§ 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 on some forbidden sets

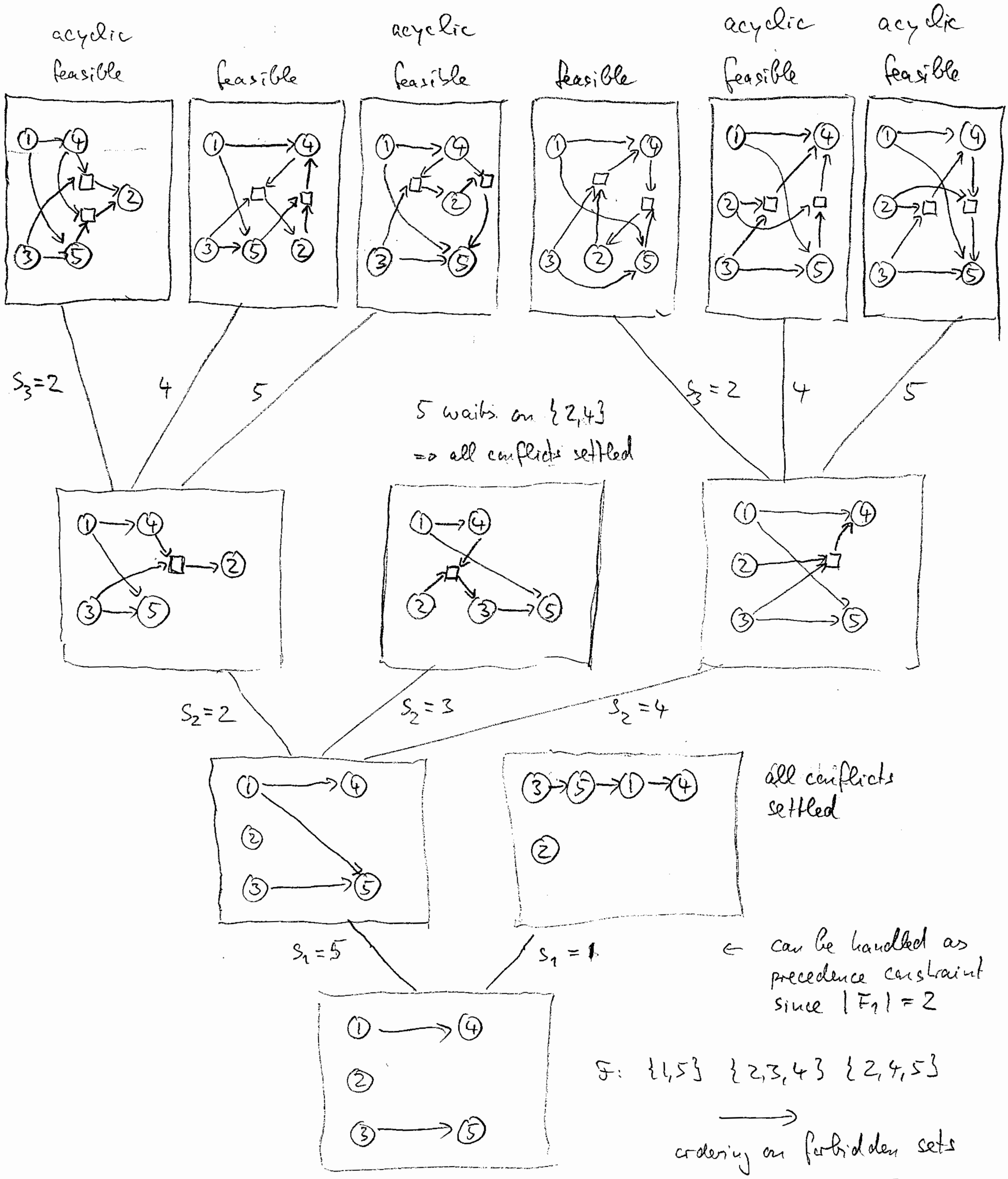
children of a node D = all AND-OR networks obtained from D by choosing a waiting job from one yet unsettled forbidden set

