺序图 G(V, E)，含 n 个顶点
  - 操作延迟 $D = \{d_i: i=0..n\}$
- 输出：
  - 调度 φ ： 确定操作 vi 的开始时间 t_i
  - 延迟 $\lambda = t_n - t_0$
- 目标：权衡资源和时间



最小延迟无约束调度

Minimum-latency Unconstrained Scheduling

- 最简单的情况：没有约束，找到最小的延迟
- 输入：顶点集 V，延迟 D 和偏序关系集E，
- 输出：对每个操作进行标记φ: V → Z^+，使得：
  - $t_i = \varphi(v_i)$
  - $t_i \geqslant t_j + d_j$ 对于所有 $(v_i, v_j) \in E$
  - $\lambda = t_n - t_0$ 是最小的
- 可在多项式时间内解决
- 使用的 ASAP 算法：拓扑顺序

ASAP调度算法

ASAP scheduling algorithm

ASAP ($G_s(V,E)$) {

    Schedule v_0 by setting $t_0 = 0$;

    repeat {

        Select a vertex v_i whose predecessors are all scheduled;

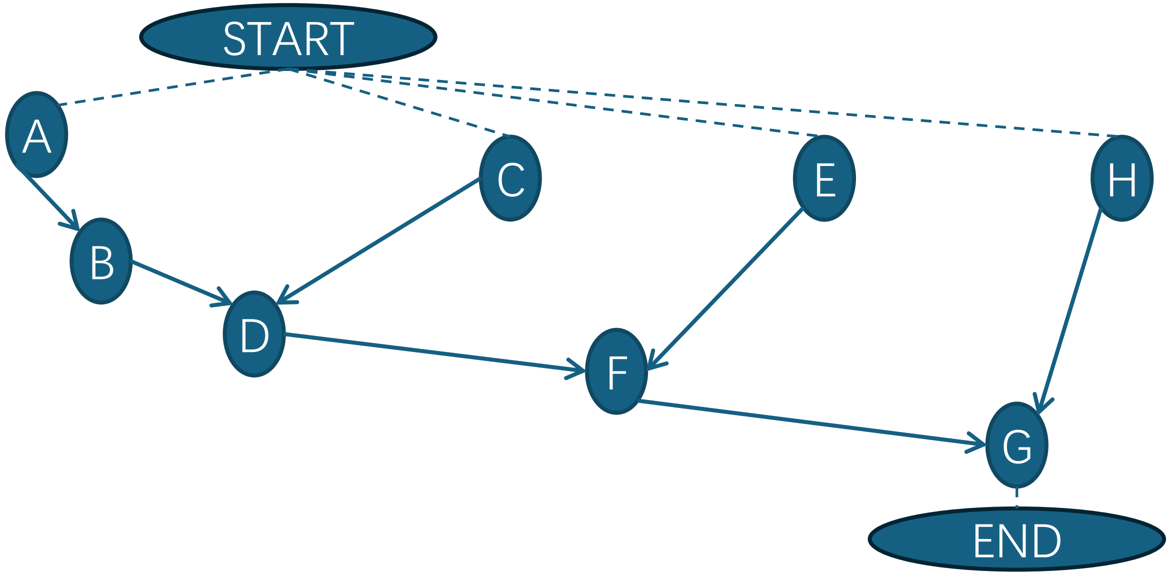
        Schedule v_i by setting $t_i = \max \ t_j + d_j$;

    }

    until (v_n is scheduled);

    return (G);

}



ALAP调度算法（最晚时间约束）

ALAP scheduling algorithm（Latency Constrained）

ALAP ($G_s(V,E)$, x) {

    Schedule v_n by setting $t_n = x+1$;

    repeat {

        Select a vertex v_i whose successors are all scheduled;

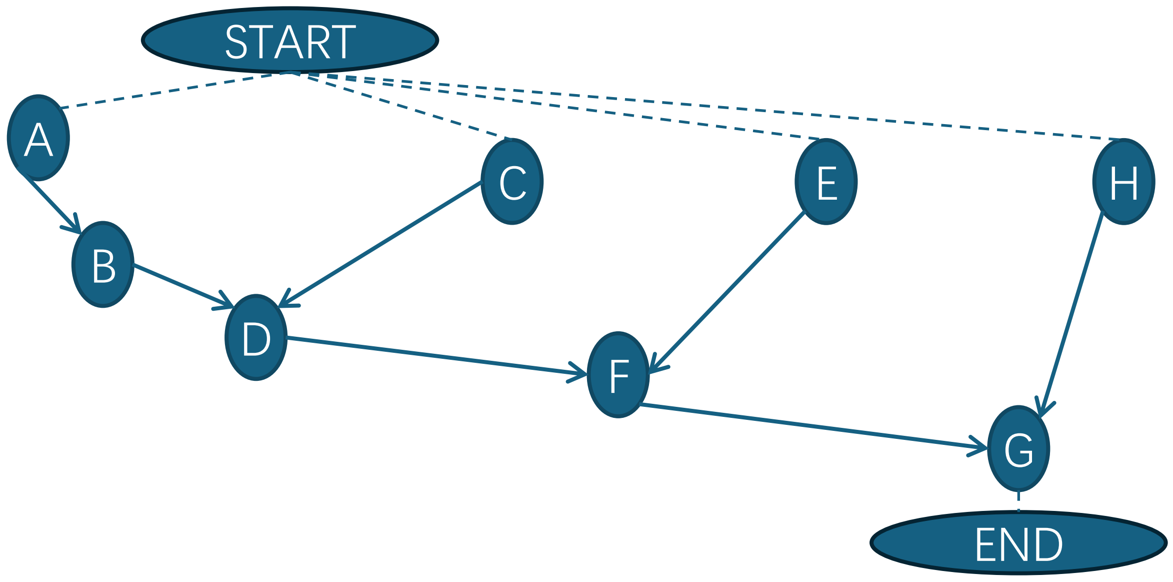
        Schedule v_i by setting $t_i = \min \; t_j - d_i$;

    }

    until (v_0 is scheduled);

    return (t);

}



输入文件内容:

7

START 1

A 2

B 1

C 3

D 2

E 1

END 0

START A

START C

A B

C B

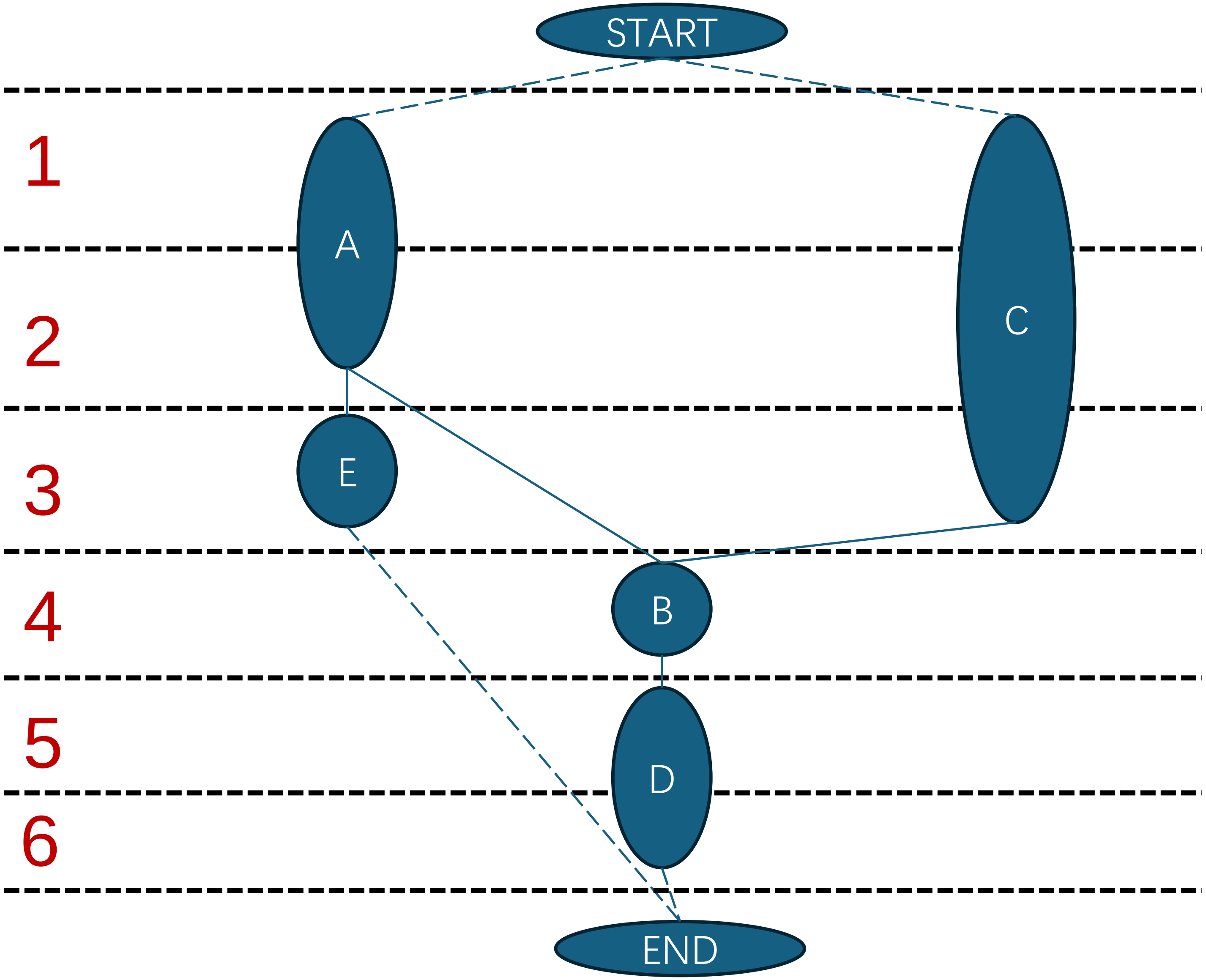
A E

B D

E END

D END

ASAP的结果



为顶点决定开始周期的顺序:

START->

A->C->B->D->E

->END

输出文件内容:

START 0

A 1

C 1

B 4

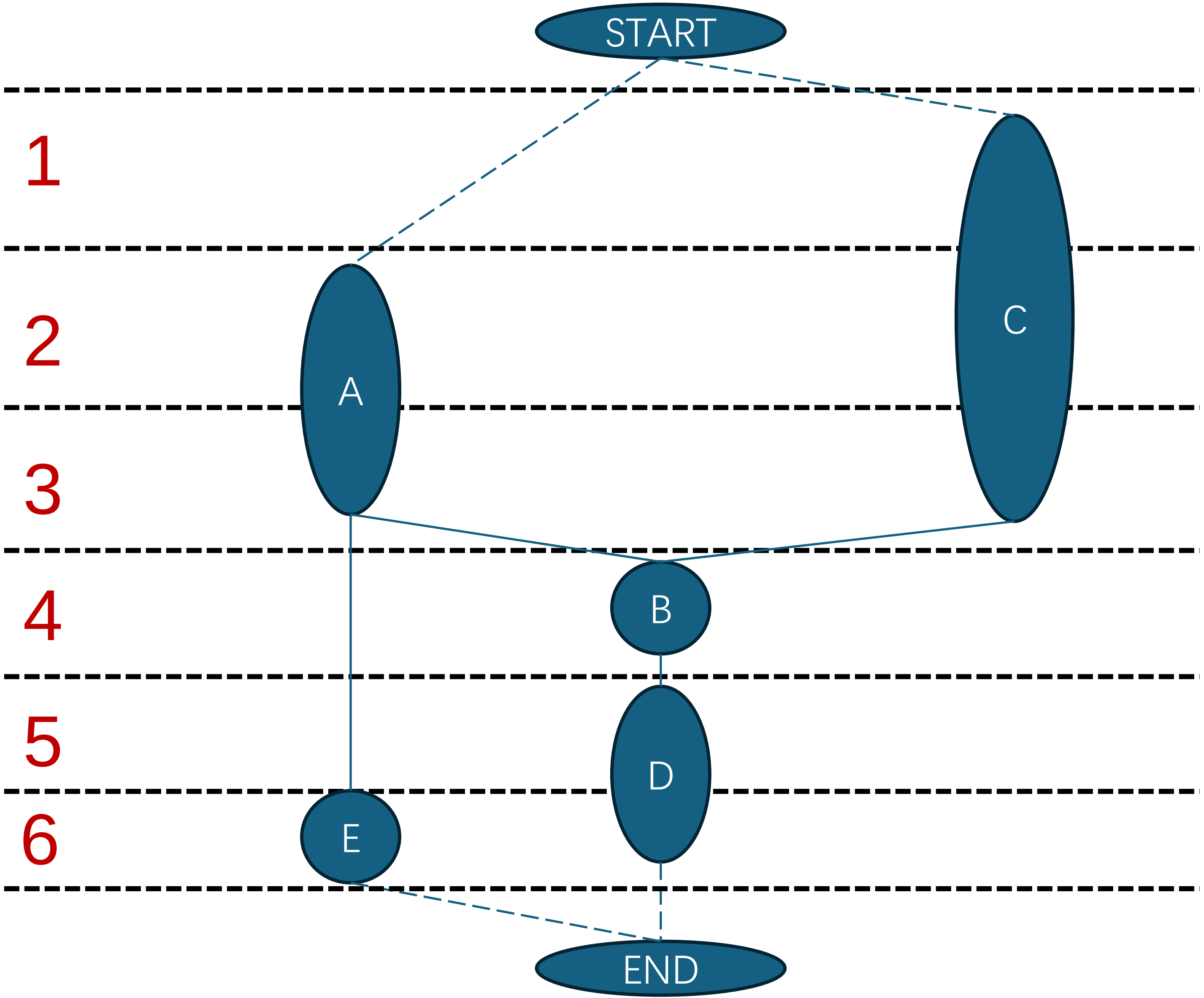
D 5

E 3

END 7

输入文件内容:
7
START 1
A 2
B 1
C 3
D 2
E 1
END 0
START A
START C
A B
C B
A E
B D
E END
D END

ALAP的结果



为顶点决定开始周期的顺序:
END->
D -> B -> C -> E -> A
->START

输出文件内容:
START 0
D 5
B 4
C 1
E 6
A 2
END 7

有约束的调度

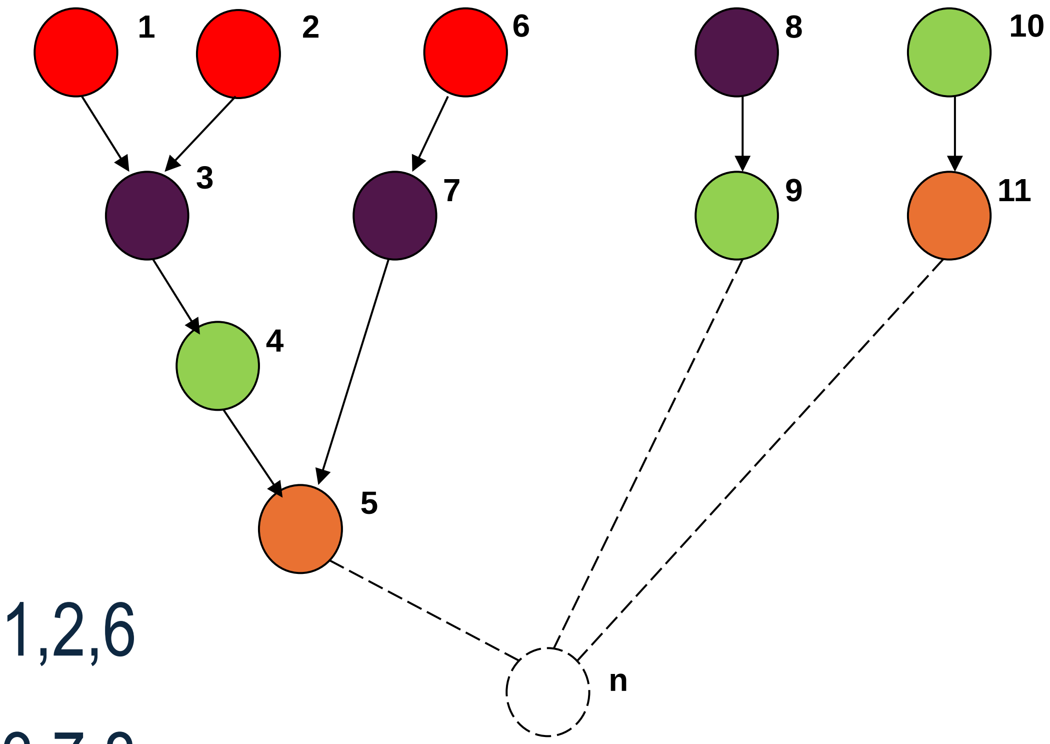
Constrained Scheduling

- 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）

优先级约束的多处理器调度

Precedence-constrained Multiprocessor Scheduling

$\bar{a} = 3$



Step 1: Op 1,2,6

Step 2: Op 3,7,8

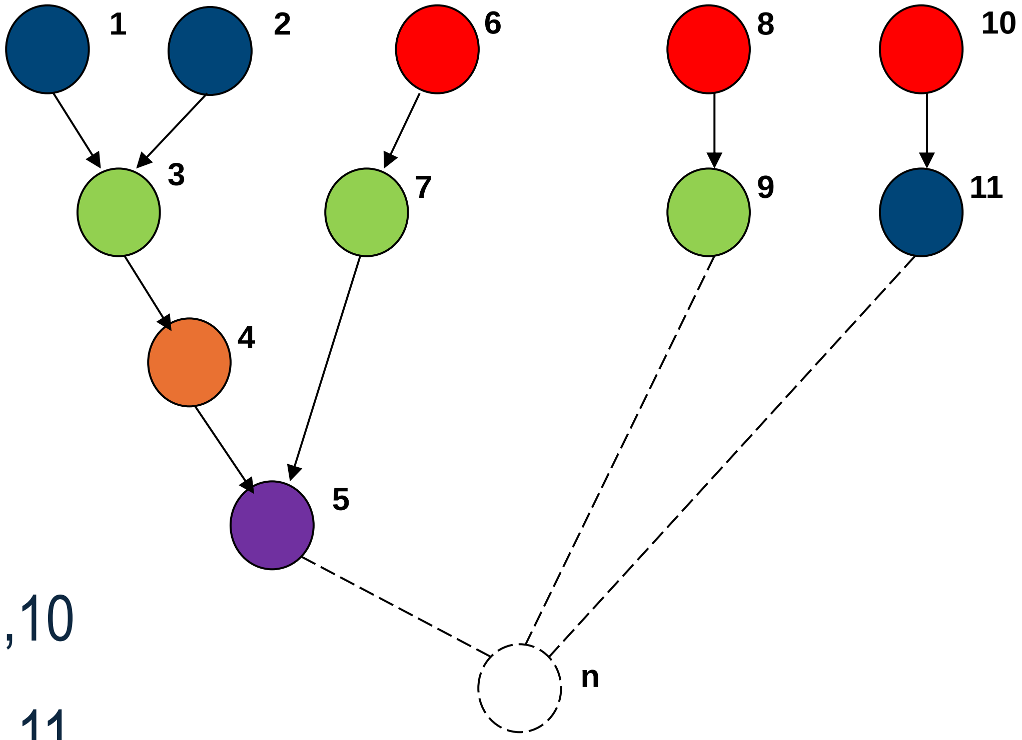
Step 3: Op 4,9,10

Step 4: Op 5,11

优先级约束的多处理器调度

Precedence-constrained Multiprocessor Scheduling

$\bar{a} = 3$



Step 1: Op 6,8,10

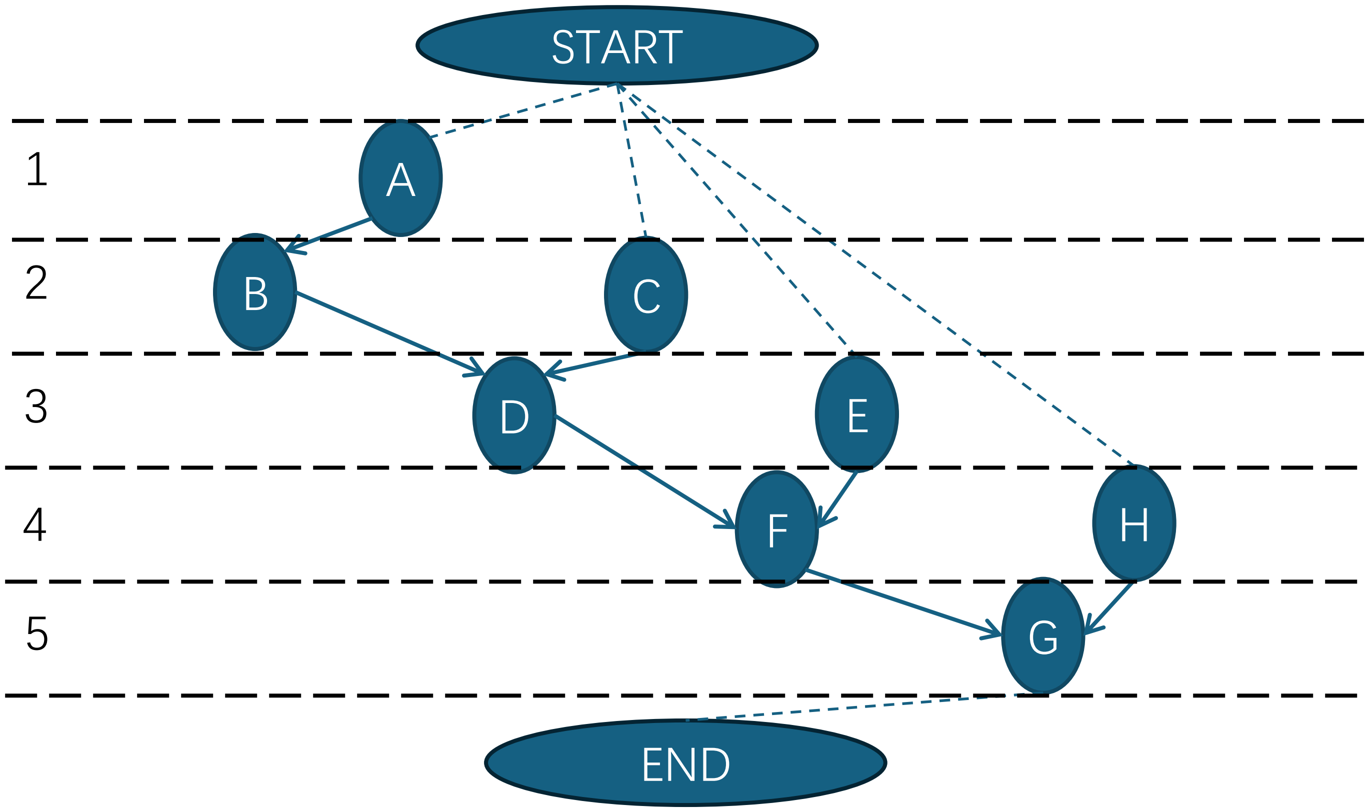
Step 2: Op 1,2,11

Step 3: Op 3,7,9

Step 4: Op 4

