Inverse problems on the blocksequential operator in Boolean networks

Julio Aracena†, **Luis Cabrera-Crot***,
Adrien Richard° and Lilian Salinas+

∗ PhD Student in Computer Science, U. of Concepción, Chile.[1]
† Department of Mathematical Engineering, U. of Concepción, Chile.
+ Department of Computer Science, U. of Concepción, Chile.
◦ CRNS and Université Côte dÁzur, France

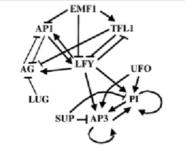
International Workshop on Boolean Networks
January 9th, 2020

[1] Funded by CONICYT-PFCHA/Doctorado Nacional/2016-21160885

Contents

Motivation

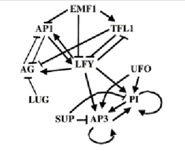
- Complex Systems



(L. Mendoza and E. Alvarez, 1998)

Motivation

- Complex Systems
- Boolean networks $f : \{0,1\}^n \to \{0,1\}^n$

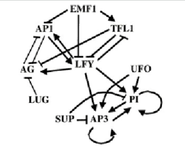


(L. Mendoza and E. Alvarez, 1998)

$$
\begin{aligned}
f_1(x) &= x_4 \\
f_2(x) &= x_1 \wedge x_2 \\
f_3(x) &= x_2 \vee x_3 \\
f_4(x) &= x_3 \wedge x_4
\end{aligned}
$$

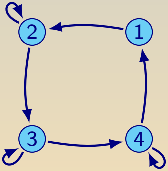
Motivation

- Complex Systems
- Boolean networks $f : \{0,1\}^n \to \{0,1\}^n$
  - Interaction Graph



(L. Mendoza and E. Alvarez, 1998)

$$
\begin{aligned}
f_1(x) &= x_4 \\
f_2(x) &= x_1 \wedge x_2 \\
f_3(x) &= x_2 \vee x_3 \\
f_4(x) &= x_3 \wedge x_4
\end{aligned}
$$



Block-sequential schedule

Definition

A *block-sequential schedule* is an ordered partition of the components of a Boolean network which defines the order in which the states of the network are updated in one unit of time.

Examples

$$s_1 = \{3,4\}\{1\}\{2\},$$

$$s_2 = \{1,2,3,4\},$$

$$s_3 = \{2\}\{3\}\{4\}\{1\}.$$

Labeled digraph

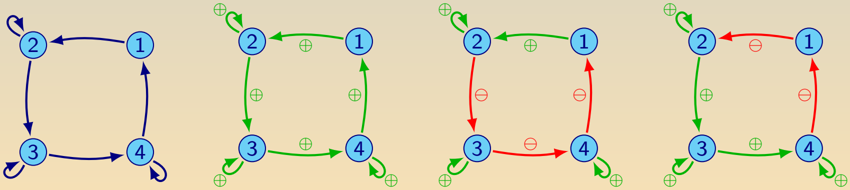
Given a interaction graph G and a block-sequential schedule s, a labeled digraph (G, s) is a digraph with a labeling function lab_s:

$$lab_s : A(G) \to \{\oplus, \ominus\}$$

$$lab_s(u, v) = \oplus \iff s(u) \geq s(v)$$

G $\quad$ $s_1 = \{1, 2, 3, 4\}$ $\quad$ $s_2 = \{2\}\{3\}\{4\}\{1\}$ $\quad$ $s_3 = \{3, 4\}\{1\}\{2\}$

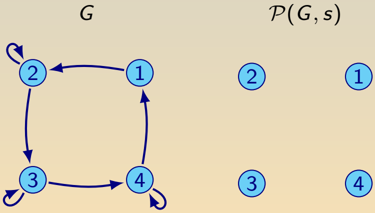
$s_4 = \{4\}\{1\}\{3, 2\}$



Parallel digraph

The *parallel digraph* is a digraph that represent the real dependence of the components of the Boolean network, according with the block-sequential schedule.
Also, is equivalent to the interaction graph of a Boolean network with equal dynamic behavior when is updated in parallel (one only block).