need to check this

may use suitable ordering of forbidden sets

9.1 EXAMPLE continued

conflict settling tree



$\mathcal{F}$: {1,5} {2,3,4} {2,4,5}

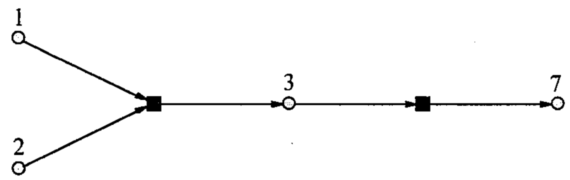
⟶

ordering on forbidden sets

F_1 F_2 F_3

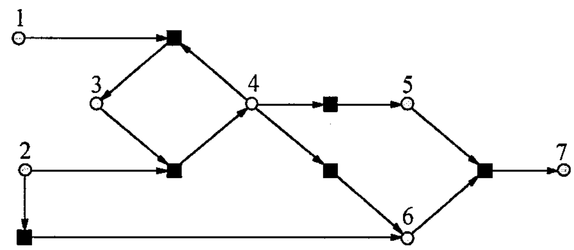
Checking if a forbidden set F is already settled by the partial selection $(s_1, \dots, s_r)$ corresponds to finding forced waiting conditions of the form $(F \setminus \{j\}, j)$

Finding forced waiting conditions



easy to answer here

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



not so easy in this case

Def: j forced to wait for U :⟺ all linear realizers have some $u \in U$ before j

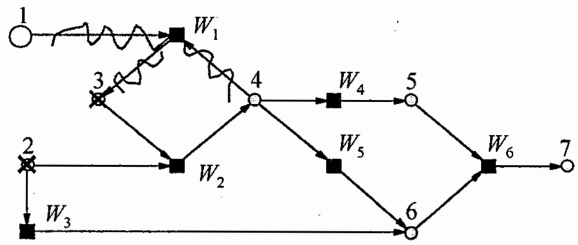
An algorithm for finding forced waiting conditions

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

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

delete all such waiting conditions

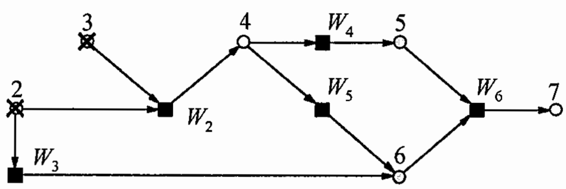
First example



$U = \{2, 3\}$

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

First example

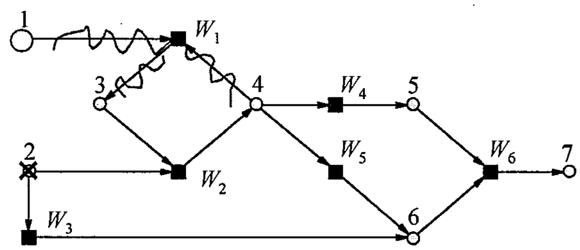


$U = \{2, 3\}$

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

Claim: Every job not in $L \cup U$ waits for $U = \{2, 3\}$

Second example

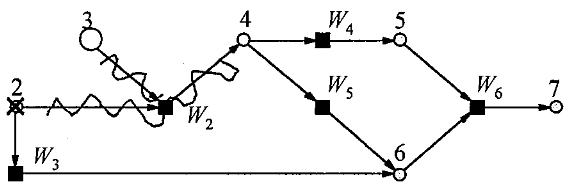


$U = \{2\}$

$L = [\ 1, \qquad]$

Second example

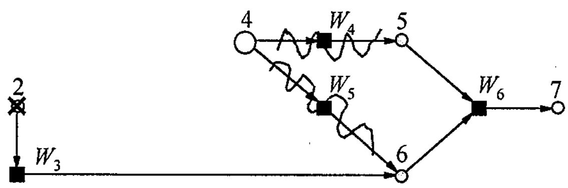
$U = \{2\}$



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

Second example

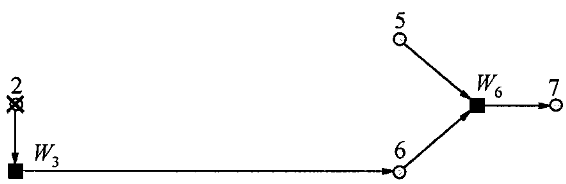
$U = \{2\}$



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

Second example

$U = \{2\}$



$L = [\,1, 3, 4, \qquad\qquad]$ $\Rightarrow$ termination with $L = [\,1, 3, 4, 5, 7\,]$

Claim: 6 waits for $U = \{2\}$

Correctness of the algorithm

(U,j) is a forced waiting condition $\Leftrightarrow j \notin L$ (and $\notin U$) 10.2 THEOREM

Key lemma: 10.3 LEMMA

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

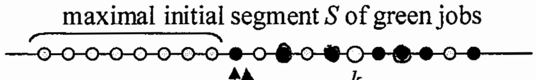
$\Rightarrow$ only U and forced waiting jobs are not in L at termination of the algorithm

Proof of key lemma (by contradiction)

green red blue

● ○ • ~ job is / is ~~red~~ ~~green~~ not in a forced waiting condition with U / is in U blue jobs