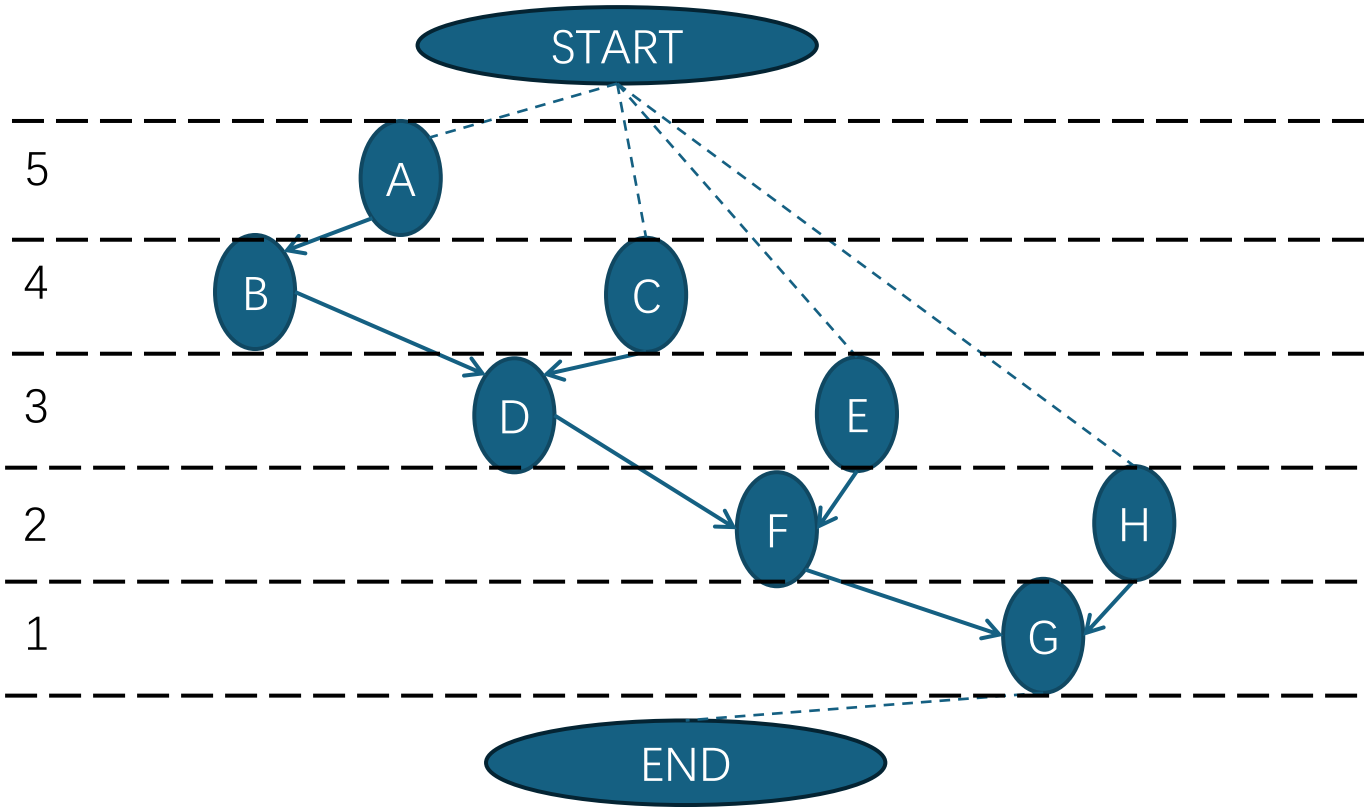
Step 5: Op 5

ALAP调度结果



END 6
G 5
F 4
D 3
B 2
A 1
C 2
E 3
H 4

距离终点的距离



END 6
G 5
F 4
D 3
B 2
A 1
C 2
E 3
H 4

HU调度算法

HU scheduling algorithm

HU (G(V, E), a) {

对顶点进行标号;

$t0 = 0$; $l = 1$;

重复执行以下步骤:

U = 仍未调度的顶点集合，这些顶点要么没有前驱，要么其所有前驱都已被调度;

选择一个子集 $S \subseteq U$，使得 $|S| \le a$，且 S 中的顶点标签值最大;

在步骤 t 处调度 S 中的所有操作，即对所有 $v_i \in S$ 设定调度时间 $t_i = l$;

$l = l + 1$;

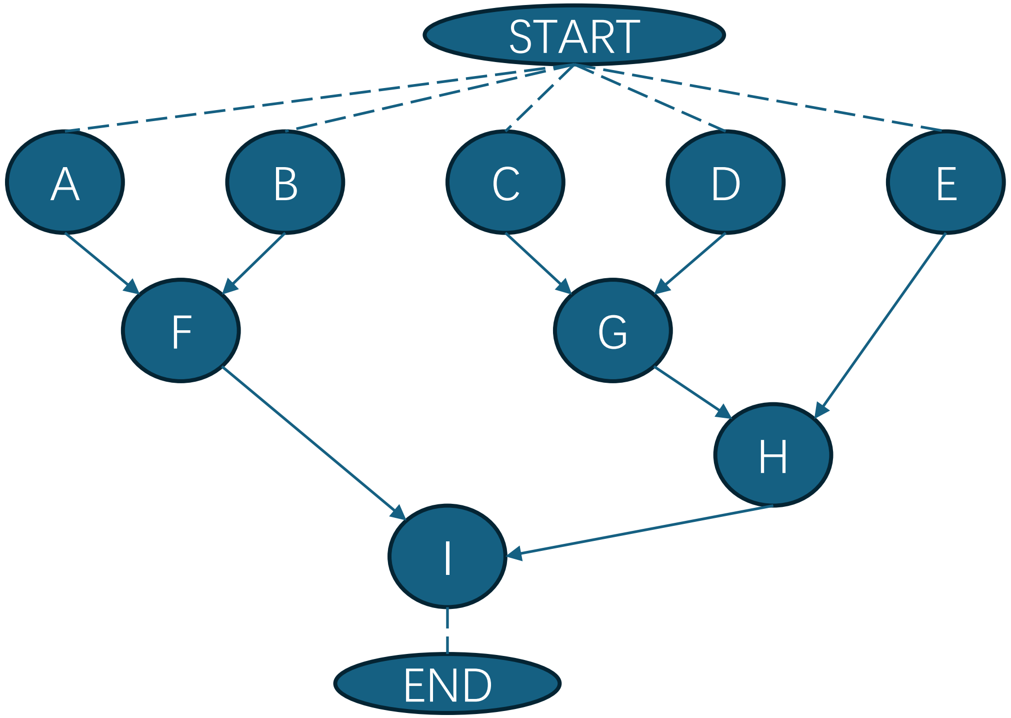
直到 v_n 被调度;

}

作业2

in-class assignment

已知输入的图如下（每个都是加法操作）：



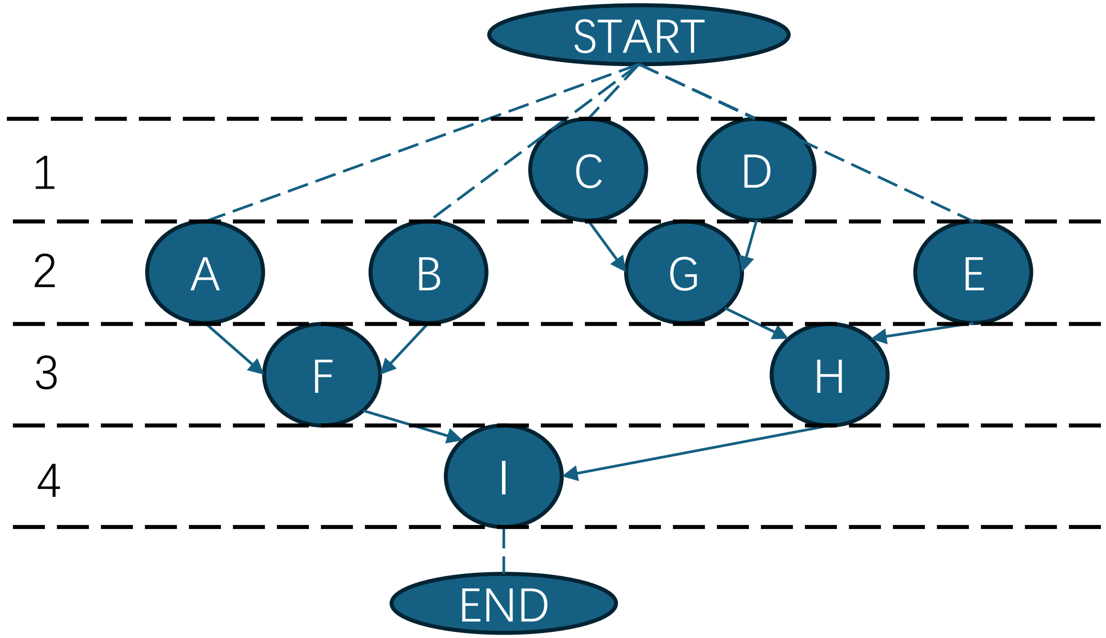
假设我们有两个加法器，请写出HU调度下的程序**的输出文件的内容**，输出文件格式如下:

每一行由一个字符串和一个数字组成，以空格隔开，分别代表顶点名和开始的调度周期

作业2

in-class assignment

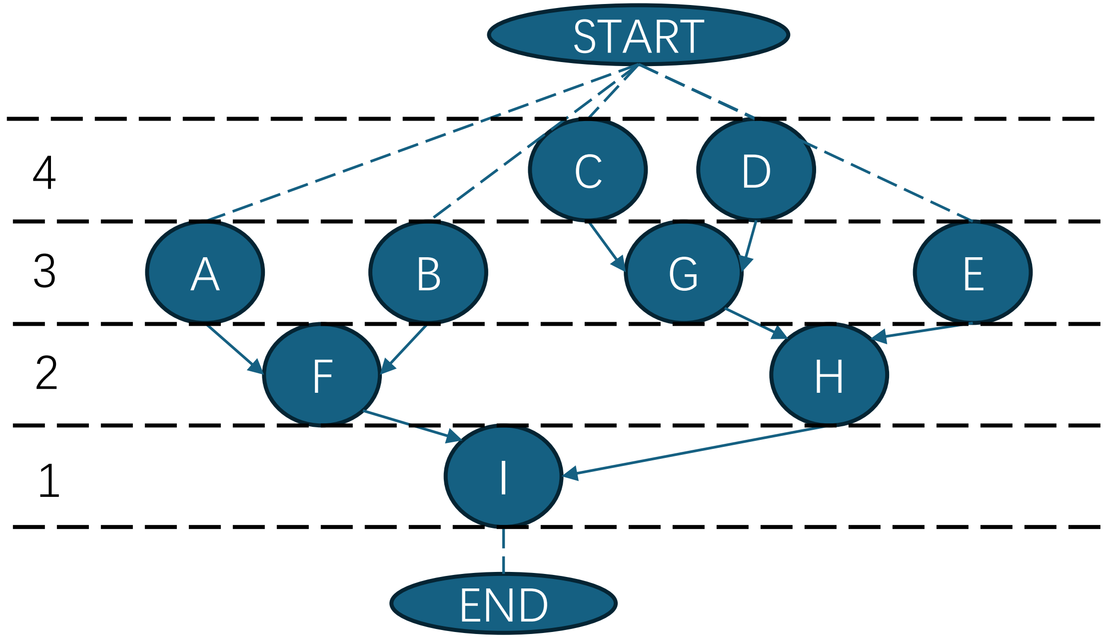
ALAP调度结果

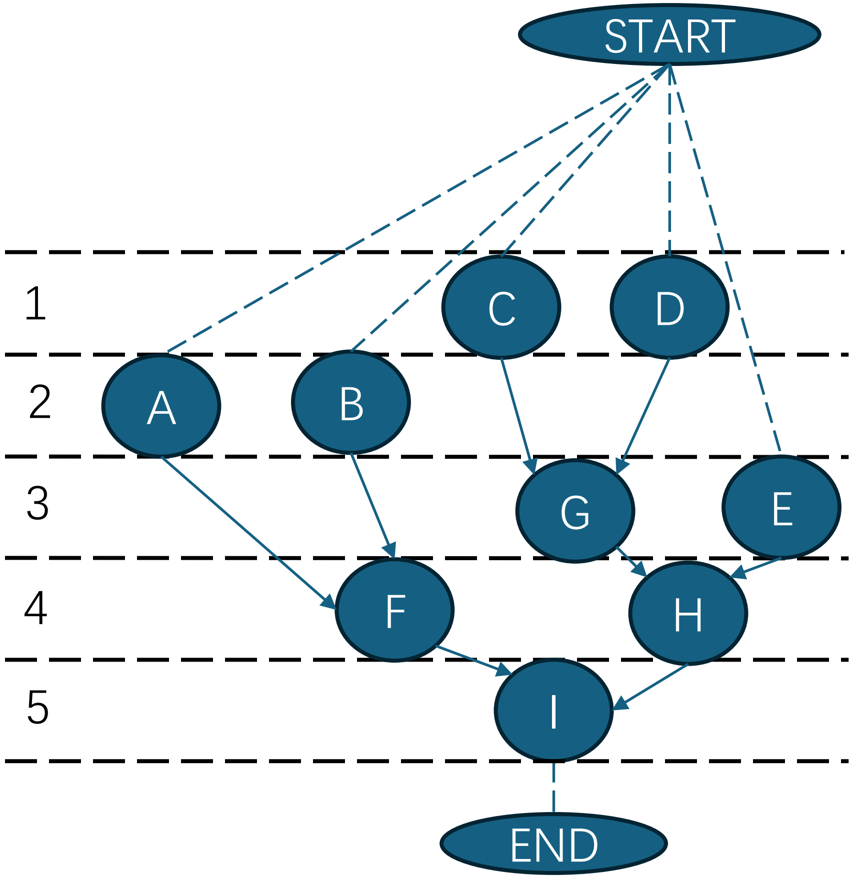


作业2

in-class assignment

最远距离标号





输出文件内容:

START 0

C 1

D 1

A 2

B 2

E 3

G 3

F 4

H 4

I 5

调度算法

Scheduling Algorithms

2025-3

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）

ILP

线性规划（Linear Programming, LP）是指在满足一组线性约束条件的前提下，优化一个线性目标函数的数学优化方法。

- 线性约束是指由决策变量的线性表达式构成的约束条件，通常形式为：

$$a1x1+a2x2+\cdots+anxn\leqslant b$$

- 线性目标函数是指对决策变量进行线性组合，以优化（最大化或最小化）某个目标的数学表达式，通常形式为：

$$Z=c1x1+c2x2+\cdots+cnxn$$

条件:

$X_1, X_2 >=0$

$2X_1+X_2<=1000$

$3X_1+4X_2<=2400$

最大化:

$8X_1+5X_2$

条件:

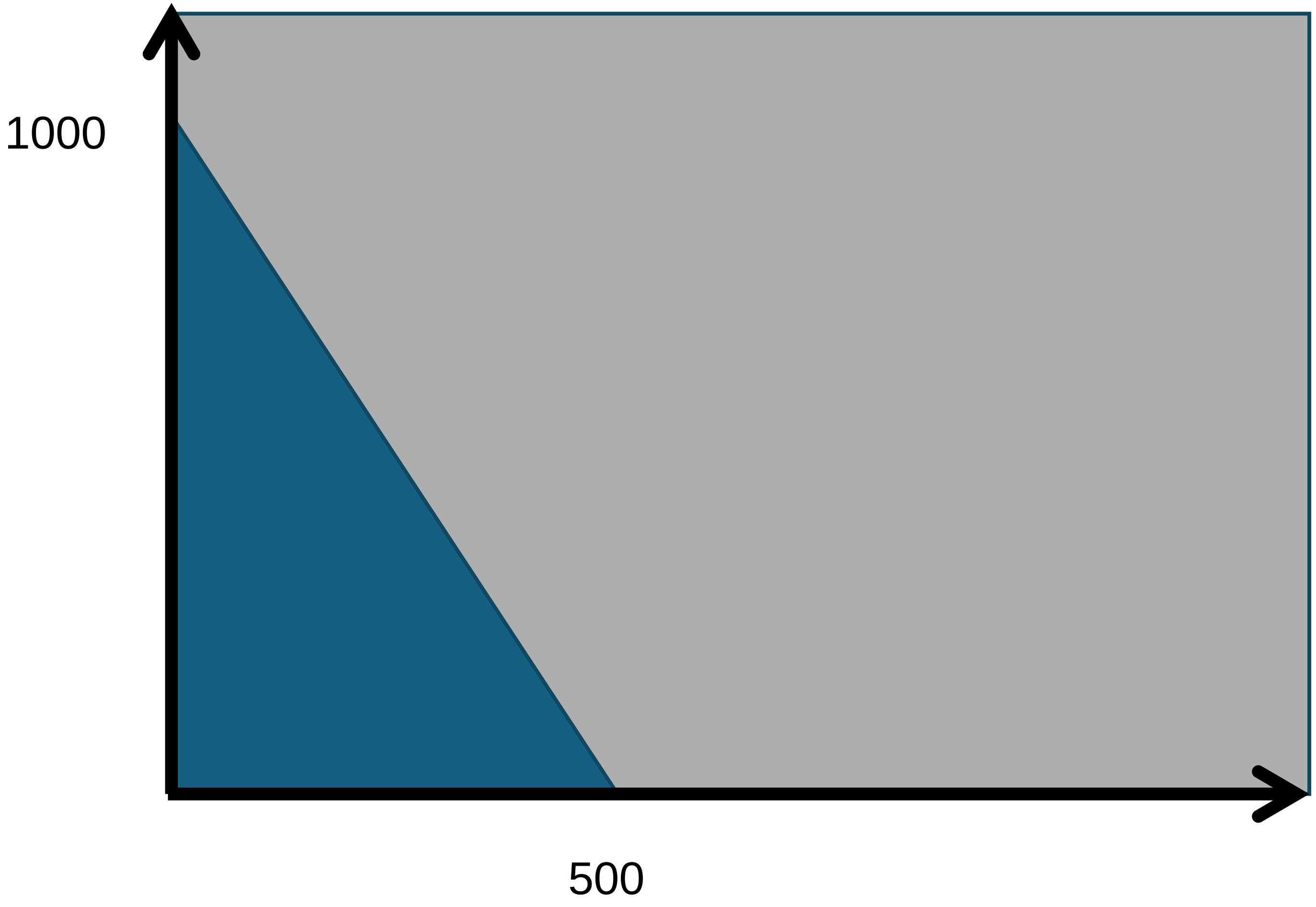
$X_1, X_2 >=0$

$2X_1+X_2<=1000$

$3X_1+4X_2<=2400$

最大化:

$8X_1+5X_2$



条件:

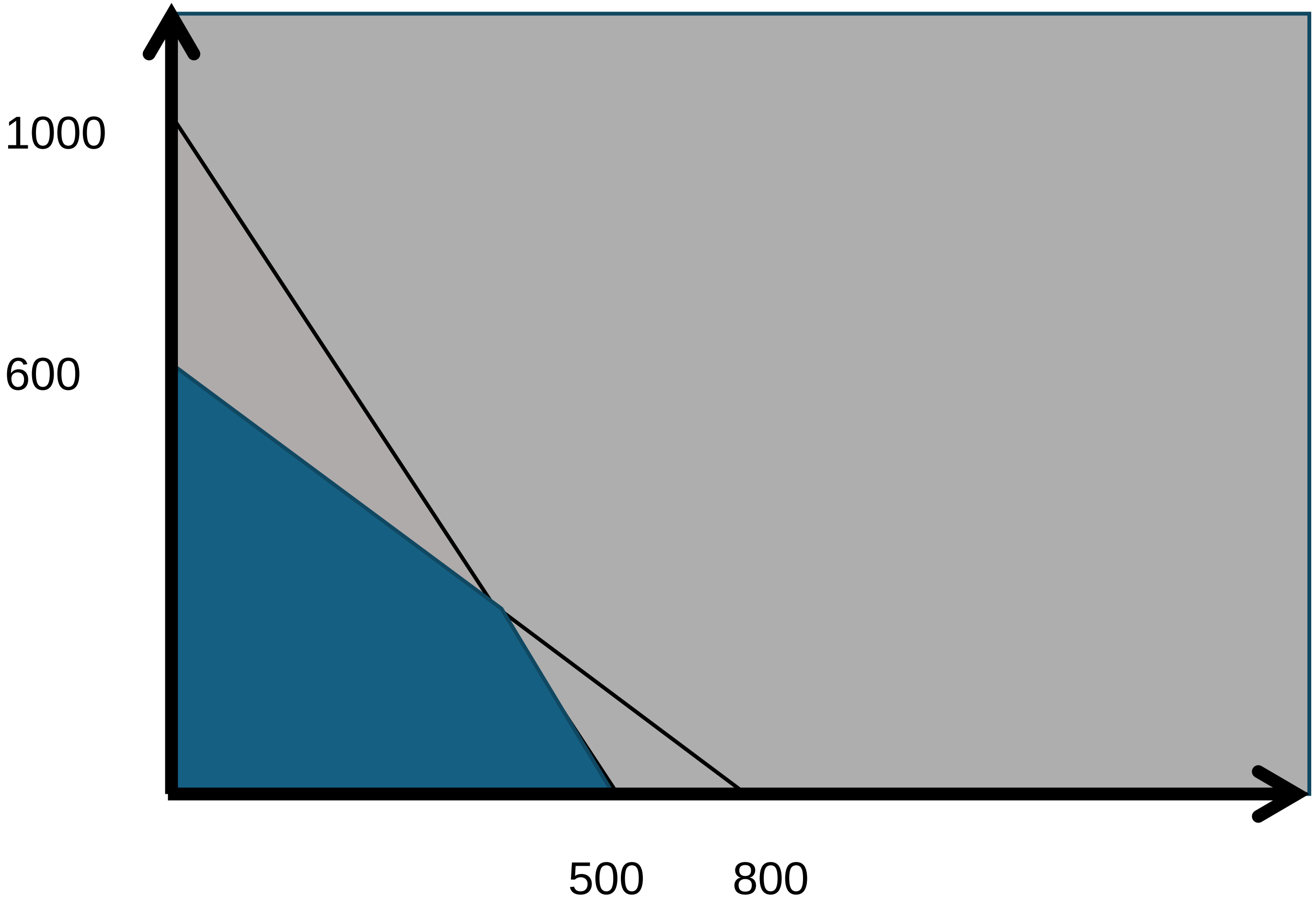
$X_1, X_2 >=0$

$2X_1+X_2<=1000$

$3X_1+4X_2<=2400$

最大化:

$8X_1+5X_2$

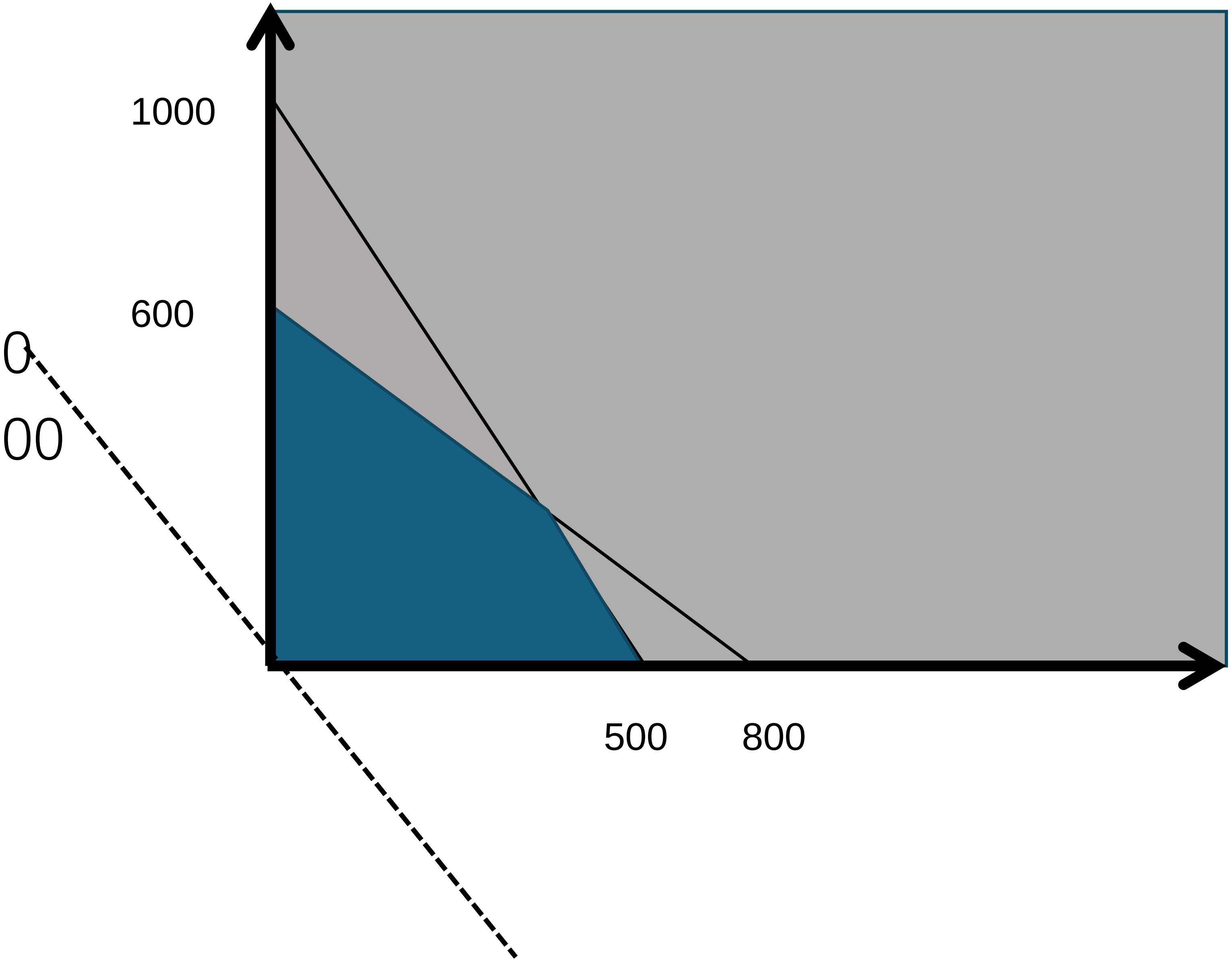


条件:

$X_1, X_2 >= 0$

$2X_1+X_2<=1000$

$3X_1+4X_2<=2400$



最大化:

$8X_1+5X_2$

条件:

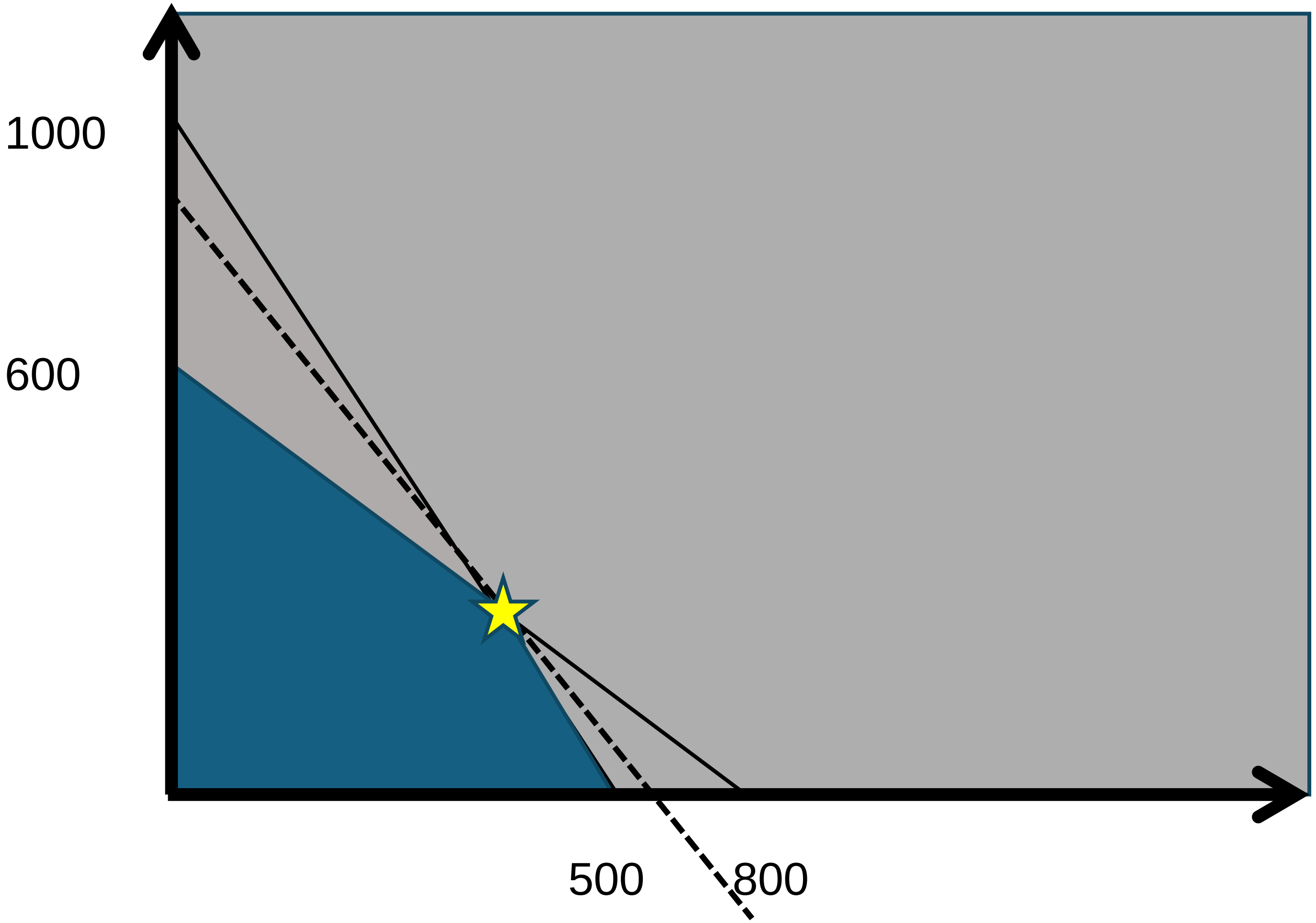
$X_1, X_2 >=0$

$2X_1+X_2<=1000$

$3X_1+4X_2<=2400$

最大化:

$8X_1+5X_2$



混合整数线性规划

Mixed Integer linear program

- 一种数学规划，满足以下条件:
  - 目标函数是线性函数
  - 所有约束条件都是线性函数
  - 一些变量是实数，一些变量是整数，即“混合整数”
- 它几乎与线性规划相同，不同之处在于某些变量是整数。

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

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%

回到我们的调度问题

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况

  $X = \{ x_{i,l}, \quad i = 1,2,\cdots. n; \quad l = 1,2,\cdots, \lambda + 1\}$

  (其中i表示第i个操作，l表示第l个周期)

  $x_{i,l}$ 为 1 仅当操作v_i 在周期l 被调度时， 否则为0

  请注意，这里的λ可以是启发式调度得到的完成时间

  因此，若一共有i个待调度的操作，则X中一共有n*(λ + 1)个变量

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

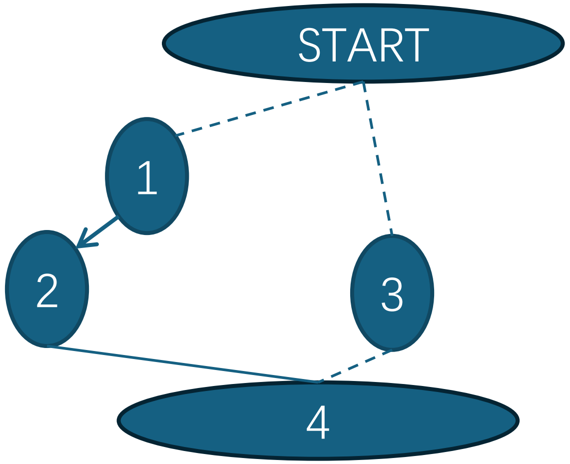
- 用二进制变量的集合X来表示最终的周期调度情况

  $X = \{ x_{i,l}, \quad i = 1,2,\cdots. n; \quad l = 1,2,\cdots, \lambda + 1\}$

  (其中i表示第i个操作，l表示第l个周期)

  $x_{i,l}$ 为 1 仅当操作v_i 在周期l 被调度时，否则为0

$x_{1,1} = 1/0$ $x_{3,1} = 1/0$
$x_{1,2} = 1/0$ $x_{3,2} = 1/0$
$x_{1,3} = 1/0$ $x_{3,3} = 1/0$
$x_{2,1} = 1/0$ $x_{4,1} = 1/0$
$x_{2,2} = 1/0$ $x_{4,2} = 1/0$
$x_{2,3} = 1/0$ $x_{4,3} = 1/0$



$\lambda + 1 = 3$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况

  $X = \{ x_{i,l} , \quad i = 1,2,\cdots. n; \quad l = 1,2,\cdots, \lambda + 1\}$

  （其中i表示第i个操作，l表示第l个周期）

  $x_{i,l}$ 为 1 仅当操作v_i 在周期l 被调度时，否则为0

  每个操作的开始周期为： t_i： $\Sigma_l \, l \cdot x_{il}$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

$x_{1,1} = 1$
$x_{1,2} = 0$
$x_{1,3} = 0$
$x_{2,1} = 0$
$x_{2,2} = 1$
$x_{2,3} = 0$

$x_{3,1} = 0$
$x_{3,2} = 1$
$x_{3,3} = 0$
$x_{4,1} = 0$
$x_{4,2} = 0$
$x_{4,3} = 1$

用二进制变量的集合X来表示最终的周期调度情况

$X = \{ x_{i,l}, \quad i = 1,2,\cdots. n; \quad l = 1,2,\cdots, \lambda + 1\}$

（其中i表示第i个操作，l表示第l个周期）

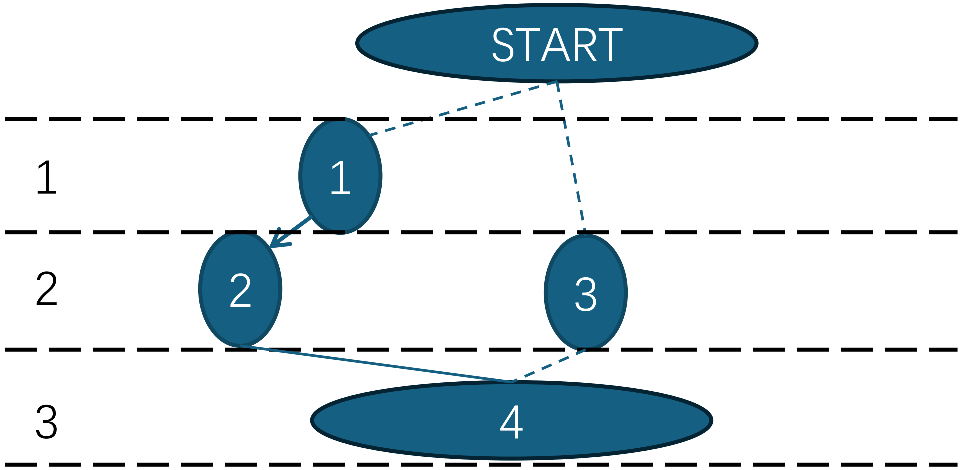
$x_{i,l}$ 为 1 仅当操作v_i 在周期l 被调度时，否则为0

调度结束后每个操作的开始周期为：t_i : $\Sigma_l \, l \cdot x_{il}$

结点1的开始周期：

$t_1=(1*x_{1,1} +2*x_{1,2} +\cdots +(\lambda + 1)*x_{1, \lambda + 1})$

$=> 1*1+2*0+3*0=1$



用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

$x_{1,1} = 1$ $x_{3,1} = 0$
$x_{1,2} = 0$ $x_{3,2} = 1$
$x_{1,3} = 0$ $x_{3,3} = 0$
$x_{2,1} = 0$ $x_{4,1} = 0$
$x_{2,2} = 1$ $x_{4,2} = 0$
$x_{2,3} = 0$ $x_{4,3} = 1$

- 用二进制变量的集合X来表示最终的周期调度情况

  $X = \{ x_{i,l} , \quad i = 1,2,\cdots. n; \quad l = 1,2,\cdots, \lambda + 1\}$

  (其中i表示第i个操作，l表示第l个周期)

  $x_{i,l}$ 为 1 仅当操作v_i 在周期l 被调度时， 否则为0

  每个操作的开始周期为： t_i： $\Sigma_l \; l \cdot x_{il}$

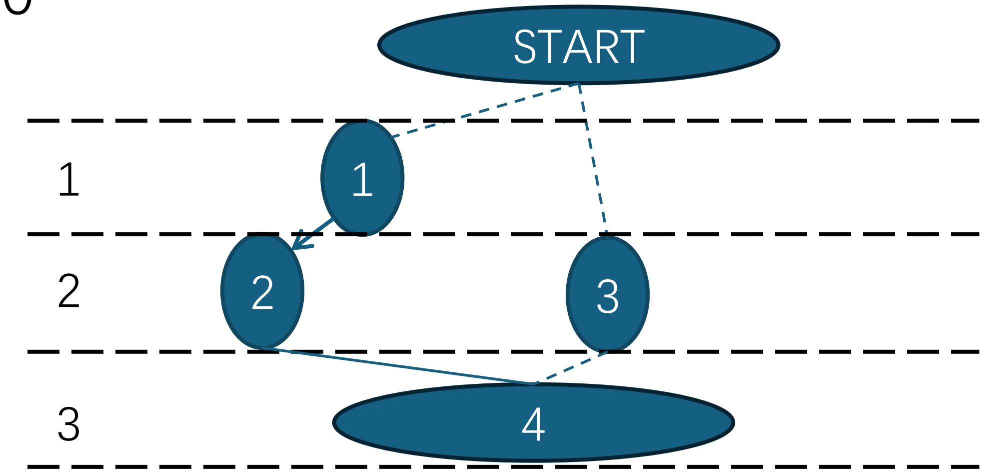
  结点1的开始周期：

  0*0+1*1+2*0=1

  结点2的开始周期：

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

  => 1*0+2*1+3*0=2



用ILP来形式化ML-RCS问题

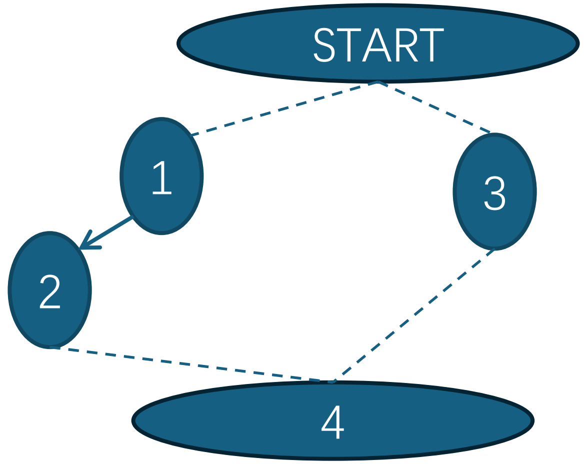
ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 唯一约束（每个操作只会在一个周期开工）
  - 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）
  - 资源约束（资源是有限的）
- 目标:
  - 最小化周期时间