$s = \{2\}\{3\}\{4\}\{1\}$

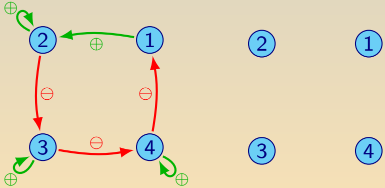


Parallel digraph

The *parallel digraph* is a digraph that represent the real dependence of the components of the Boolean network, according with the block-sequential schedule.
Also, is equivalent to the interaction graph of a Boolean network with equal dynamic behavior when is updated in parallel (one only block).

$s = \{2\}\{3\}\{4\}\{1\}$

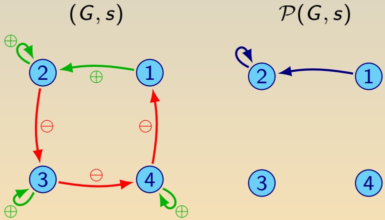
(G, s) $\mathcal{P}(G, s)$



Parallel digraph

The *parallel digraph* is a digraph that represent the real dependence of the components of the Boolean network, according with the block-sequential schedule.
Also, is equivalent to the interaction graph of a Boolean network with equal dynamic behavior when is updated in parallel (one only block).

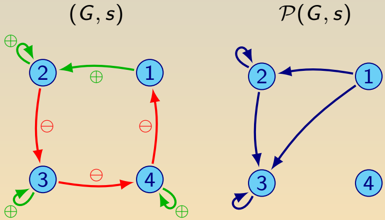
$s = \{2\}\,\{3\}\,\{4\}\,\{1\}$



Parallel digraph

The *parallel digraph* is a digraph that represent the real dependence of the components of the Boolean network, according with the block-sequential schedule.
Also, is equivalent to the interaction graph of a Boolean network with equal dynamic behavior when is updated in parallel (one only block).

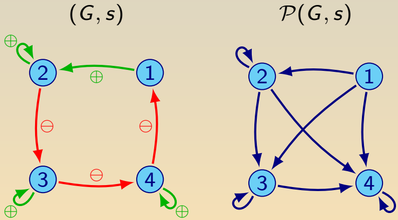
$s = \{2\}\,\{3\}\,\{4\}\,\{1\}$



Parallel digraph

The *parallel digraph* is a digraph that represent the real dependence of the components of the Boolean network, according with the block-sequential schedule.
Also, is equivalent to the interaction graph of a Boolean network with equal dynamic behavior when is updated in parallel (one only block).

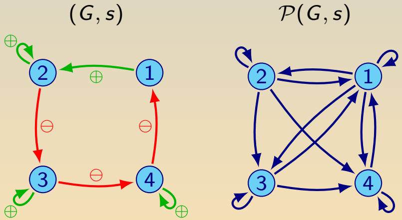
$s = \{2\}\{3\}\{4\}\{1\}$



Parallel digraph

The *parallel digraph* is a digraph that represent the real dependence of the components of the Boolean network, according with the block-sequential schedule.
Also, is equivalent to the interaction graph of a Boolean network with equal dynamic behavior when is updated in parallel (one only block).

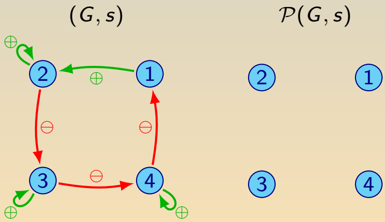
$s = \{2\}\{3\}\{4\}\{1\}$



Parallel digraph

Can be obtained from the labeled digraph.
$\forall(u,v) \in V(G) \times V(G), (u,v) \in A(\mathcal{P}(G,s))$ if and only if:

$s = \{2\}\{3\}\{4\}\{1\}$

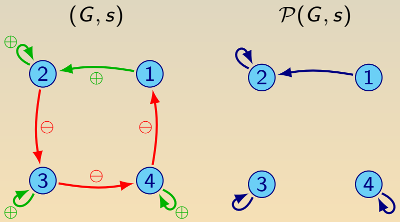


Parallel digraph

Can be obtained from the labeled digraph.
$\forall (u,v) \in V(G) \times V(G), (u,v) \in A(\mathcal{P}(G,s))$ if and only if:

- (u,v) is labeled $\oplus$.