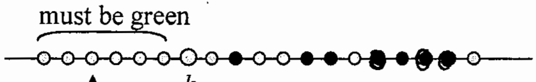
maximal initial segment S of green jobs

R'  R' constructed by algo

must be blue k

k green $\overset{def}{\Rightarrow}$ there is a linear realization R with U after k

must be green

R 

k

j first green job not in S $\Rightarrow$ cannot occur here

$\Rightarrow$ there is a waiting condition (X,j) with X is after S in R' $\Rightarrow$ $X \cap S = \emptyset$

$\Rightarrow$ there is some $i \in X$ before j in R

but all jobs before j in R are in $S \Rightarrow$ contradiction

Summary: Can detect forced waiting conditions in linear time

Computing earliest start times for a preselective policy

Input: Preselective policy Π, processing time vector x

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

= Earliest start w.r.t. system of waiting conditions given by selection s defining Π and graph G of precedence constraints

$\swarrow$

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

$\Downarrow$ §9

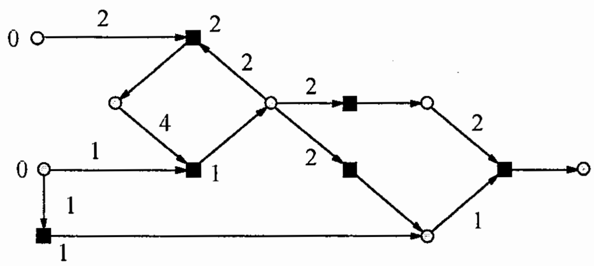
Solve a system of min-max inequalities and compute the unique componentwise minimal solution

special case needed here
have positive arc weights
$d_{jw} := x_j$ for arcs $(j) \to \boxed{w}$
may assume arc weight
$d_{wj} = 0$ for arcs $\boxed{w} \to (j)$
since only one outgoing arc
from every OR-node

general case:
arbitrary arc weights
d_{jw}, d_{wj}
(also negative)

Solve the special case by a Dijkstra like algorithm:

A Dijksta-like algorithm for positive arc weights



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

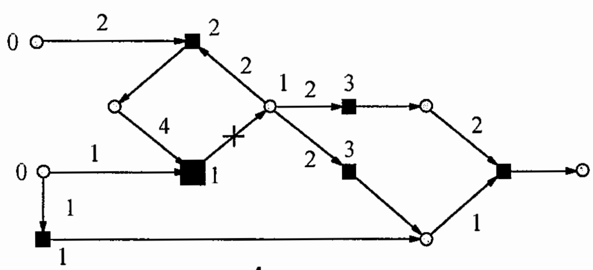
set $S_j := 0$ if there is no $(w, j) \in A$

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

set $S_w = \infty$ otherwise

10.4 ALGORITHM

A Dijksta-like algorithm for positive arc weights



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

reduce indegree of j by 1

if indegree(j) = 0 then

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

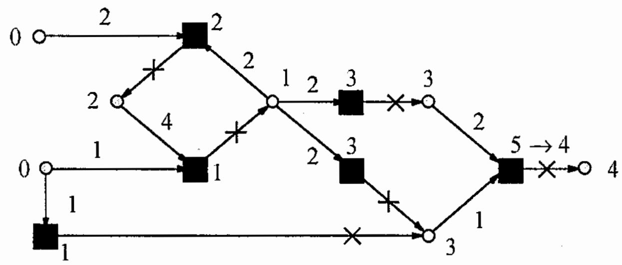
for unmarked OR nodes $w \in \text{out}(j)$, set $S_w = \min\{S_j + d_{jw} \mid (j, w) \in A\}$

repeat

big ■ ≙ marked

—×→ ≙ reducing indegree

A Dijksta-like algorithm for positive arc weights



choose OR-node $w = (X, j)$ with minimum S_w and mark w
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 $S_w = \min\{S_j + d_{jw} \mid (j, w) \in A\}$

repeat

$5 \to 4 \hat{=}$ change of distance at OR-node

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, W)
Assume that $S \neq S^*$.
Then chose 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 $S_w = S_v + \overbrace{d_{wv}}^{0}$

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

$\Rightarrow$ (choice of v) $S_v^* = S_w^*$ 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 $S_v^* = S_i^* + \underbrace{d_{iv}}_{>0}$ (*)

$$i \xrightarrow{d_{iv}} \boxed{v}$$

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

(if unmarked), or $S_v \leq S_i + d_{iv} = S_i^* + d_{iv} = S_v^*$

(if already marked)

$\Rightarrow$ 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 □

For a min-max system, $(\infty, \infty, \dots, \infty)$ is a solution.

