§10 Constructing and evaluating preselective policies

Systematic construction similar to ES-policies along a "conflict settling tree"

root = G

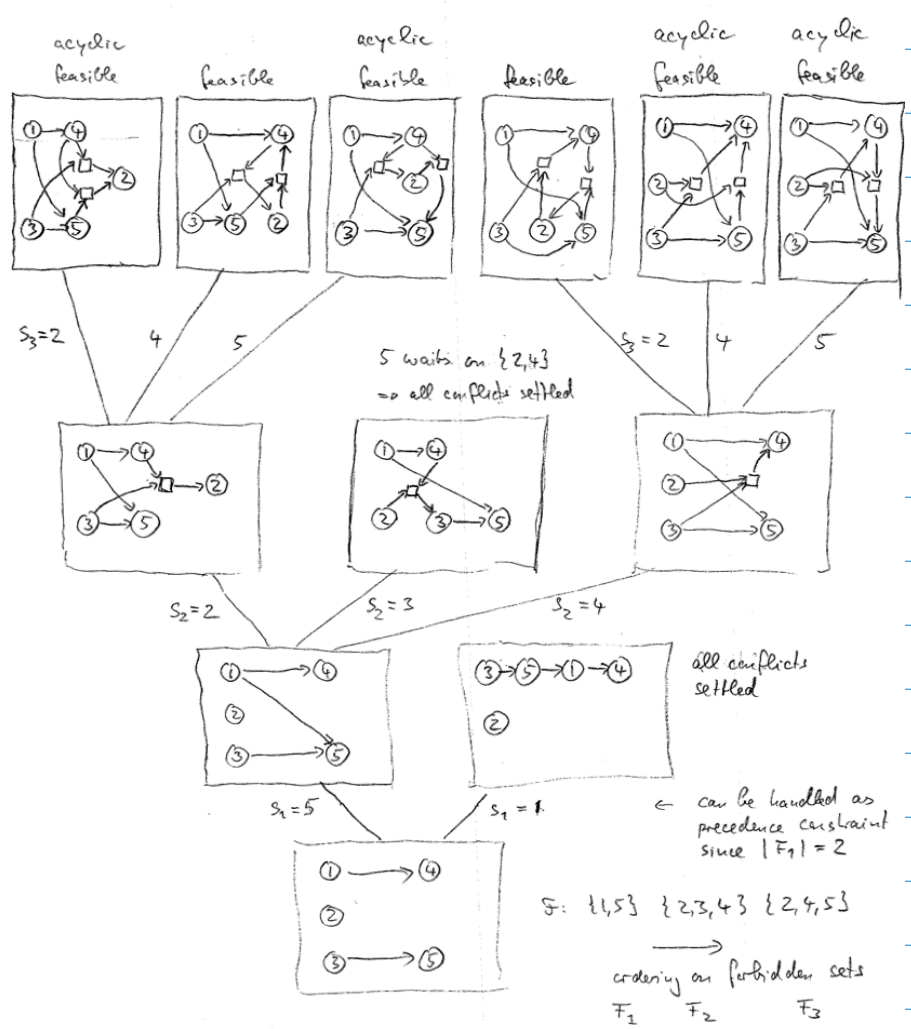
nodes = AND-OR networks arising from choices of waiting jobs for some forbidden sets

children of a node D

= all AND-OR networks obtained from D by choosing a waiting job from one yet unsettled forbidden set

need to check that

may use suitable ordering of forbidden sets

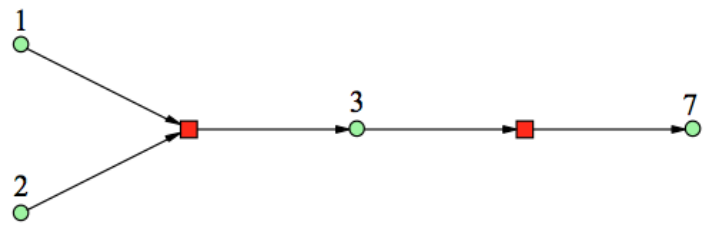


Checking if a forbidden set F is already settled by the partial selection $(s_1, \dots, s_r)$