$s = \{2\}\{3\}\{4\}\{1\}$

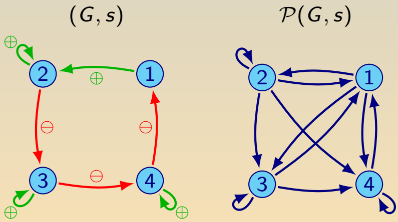


Parallel digraph

Can be obtained from the labeled digraph.
$\forall (u,v) \in V(G) \times V(G), (u,v) \in A(\mathcal{P}(G,s))$ if and only if:

- (u,v) is labeled $\oplus$.
- $\exists w \in V(G), (u,w)$ is labeled $\oplus$ and exists a path from w to v labeled $\ominus$.

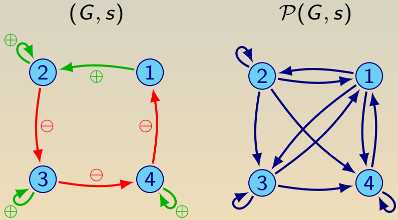
$s = \{2\}\{3\}\{4\}\{1\}$



Parallel digraph

The function that associates (G, s) with the digraph $\mathcal{P}(G, s)$ is called **block-sequential operator** and can be constructed in polynomial time.

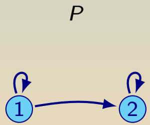
$s = \{2\}\{3\}\{4\}\{1\}$



Inverse problem

Given a digraph P and a block-sequential schedule s, does there exist a digraph G such that $\mathcal{P}(G, s) = P$?

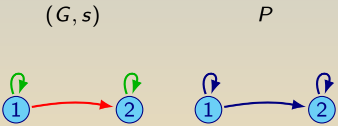
$$s = \{1\}\{2\}$$



Inverse problem

Given a digraph P and a block-sequential schedule s, does there exist a digraph G such that $\mathcal{P}(G, s) = P$?

$s = \{1\}\{2\}$



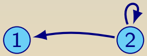
Unique solution

Inverse problem

Given a digraph P and a block-sequential schedule s, does there exist a digraph G such that $\mathcal{P}(G, s) = P$?

$$s = \{1\}\{2\}$$

P

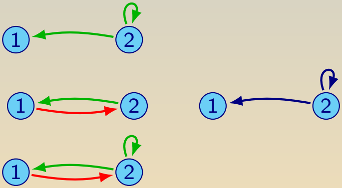


Inverse problem

Given a digraph P and a block-sequential schedule s, does there exist a digraph G such that $\mathcal{P}(G,s) = P$?

$s = \{1\}\{2\}$

(G,s) $\quad$ P

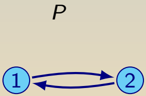


Multiple solutions

Inverse problem

Given a digraph P and a block-sequential schedule s, does there exist a digraph G such that $\mathcal{P}(G, s) = P$?

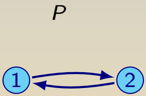
$s = \{1\}\{2\}$



Inverse problem

Given a digraph P and a block-sequential schedule s, does there exist a digraph G such that $\mathcal{P}(G, s) = P$?