We call a solution S <u>feasible</u>, if every $S_j < \infty$, and the min-max system <u>feasible</u> if there is a feasible solution

10.6 LEMMA: A min-max system with $x_{jw} > 0$ and $x_{wj} \geq 0$ has a feasible solution S iff the weighting conditions are feasible (feasibility of min-max system = structural feasibility)

Proof: "$\Rightarrow$" all $x_{jw} > 0 \Rightarrow$ for every $w = (X, j)$ there is an $i \in X$ with $S_i < S_w \leq S_j$

(otherwise S is not feasible)

The arcs (i,j) define a realization of W.

If not, they contain a cycle, and we would have $S_\ell < S_k$ for every arc (ℓ, k) on the cycle, which cannot be the case

$\Leftarrow$ W feasible $\Rightarrow \exists$ realisation R. ES_R defines a feasible solution of the min-max system □

Remark: For $d_{jw} \geq 0$, $d_{wj} \geq 0$, structural feasibility is not equivalent to feasibility of the min-max system

Still possible to decide feasibility in polynomial time (Exercise)

The general case: arbitrary d_{jw} and d_{wj}

Some first observations

$j \in V$ AND node: $S_j \geq \max\limits_{(w,j)\in A} \left[S_w + d_{wj}\right]$

$w \in \mathcal{W}$ OR node: $S_w \geq \min\limits_{(j,w)\in A} \left[S_j + d_{jw}\right]$

min-max system

$(\infty, \infty, \ldots, \infty)$ is a solution

$(S_1, \ldots, S_n)$ and $(T_1, \ldots, T_n)$ solutions $\Rightarrow$ $(\min\{S_1, T_1\}, \ldots, \min\{S_n, T_n\})$ solution

- There is a unique componentwise minimal solution $S \geq 0$
- Problem is feasible iff every $S_j < \infty$
- There is a unique maximal feasible subset of jobs

Certificate for SOLVABILITY $\in$ NP

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

Certificate for SOLVABILITY $\in$ coNP

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

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

Then every cycle has non-negative length

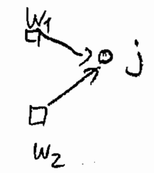
needs small proofs

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

Proof: relaxing

delete all possible s.t. S^* is not changed

Suppose $\exists j$ with indegree > 1



~~if~~ let S^1 be the best schedule after deleting (w_1, j)

S^2 ... (w_2, j)

$\Rightarrow \quad S_j^* > S_j^1 \qquad S_j^* > S_j^2 \qquad$ and w.l.o.g. $S_j^1 \le S_j^2$

Set $\quad S_i := \min\{ S_i^1 + S_i^2 - S_j^1 ,\ S_i^2 \}$

$\forall$ nodes (AND, OR) show that S is a schedule

$\Rightarrow \quad S \le S_2 \ , \quad S_j = S_j^2$

Look at $\quad \boxed{w} \xrightarrow{0} (j) \qquad w \ne w_2$

$\Rightarrow \quad S_j^2 \ge S_w \quad \Rightarrow \quad S_j = S_j^2 \ge S_w$

$\boxed{w_2} \to j \quad \Rightarrow \quad S_j = \underbrace{S_j^1}_{\ge S_{w_2}} + \underbrace{S_j^2 - S_j^1}_{\ge 0} \ge S_{w_2}$

↑ Def S

$\Rightarrow S_j \ge \max_{(w,j)} S_w$

For other AND nodes $(\ne j)$

$\boxed{w} \to (k) \qquad \left.\begin{array}{l} S_k^1 \ge S_w^1 \\ S_k^2 \ge S_w^2 \end{array}\right\} \Rightarrow S_k^1 + S_k^2 - \overbrace{S_j^1}^{\text{konstant}} \ge S_w^1 + S_w^2 - S_j^1$

Similar for OR nodes

$\Rightarrow \quad S = \min\{ S^1 + S^2 - S_j^1 ,\ S^2 \}$ is a schedule

contradiction to $\quad S_j = S_j^2 < S_j^*$

↑ best schedule by construction

Complexity of checking solvability

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

⇓

SOLVABILITY ∈ NP ∩ coNP

TO.7 THEOREM

No polynomial algorithm known

Not known to be NP-complete / coNP complete

Same complexity status as

LINEAR PROGRAMMING → ∈ P by Ellipsoid method

PRIMES → solved 2002, ∈ P
AKS primality test
Agrawal - Kayal - Saxena

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