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 唯一约束（每个操作只会在一个周期开工）
  - $\Sigma_l\, x_{i,l} = 1$, $i = 1,2,\cdots. n;$ $l = 1,2,\cdots, \lambda + 1$



$$x_{1,1} + x_{1,2} + \cdots + x_{1,\lambda+1} = 1$$
$$x_{2,1} + x_{2,2} + \cdots + x_{2,\lambda+1} = 1$$
$$x_{3,1} + x_{3,2} + \cdots + x_{3,\lambda+1} = 1$$
$$x_{4,1} + x_{4,2} + \cdots + x_{4,\lambda+1} = 1$$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 顺序约束（当两个结点间有单向边时，代表完成前一个操作后一个操作才能开工）

$$t_i \geq t_j + d_j \quad \rightarrow \quad t_i - t_j - d_j \geq 0 \quad \text{for all } (v_j, v_i) \in E$$

  - 每个操作的开始周期为: t_i: $\Sigma_l\, l \cdot x_{il}$

用ILP来形式化ML-RCS问题

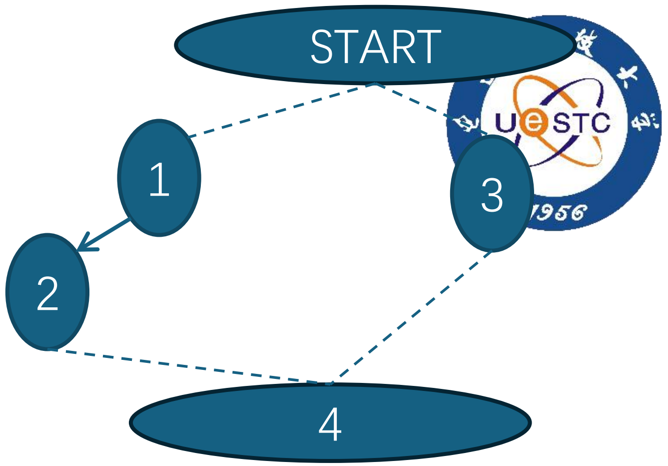
ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 顺序约束（当两个结点间有单向边时，代表完成前一个操作后一个操作才能开工）

$$\underbrace{\Sigma_l\, l \bullet x_{i,l}}_{t_i} - \underbrace{\Sigma_l\, l \bullet x_{j,l}}_{t_j} - d_j \geq 0 \qquad \text{for all } (v_j, v_i) \in E$$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS



- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 顺序约束（当两个结点间有单向边时，代表完成前一个操作后一个操作才能开工）

$$t_i \geq t_j + d_j \quad \rightarrow \quad t_i - t_j - d_j \geq 0 \quad \text{for all } (v_j, v_i) \in E$$

  - 每个操作的开始周期为： t_i： $\Sigma_l\, l \cdot x_{il}$

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

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

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

$t_4=(1*x_{3,1} +\cdots +(\lambda + 1)*x_{3,\lambda+1})$

对于从1指向2的边有： $t_2 - t_1 - 1 >= 0$

对于从2指向4的边有： $t_4 - t_2 - 1 >= 0$

对于从3指向4的边有： $t_4 - t_3 - 1 >= 0$

用ILP来形式化ML-RCS问题

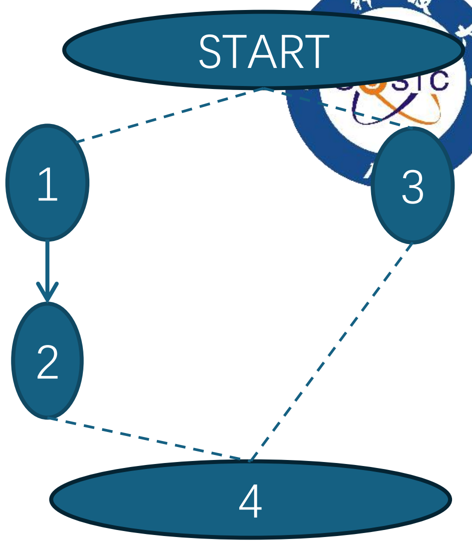
ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 资源约束（资源是有限的）

$$\sum_{i:T(v_i)=k} x_{i,l} \leq a_k \qquad \text{for all } l \text{ and } k$$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS



- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 资源约束（资源是有限的）

$$\sum_{i:T(v_i)=k} x_{i,l} \le a_k \qquad \text{for all l and k}$$

当都是加法操作且有一个加法器时

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

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

…

对周期$\lambda + 1$： $x_{1,\lambda+1} + x_{2,\lambda+1} +x_{3,\lambda+1} + x_{4,\lambda+1} <=1$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

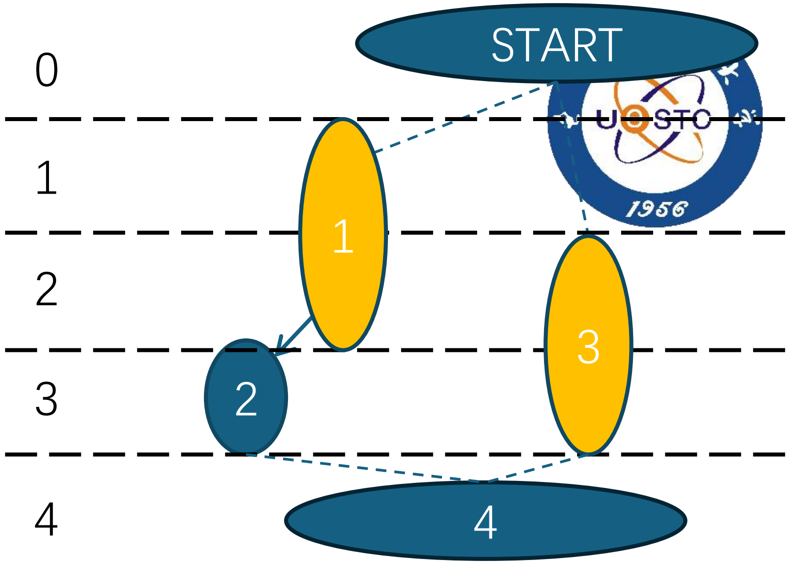
- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 资源约束（资源是有限的）

$$\sum_{i:T(v_i)=k} x_{i,l} \leq a_k \qquad \text{for all } l \text{ and } k$$

好像还不够?

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS



- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 资源约束（资源是有限的）

$$\sum_{i:T(v_i)=k} x_{i,l} \le a_k \qquad \text{for all } l \text{ and } k$$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

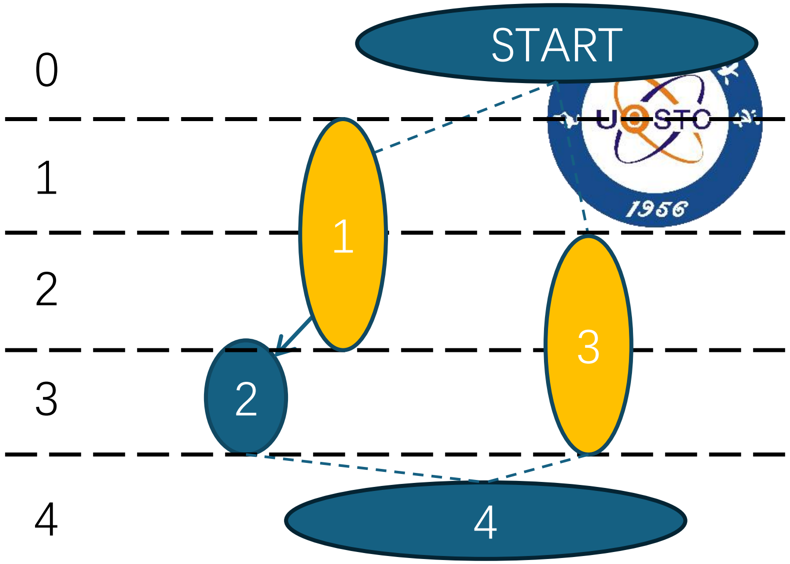
- 用二进制变量的集合X来表示最终的周期调度情况

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

  - 上面这个式子为1时代表操作i在周期l正在工作，否则为0

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS



- 用二进制变量的集合X来表示最终的周期调度情况

$$x_{1,1} = 1$$
$$x_{1,2} = 0$$
$$x_{1,3} = 0$$
$$x_{1,4} = 0$$

操作1在周期2有没有在工作:

$m=2-(2-1)=1$

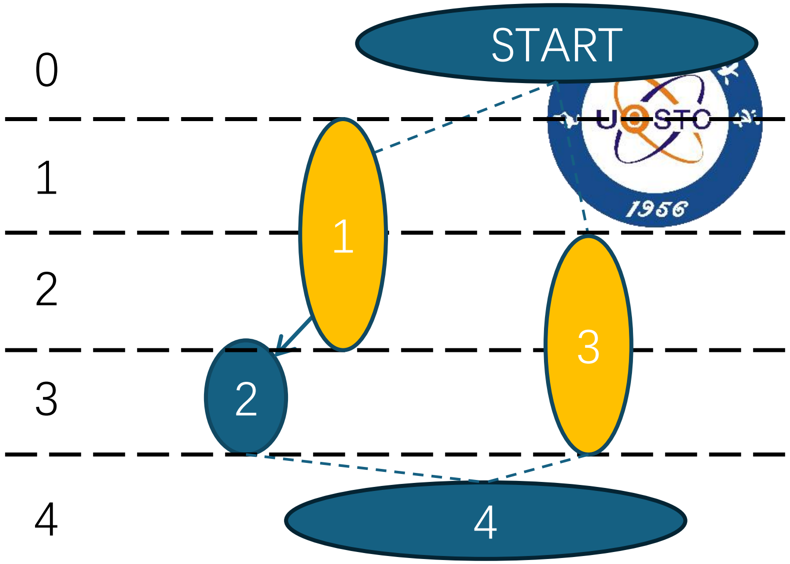
$x_{1,1}+ x_{1,2}=1$，有在工作

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

- 上面这个式子为1时代表操作i在周期l正在工作，否则为0

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS



- 用二进制变量的集合X来表示最终的周期调度情况

$x_{1,1} = 1$
$x_{1,2} = 0$
$x_{1,3} = 0$
$x_{1,4} = 0$

操作1在周期3有没有在工作:

m=3-(2-1)=2

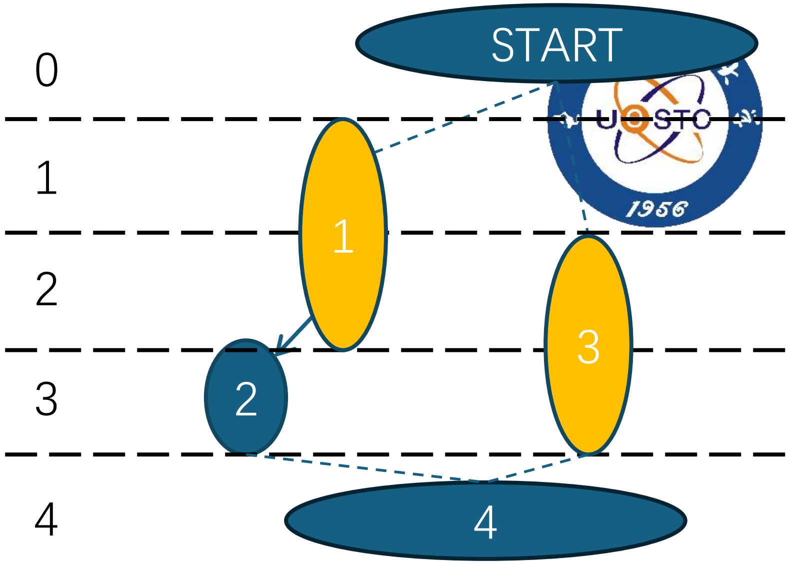
$x_{1,2}+ x_{1,3}=0$，没在工作

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

- 上面这个式子为1时代表操作i在周期l正在工作，否则为0

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS



- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 资源约束（资源是有限的）

$$\sum_{i:T(v_i)=k} x_{i,l} \le a_k \qquad \text{for all l and k}$$

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

用ILP来形式化ML-RCS问题

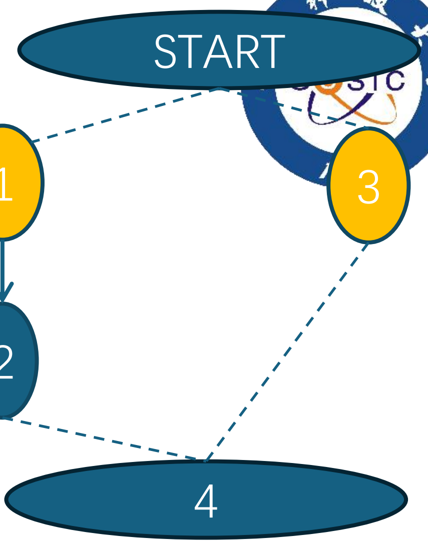
ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 资源约束（资源是有限的）

$$\sum_{i:T(v_i)=k} \; \sum_{m=l-(d_i-1)}^{l} x_{i,m} \le a_k \qquad \text{for all } l \text{ and } k$$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS



- 用二进制变量的集合X来表示最终的周期调度情况
- 条件:
  - 资源约束（资源是有限的）

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

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

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

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

- 用二进制变量的集合X来表示最终的周期调度情况
- 目标:
  - 最小化周期时间

$$\text{Min: } t_n = \Sigma_l\, l \cdot X_{n,l}$$

用ILP来形式化ML-RCS问题

ILP Formulation of ML-RCS

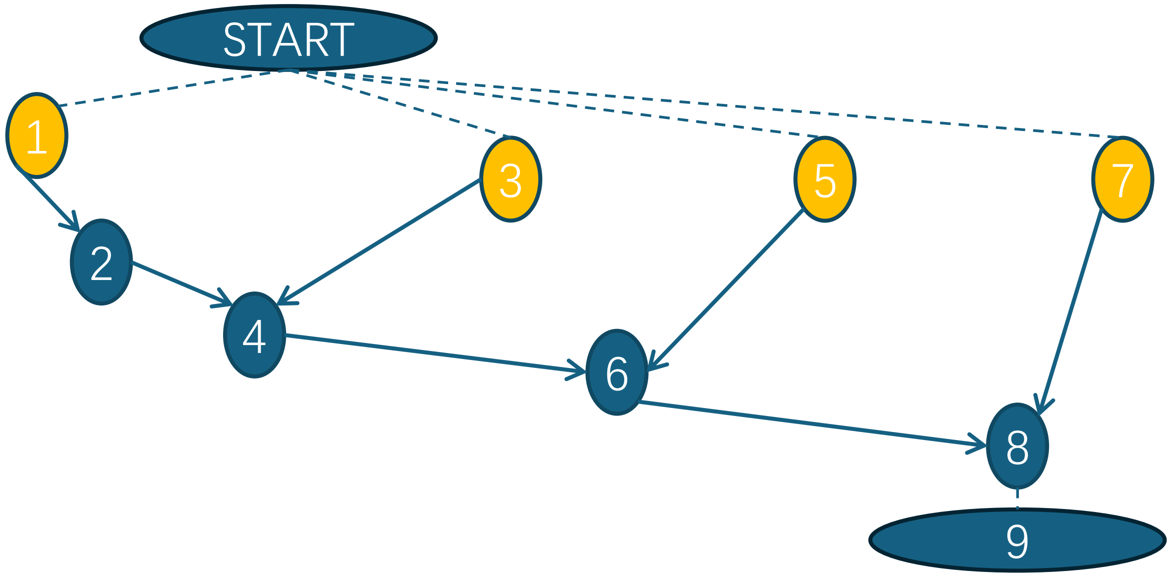
$$min\ \ \Sigma_l\ l \cdot X_{n,l}\ such\ that$$

$$\sum_j x_{ij} = 1 \qquad i = 1, 2, \ldots, n, \qquad l = 1, 2, \ldots, \lambda + 1$$

$$\sum_l l \bullet x_{il} - \sum_l l \bullet x_{jl} - d_j \geq 0 \qquad i, j = 1, 2, \ldots, n, (v_j, v_i) \in E$$

$$\sum_{i:T(v_i)=k} \ \sum_{m=l-d_i+1}^{l} x_{im} \leq a_k \qquad k = 1, 2, \ldots, n_{res}; l = 0, 1, \ldots, t_n$$

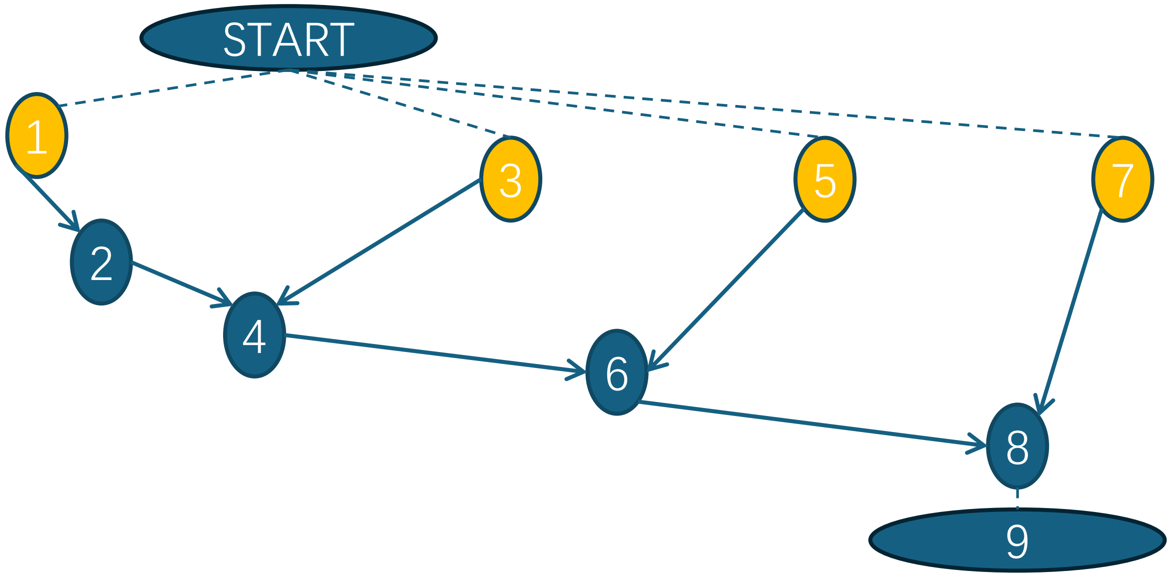
例子



假设黄色是乘法操作，蓝色是加法操作，有两个乘法器和一个加法器，乘法延迟为2，加法延迟为1

列表调度算出来的λ是6

例子



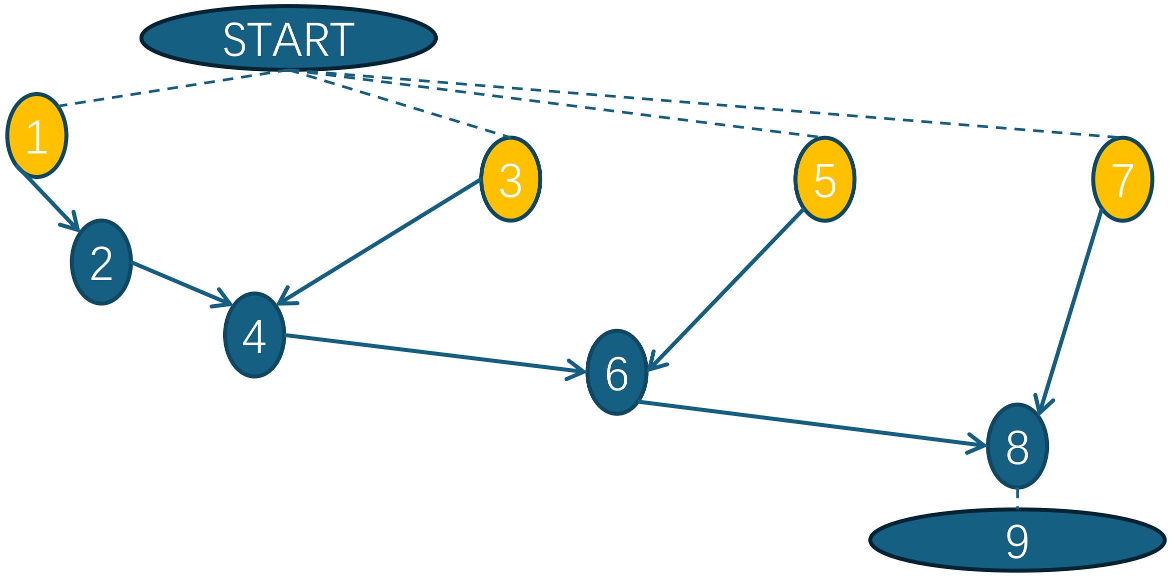
假设黄色是乘法操作，蓝色是加法操作，有两个乘法器和一个加法器，乘法延迟为2，加法延迟为1

列表调度算出来的λ是6

- 唯一约束（每个操作只会在一个周期开工）
- $\sum_l x_{i,l} = 1$

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

例子



假设黄色是乘法操作，蓝色是加法操作，有两个乘法器和一个加法器，乘法延迟为2，加法延迟为1

列表调度算出来的λ是6

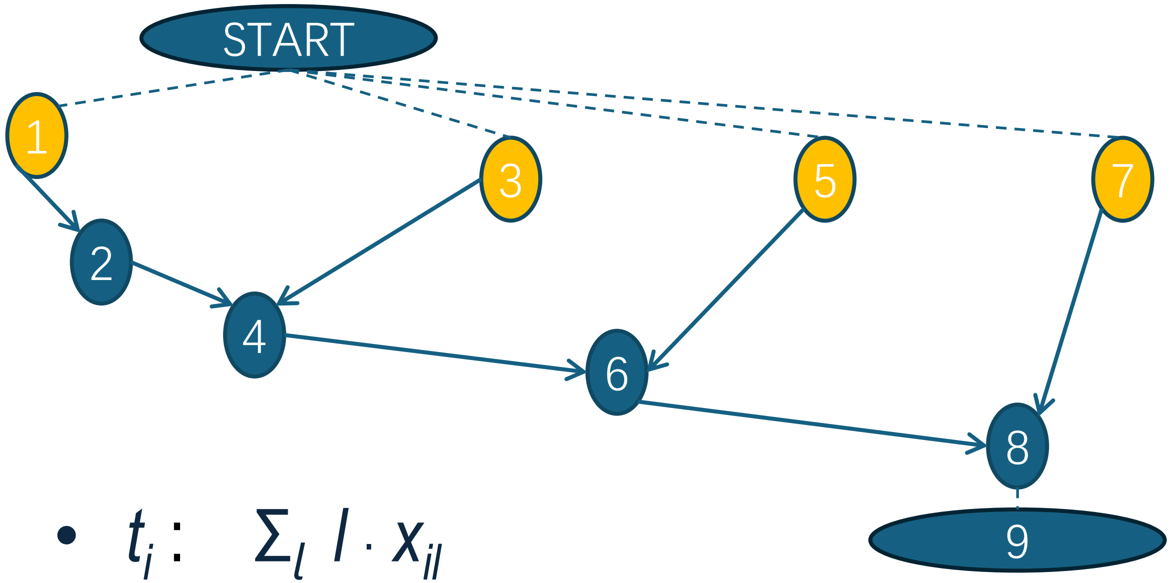
- 唯一约束（每个操作只会在一个周期开工）
- $\sum_l x_{i,l} = 1$

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

…

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

例子



假设黄色是乘法操作，蓝色是加法操作，有两个乘法器和一个加法器，乘法延迟为2，加法延迟为1

列表调度算出来的λ是6

- t_i : $\Sigma_l\ l \cdot x_{il}$

- 顺序约束（当两个结点间有单向边时，代表完成前一个操作后一个操作才能开工）
- $\Sigma l\ l \bullet x_{i,l} - \Sigma l\ l \bullet x_{j,l} - d_j \geq 0$ for all $(v_j, v_i) \in E$