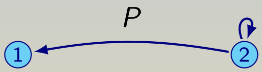
$s = \{1\}\{2\}$



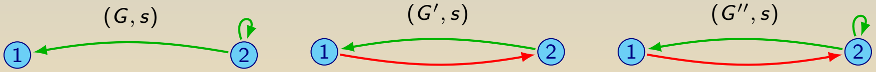
No solution

Contents

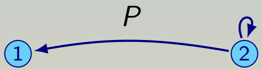
Results



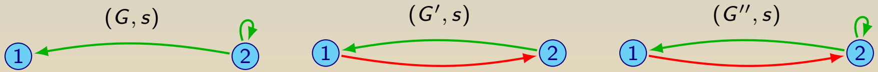
For example, for the schedule $\{1\}\,\{2\}$, there are three preimages:



Results



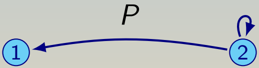
For example, for the schedule $\{1\}\{2\}$, there are three preimages:



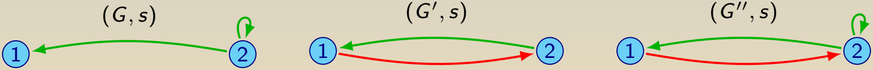
Theorem

Let s be a block-sequential schedule and G and G' two digraphs such that $\mathcal{P}(G,s) = \mathcal{P}(G',s)$. Then $\mathcal{P}(G \cup G', s) = \mathcal{P}(G,s)$.

Results



For example, for the schedule $\{1\}\{2\}$, there are three preimages:



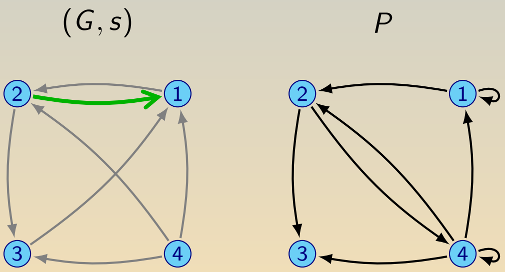
Theorem

For this reason, if for a digraph P and a block-sequential schedule s there exists at least one preimage G, then there exists a maximum preimage.

Rules

$\overline{P}$ Rule

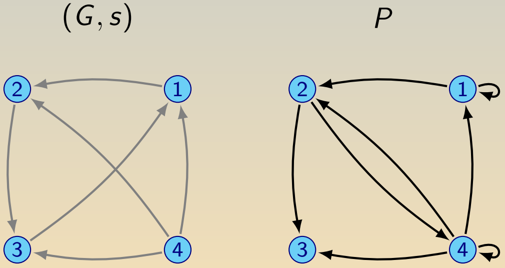
$\forall (u, v)$ such that $lab(u, v) = \oplus$, if $(u, v) \notin P$, then $(u, v) \notin G$.



Rules

$\overline{P}$ Rule

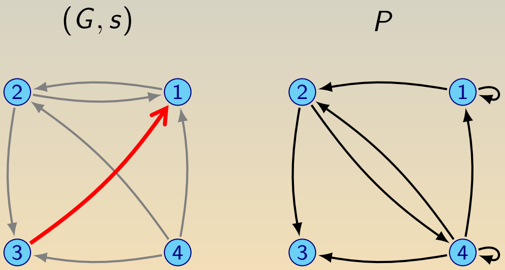
$\forall(u,v)$ such that $lab(u,v) = \oplus$, if $(u,v) \notin P$, then $(u,v) \notin G$.



Rules

Transitive Rule

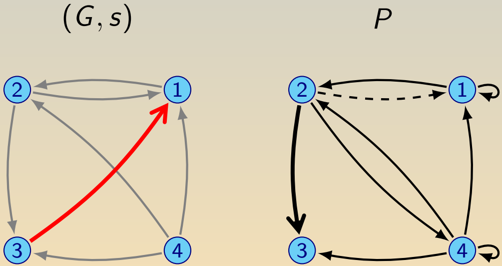
$\forall(u,v)$ such that $lab(u,v) = \ominus$, if $\exists w$ such that $(w,u) \in P$ and $(w,v) \notin P$, then $(u,v) \notin G$.