$\hat{=}$ finding forced waiting conditions of the form $(F \setminus \{j\}, j)$

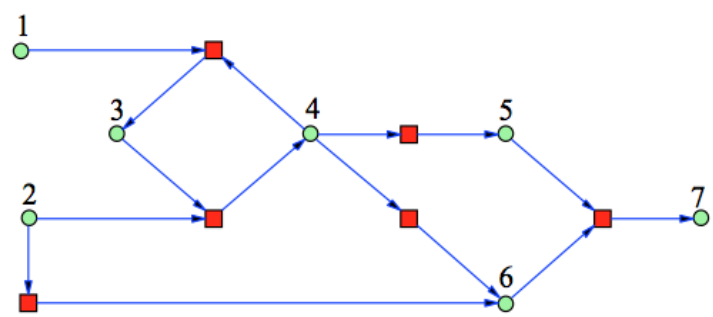
$\hat{=}$ "transitivity" of waiting conditions

Finding "transitive" waiting conditions



easy here

Is ({1,2},7) implied by the given system $\mathcal{W}$?



not so easy in this case

Def: j is forced to wait for U $:\Longleftrightarrow$ all linear realizers have any $i \in U$ before j

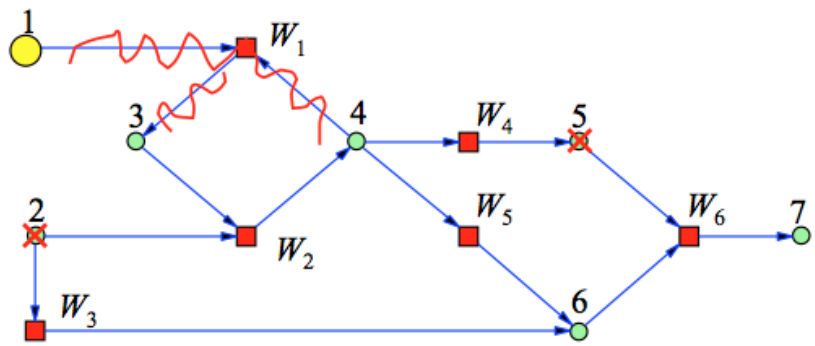
An algorithm for finding forced waiting conditions

- **Input**: Jobs V, feasible waiting conditions $\mathcal{W}$, set $U \subseteq V$
- **Output**: A list L of jobs

(U, j) is a forced waiting condition $\Leftrightarrow j \notin L$ (and $\notin U$)

List L := [ ]
while (there is a job $i \in V \setminus U$ that is not a waiting job in $\mathcal{W}$)
  begin
    insert i at the end of L and delete it from V
    if (some waiting condition (X, j) becomes satisfied)
      delete (X, j) from $\mathcal{W}$
  end
return L

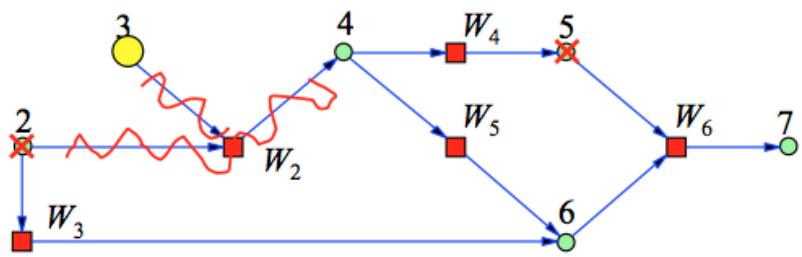
$U = \{2, 5\}$



x ∈ is in U

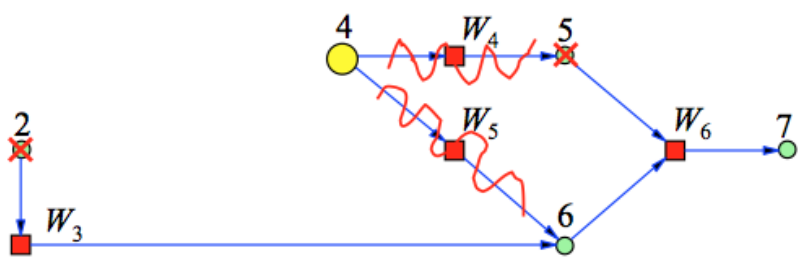
$L = [\ 1,\qquad\qquad\qquad\ ]$

$U = \{2, 5\}$



$L = [\ 1, 3,\qquad\qquad\ ]$

$U = \{2, 5\}$



$L = [\ 1, 3, 4,\qquad\ ]$

$U = \{2, 5\}$

$L = [\ 1, 3, 4,\qquad\qquad\qquad\ ]$ $\Rightarrow$ termination with $L = [\ 1, 3, 4\ ]$