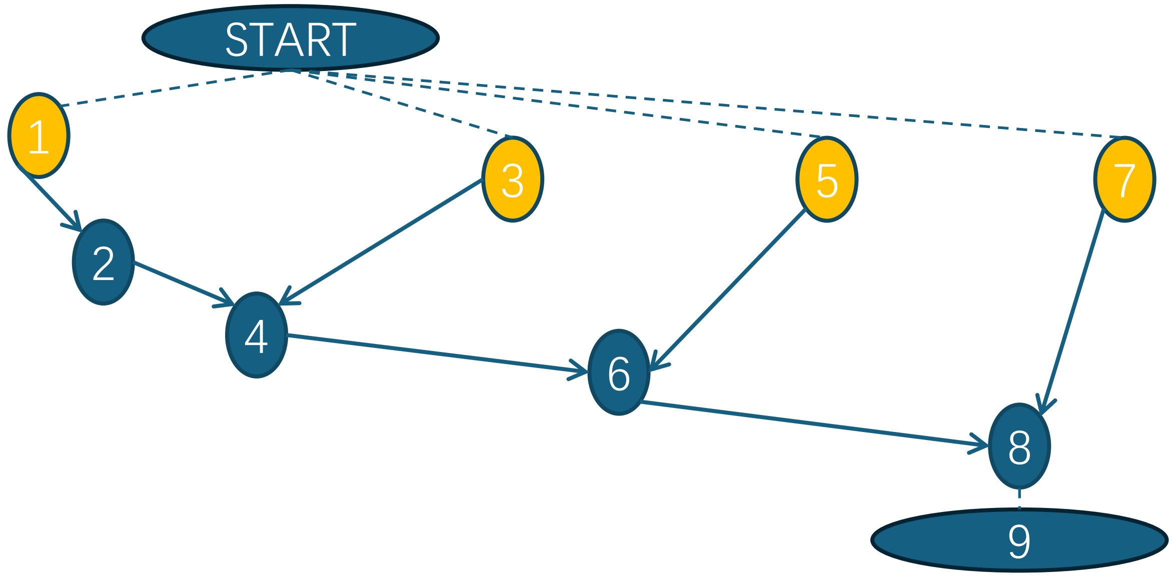
t1=1*x11+2*x12+3*x13+4*x14+5*x15+6*x16+7*x17

…

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

t2-t1-2>=0	t6-t5-2>=0
t4-t2-1>=0	t8-t6-1>=0
t4-t3-2>=0	t8-t7-2>=0
t6-t4-1>=0	t9-t8-1>=0

例子



假设黄色是乘法操作，蓝色是加法操作，有两个乘法器和一个加法器，乘法延迟为2，加法延迟为1

列表调度算出来的λ是6

- 资源约束（资源是有限的）

$$\sum_{i:T(v_i)=k} \; \sum_{m=l-(d_i-1)} x_{i,m} \le a_k \qquad \text{for all l and k}$$

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

…

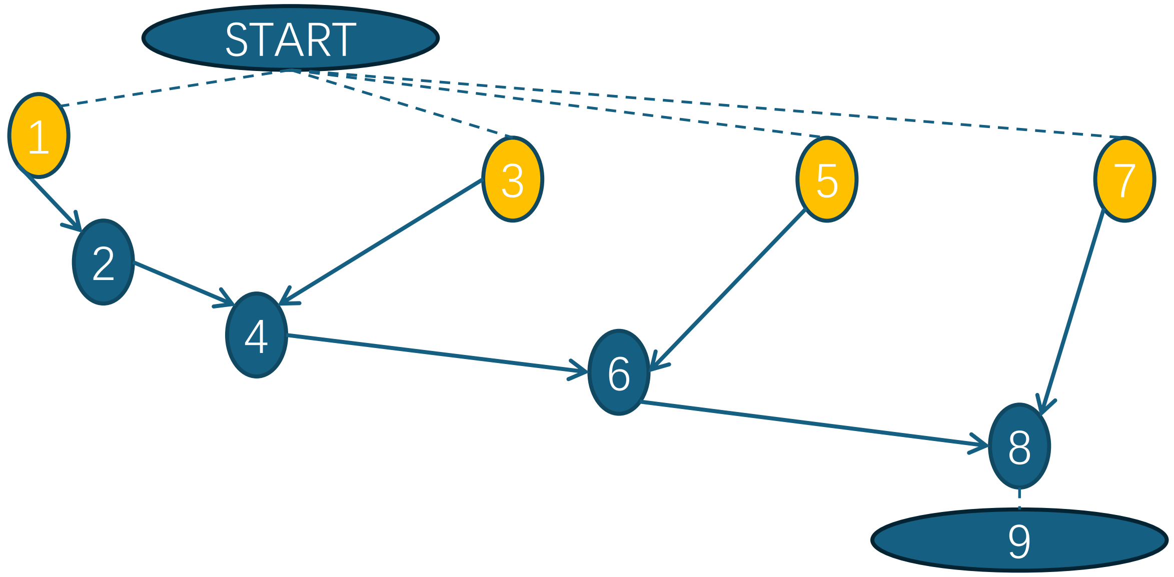
(x16+x17)+(x36+x37)+(x56+x57)+(x76+x77)<=2

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

…

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

例子



假设黄色是乘法操作，蓝色是加法操作，有两个乘法器和一个加法器，乘法延迟为2，加法延迟为1

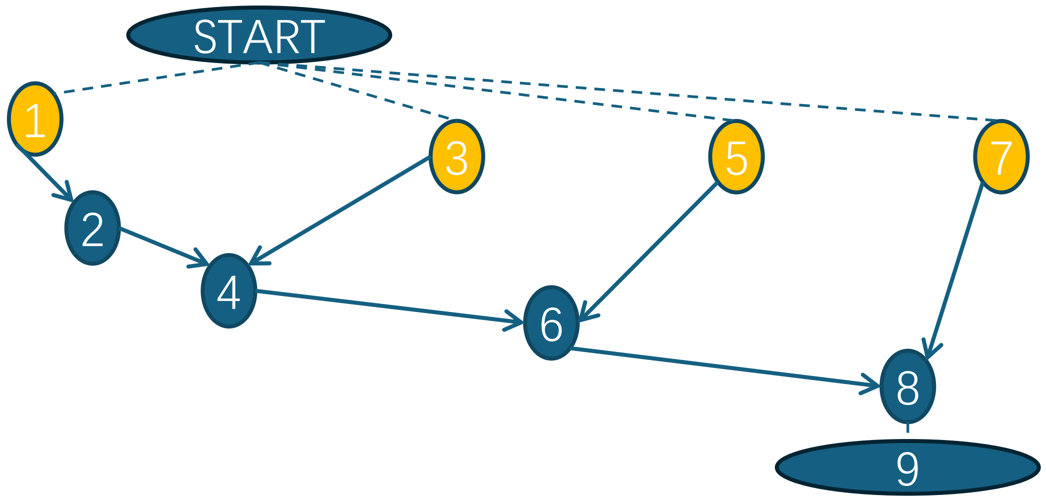
列表调度算出来的λ是6

- 目标：最小化周期时间

$$\Sigma_l \; l \cdot X_{n,l}$$

最小化：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:
  - 最小化周期时间



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 …

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

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

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

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

(0+x11)+(0+x31)+(0+x51)+(0+x71)<=2 …

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

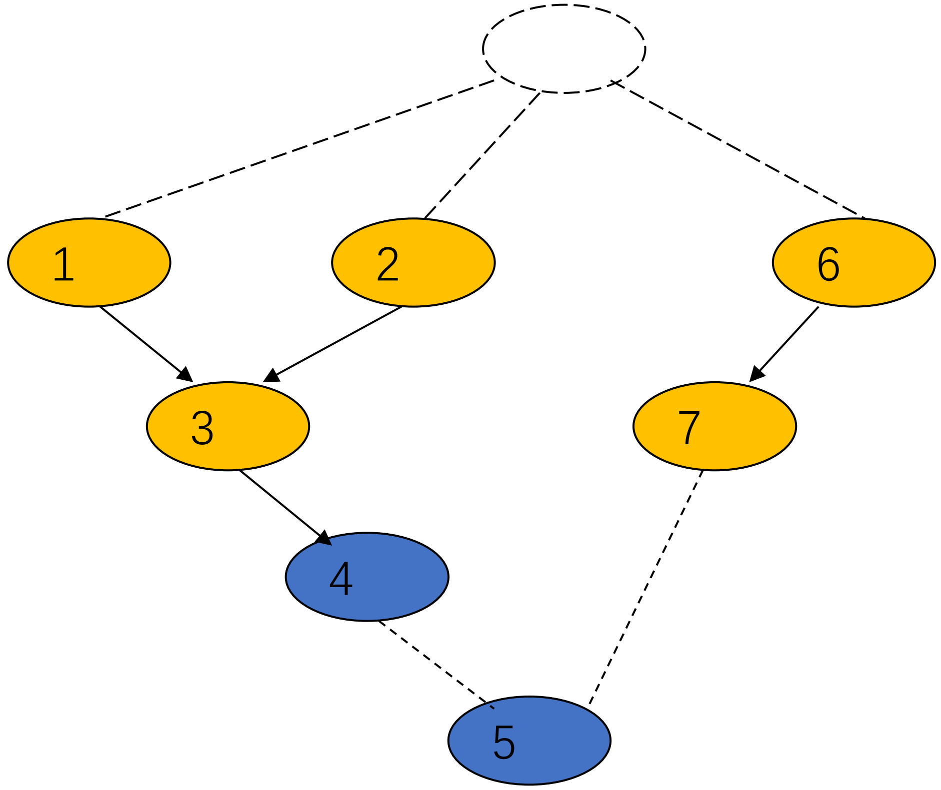
x21+x41+x61+x81+x91<=1 …

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

Binary

x11 x12 x13 x14 x15 x16 x17…

x91 x92 x93 x94 x95 x96 x97



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

目标是最小化$t_n = \sum_l l \cdot X_{n,l}$

- 唯一约束（每个操作只会在一个周期开工）

$$\sum_l x_{i,l} = 1$$

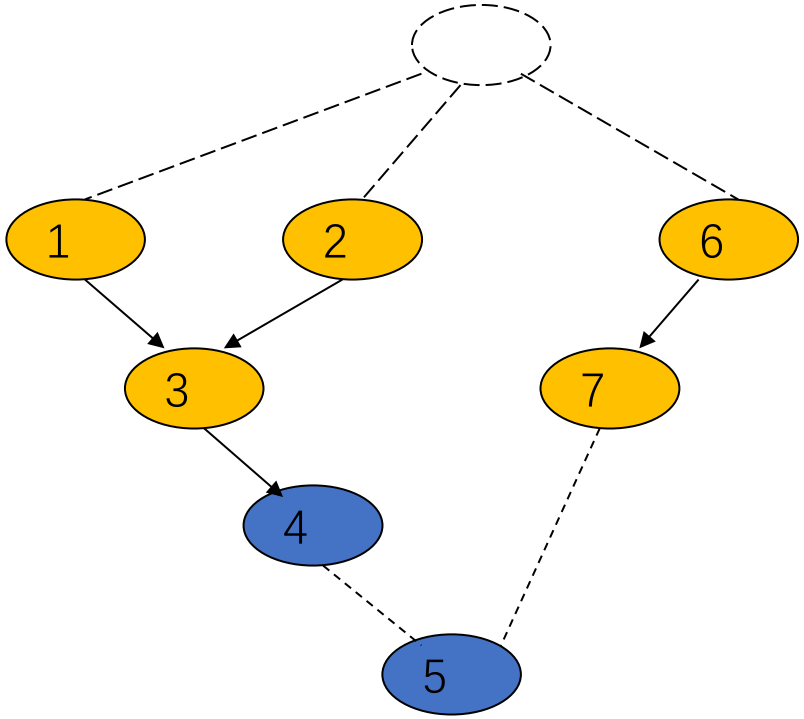
- 顺序约束（当两个结点间有单向边时，代表完成前一个操作后一个操作才能开工）

$$\sum_l l \cdot x_{i,l} - \sum_l l \cdot x_{j,l} - d_j \geq 0 \qquad \text{for all } (v_j, v_i) \in E$$

- 资源约束（资源是有限的）

$$\sum_{i:T(v_i)=k} \; \sum_{m=l-(d_i-1)}^{l} x_{i,m} \leq a_k \qquad \text{for all } l \text{ and } k$$

- 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:
  - 最小化周期时间



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

x10 x11 x12 x13 x14 x15 x16 …

x70 x71 x72 x73 x74 x75 x76