Rules

Transitive Rule

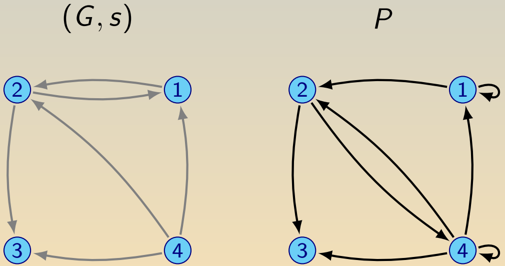
$\forall(u, v)$ such that $lab(u, v) = \ominus$, if $\exists w$ such that $(w, u) \in P$ and $(w, v) \notin P$, then $(u, v) \notin G$.



Rules

Transitive Rule

$\forall(u,v)$ such that $lab(u,v) = \ominus$, if $\exists w$ such that $(w,u) \in P$ and $(w,v) \notin P$, then $(u,v) \notin G$.

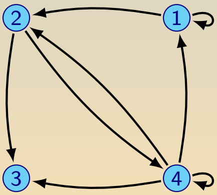


Algorithm MaxPI: Step 1 - Build and label

Input

Given a digraph P and a block-sequential schedule $s = \{3\}\{1\}\{2,4\}$.

P



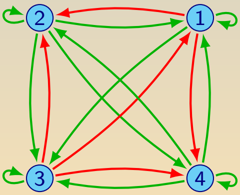
Algorithm MaxPI: Step 1 - Build and label

Input

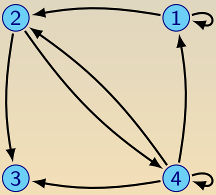
Given a digraph P and a block-sequential schedule $s = \{3\}\{1\}\{2,4\}$.

Initially: $G \leftarrow K_n,\ n = |V(P)|$.

(G, s)



P

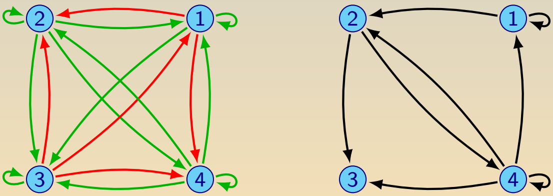


Algorithm MaxPI: Step 2a
Removing green arcs

Rule

$\forall (u, v) \in A(G)$ that does not satisfy the "$\overline{P}$ rule",
(u, v) is removed from G.

(G, s) P

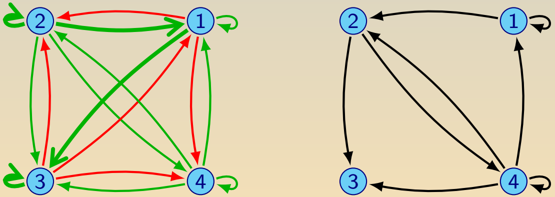


Algorithm MaxPI: Step 2a
Removing green arcs

Rule

$\forall (u, v) \in A(G)$ that does not satisfy the "$\overline{P}$ rule",
(u, v) is removed from G.

(G, s) P

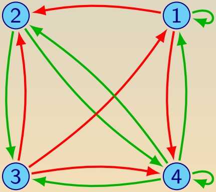


Algorithm MaxPI: Step 2a
Removing green arcs

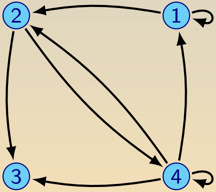
Rule

$\forall (u, v) \in A(G)$ that does not satisfy the "$\overline{P}$ rule",
(u, v) is removed from G.

(G, s)



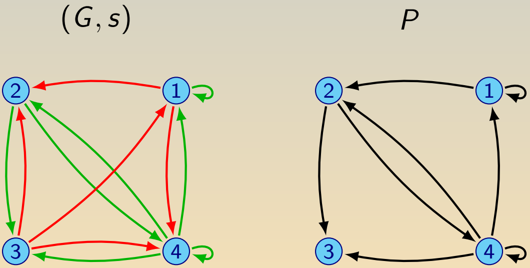
P



Algorithm MaxPI: Step 2b
Removing red arcs

Rule

$\forall(u,v) \in A(G)$ that does not satisfy the "Transitive rule", (u,v) is removed from G.