Claim: 6, 7 wait for $U = \{2, 5\}$

Correctness of the algorithm

(U, j) is a forced waiting condition $\Leftrightarrow j \notin L$ (and $\notin U$)

10.2 THEOREM

Key Lemma:

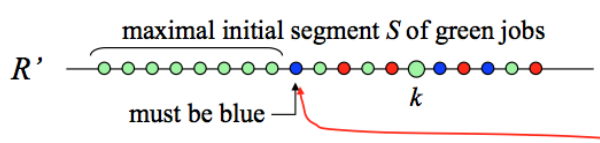
There is a linear realization of $\mathcal{W}$ starting with all jobs j for which (U, j) is not a forced waiting condition

10.3 LEMMA

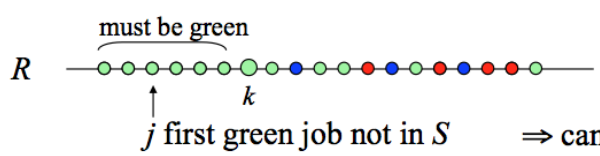
$\Rightarrow$ only U and forced waiting jobs are not in L at termination of the algorithm

Proof of key lemma (by contradiction)

● ○ ● ~ job is / is not in a forced waiting condition with U / is in U



k green $\Rightarrow$ there is a linear realization R with U after k



$\Rightarrow$ cannot occur here

$\Rightarrow$ there is a waiting condition (X, j) with X is after S in R'
$\Rightarrow$ there is some $i \in X$ before j in R
but all jobs before j in R are in $S \Rightarrow$ contradiction

R' realizer of $\mathcal{W}$ with maximal initial segment of green jobs

Computing earliest start dates for a preselective policy

Input: preselective policy Π, processing time vector x

Output: vector $\Pi[x]$ of start times

= ES w.r.t. to system $\mathcal{W}$ of waiting conditions given by selection s and graph G

translate this to an algorithm on the AND/OR-network representing Π and G

$\Downarrow$ §9

Solve a system of min-max inequalities and compute the unique componentwise minimal solution $ES_{\mathcal{W}}$

special case here:

have positive arc weights

$d_{jw} =: x_j$ for arcs $j \xrightarrow{x_j} \boxed{w}$

and $d_{wj} = 0$ $\quad \boxed{w} \longrightarrow j$

$\Downarrow$

Dijkstra-like algorithm

general case

arbitrary arc weights

d_{jw}, d_{wj}

(also negative)

$\Downarrow$

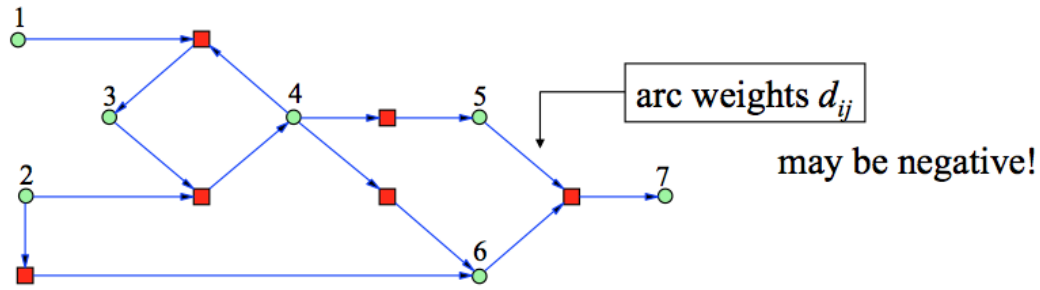
no polynomial algo known

open if NP-complete

known to be in

$NP \cap coNP$

Earliest start times



Start times S_j of jobs must satisfy

$$\left.\begin{array}{ll} j \in V & \text{AND node: } S_j \geq \max\limits_{(w,j)\in A} \{S_w + d_{wj}\} \\ w \in W & \text{OR node: } S_w \geq \min\limits_{(j,w)\in A} \{S_j + d_{jw}\} \end{array}\right\} \text{min-max system}$$