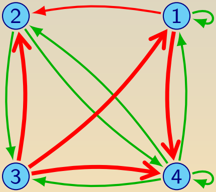


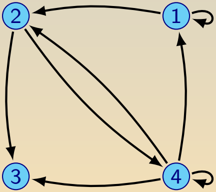
Algorithm MaxPI: Step 2b
Removing red arcs

Rule

$\forall(u, v) \in A(G)$ that does not satisfy the "Transitive rule",
(u, v) is removed from G.

(G, s) P



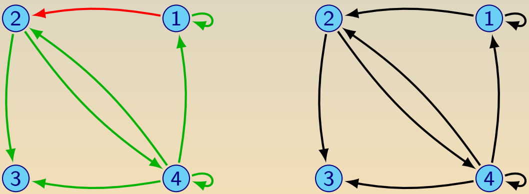


Algorithm MaxPI: Step 2b Removing red arcs

Rule

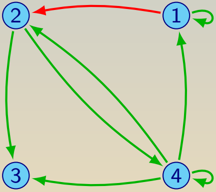
$\forall (u, v) \in A(G)$ that does not satisfy the "Transitive rule", (u, v) is removed from G.

(G, s) $\quad\quad P$

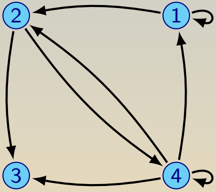


Algorithm MaxPI: Step 3 - Validation

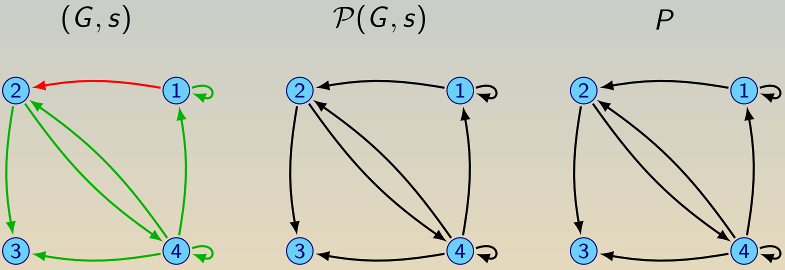
(G, s)



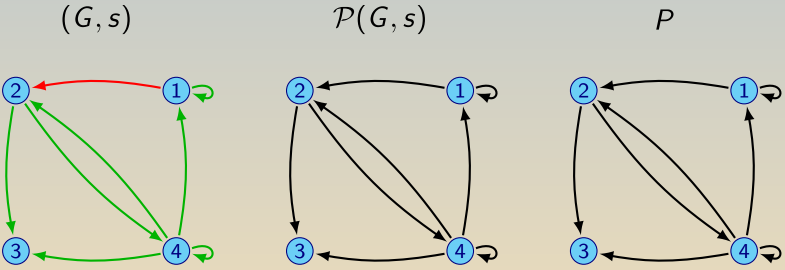
P



Algorithm MaxPI: Step 3 - Validation



Algorithm MaxPI: Step 3 - Validation



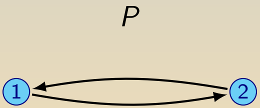
Output

Since $\mathcal{P}(G, s) = P$, the algorithm return (G, s) as maximun preimage of P with the schedule s.

Another example of MaxPI: Step 1

Input

Given a digraph P and a block-sequential schedule $s = \{1\}\{2\}$.