$$S_j \geq 0$$

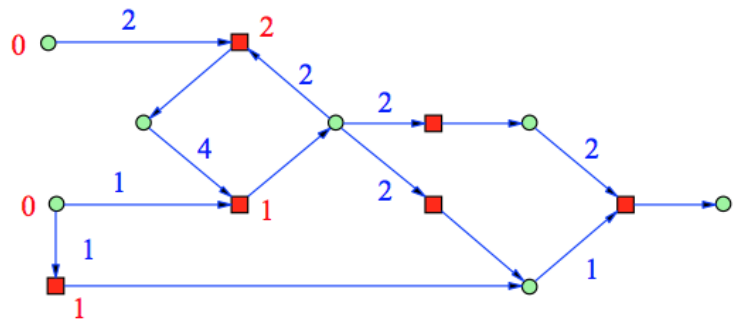
$d_{wj} = 0$, $d_{jw} = x_j > 0$ for scheduling

Related work

- Knuth [1977] Shortest paths
- Igelmund & Radermacher [1983] Scheduling
- Dinic [1990] Shortest paths
- Gallo, Longo, Pallatino &Nguyen [1993] Directed hypergraphs
- Zwick & Paterson [1995] Game theory
- Schwindt [1996] Scheduling
- Schwiegelshohn & Thiele [1997] Min/Max inequalities
- Chauvet, Levner & Proth [1998] Scheduling
- Jurdzinski [1998] Game theory
- Levner, Sung & Vlach [2000] Scheduling
- Adelson-Velsky & Levner [2002] Scheduling

A Dijksta-like algorithm for positive arc weights

Algorithm 10.4

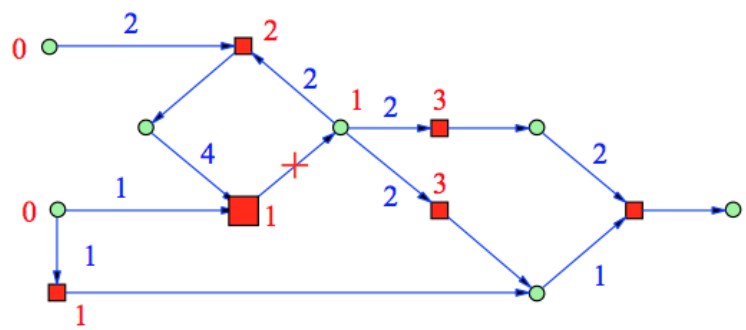


Init

Assume w.l.o.g that $d_{wj} = 0$ (only one outgoing arc per OR node) and that waiting conditions are feasible

set $S_j := 0$ if there is no $(w, j) \in A$, set $S_j := \infty$ for all other jobs, set $S_w = \infty \ \forall w$

for unmarked OR nodes $w \in \text{out}(j)$, set $S_w = \min\{S_k + d_{kw} \mid (k, w) \in A\}$



choose unmarked OR-node $w = (X, j)$ with minimum S_w

mark w and reduce indegree of j by 1

if indegree$(j) = 0$ then

  set $S_j := \max\{S_w \mid (w, j) \in A\}$

  for unmarked OR nodes $w \in \text{out}(j)$, set $S_w = \min\{S_k + d_{kw} \mid (k,w) \in A\}$

repeat

main loop

10.5 THEOREM: Algorithm 10.4 computes the unique minimal feasible solution ≥ 0 of the min-max system given by the AND/OR-graph $D = (V \cup W, A)$ in $O(|V| + |W| \cdot \log|W| + |A|)$ time

Proof: (1) Correctness

Let S be the vector of start times constructed by the algorithm

Let S^* be the ES-vector (see lemma 9.6) of $V \cup W$.

Assume that $S \neq S^*$

Then choose node v with $S_v > S_v^*$ and S_v^* minimum

<u>Case 1</u> v is an AND-node

$\Rightarrow \exists$ OR-node $w = (X, v)$ with $S_w = S_v + \overbrace{d_{wv}}^{0}$

$\boxed{w} \longrightarrow (v)$ $\Rightarrow S_w = S_v > S_v^* \geq S_w^*$

choice of v $\Rightarrow S_v^* = S_w^*$ and $S_w > S_w^*$

$\Rightarrow$ reduced to the case that v is an OR-node

<u>Case 2</u> v is an OR-node

$\Rightarrow \exists$ AND-node i with $\boxed{S_v^* = S_i^* + \underbrace{d_{iv}}_{>0}}$ $(*)$

$(i) \longrightarrow \boxed{v}$

$\boxed{\text{Claim: } S_i > S_i^*}$

Suppose not, i.e. $S_i = S_i^*$

then, when the algorithm assigns to i the value S_i,

S_v is set to $\min\limits_{(j,v)} \{ S_j + d_{jv} \} \leq S_i + d_{iv} = S_i^* + d_{iv} = S_v^*$

if v is unmarked

otherwise $S_v \leq S_i + d_{iv} = S_i^* + d_{iv} = S_v^*$

in both cases $S_v \leq S_v^*$, contradiction to $S_v > S_v^*$

So $S_i > S_i^*$ and $S_i^* < S_v^*$ because of $(*)$

$\Rightarrow$ contradiction to the choice of v $\square$

(2) Run time: Exercise $\square$

AND-OR-networks feature two notions of feasibility

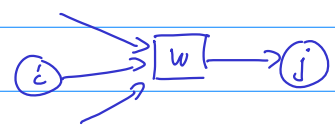
structural feasibility
i.e. $\exists$ realizer

the min-max system
has a feasible solution S

Question? Is there a connection between the two

10.6 LEMMA: The two notions of feasibility are equivalent if $d_{jw} > 0$, $d_{wj} \geq 0$ (this includes the scheduling case)

Proof: all $d_{jw} > 0$ $\Rightarrow$ $\forall\ w = (X, j)$ $\exists\ i \in X$ with $S_i < S_w \leq S_j$ for every feasible solution S of the min-max system



$$S_i + \underbrace{d_{iw}}_{>0} \leq S_w \leq S_j$$

The arcs (i,j) define a realizer of W
If not, they contain a directed cycle and we would have $S_i < S_i$ for every arc on the cycle, which cannot be the case

In the other direction, if W has a realizer R, then ES_R is a feasible solution of the min-max system $\square$

The equivalance may not hold if negative values d_{jw} and d_{wj} are allowed

solvability of a general system of min–max inequalities is in NP ∩ coNP

Certificate for SOLVABILITY ∈ NP

Any feasible schedule $S = (S_1, \ldots, S_n)$ is a certificate for SOLVABILITY ∈ NP

Certificate for SOLVABILITY ∈ coNP

Let $S = (S_1, \ldots, S_n)$ be the unique minimal solution (maybe with ∞)

some S_j or S_w may be ∞

For every AND node j, one can delete all but one incoming arcs without changing S_j

needs a little proof

Then every cycle has non-negative length

needs a little proof

Relaxing in every AND-node
⇒ relaxed problem with only min inequalities (~ OR nodes)
⇒ can check $S_j > K$ by shortest path algorithms in polynomial time
⇒ relaxed problem is certificate for SOLVABILITY ∈ coNP

Complexity of checking solvability

A schedule and a relaxed subproblem are polynomially checkable certificates for membership in NP and coNP

⇓

SOLVABILITY ∈ NP ∩ coNP

no polynomial algorithm known, not known to be NP-complete
or coNP-complete

same complexity status as

LINEAR PROGRAMMING and PRIMES

$\in NP \cap coNP$

was known in the 60ies

first polynomial algo = Ellipsoid Method

early 70ies

$\in NP$

2002 Agarwal, Kayal, Saxena

needs number-theoretic methods

$\in coNP$

easy, a factorization

$\exists$ simple pseudo-polynomial algo for MIN-MAX systems

Exercises:

10.1 Show that Algorithm 10.4. can be implemented to run in $O(|V| + |W| \cdot \log |W| + |A|)$ time

10.2* Derive a polynomial-time algorithm for finding the unique minimal (feasible) solution ≥ 0 of a min-max system with non-negative arc weights

Hint: Try to relate feasibility of the min-max system to structural feasibility in the sense of Lemma 10.6. What changes?

10.3 Derive a pseudopolynomial-time algorithm for finding the unique minimal (feasible) solution ≥ 0 of a min max system with arbitrary arc weights