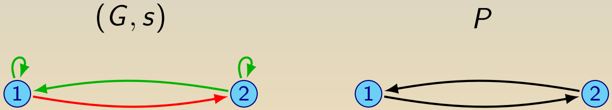


Another example of MaxPI: Step 1

Input

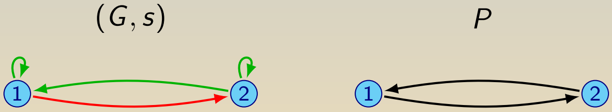
Given a digraph P and a block-sequential schedule $s = \{1\}\{2\}$.

Initially: $G \leftarrow K_n,\ \ n = |V(P)|$.



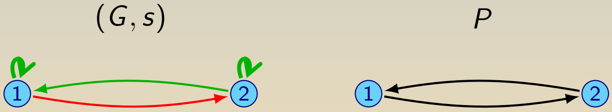
Another example of MaxPI: Step 2a

Removing green arcs, according "$\overline{P}$ rule".



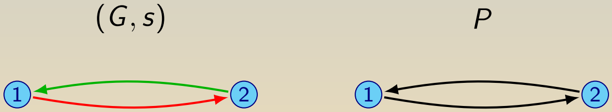
Another example of MaxPI: Step 2a

Removing green arcs, according "$\overline{P}$ rule".



Another example of MaxPI: Step 2a

Removing green arcs, according "$\overline{P}$ rule".



Another example of MaxPI: Step 2b

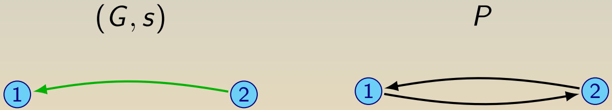
Removing red arcs, according "Transitive rule".

Another example of MaxPI: Step 2b

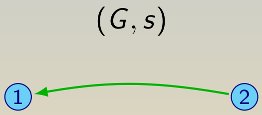
Removing red arcs, according "Transitive rule".

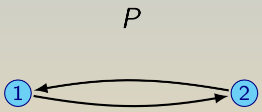
Another example of MaxPI: Step 2b

Removing red arcs, according "Transitive rule".



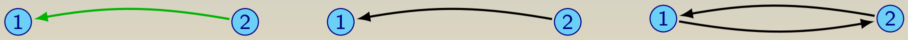
Another example of MaxPI: Step 3



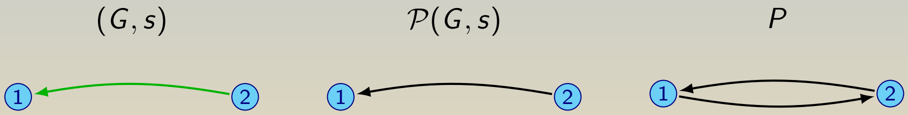


Another example of MaxPI: Step 3

(G, s) $\mathcal{P}(G, s)$ P



Another example of MaxPI: Step 3

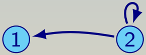


Output

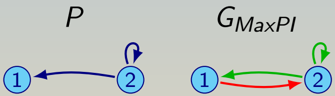
Since $\mathcal{P}(G,s) \neq P$, P does not have preimage for the schedule s.

Algorithm for enumeration of preimages

P $\qquad$ G_{MaxPI} $\qquad$ $s = \{1\}\{2\}$



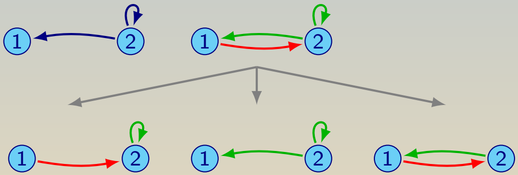
Algorithm for enumeration of preimages



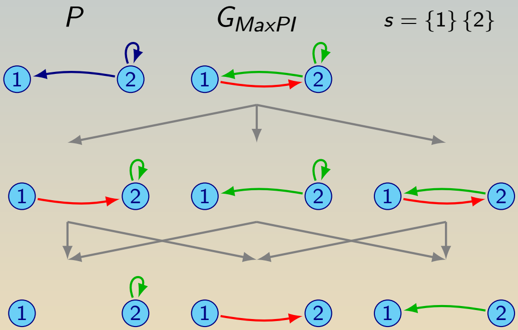
$s = \{1\}\{2\}$

Algorithm for enumeration of preimages

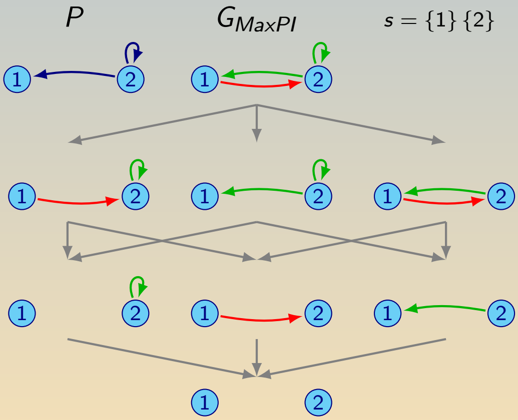
P $\quad G_{MaxPI}$ $\quad s = \{1\}\{2\}$



Algorithm for enumeration of preimages



Algorithm for enumeration of preimages



Algorithm for enumeration of preimages

Lemma

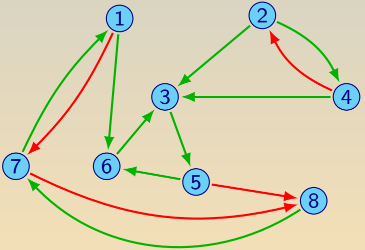
Let s be a block-sequential schedule and G and G' two digraphs such that $V(G) = V(G')$. If $G \subseteq G'$, then $\mathcal{P}(G, s) \subseteq \mathcal{P}(G', s)$.

Algorithm for enumeration of preimages

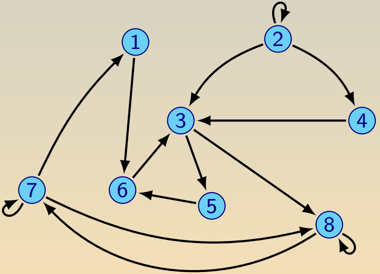
Lemma

Let s be a block-sequential schedule and G and G' two digraphs such that $V(G) = V(G')$. If $G \subseteq G'$, then $\mathcal{P}(G, s) \subseteq \mathcal{P}(G', s)$.

(G, s)



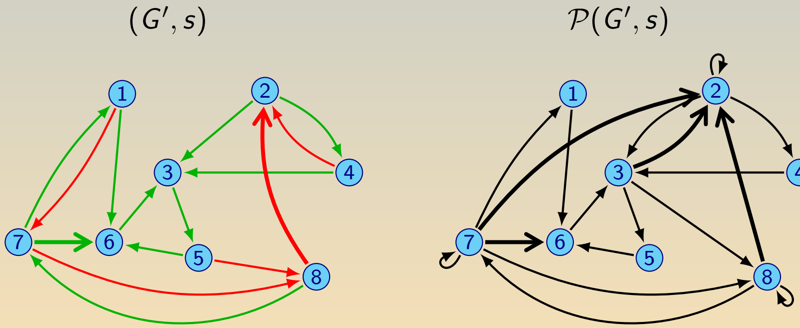
$\mathcal{P}(G, s)$



Algorithm for enumeration of preimages

Lemma

Let s be a block-sequential schedule and G and G' two digraphs such that $V(G) = V(G')$. If $G \subseteq G'$, then $\mathcal{P}(G, s) \subseteq \mathcal{P}(G', s)$.



Algorithm for enumeration of preimages

Proposition

Let s be a block-sequential schedule and G and G'' two digraphs such that $G'' \subseteq G$. If $\mathcal{P}(G, s) = \mathcal{P}(G'', s)$, then
$\forall G', G'' \subseteq G' \subseteq G, \mathcal{P}(G', s) = \mathcal{P}(G, s) = \mathcal{P}(G'', s)$.

Algorithm for enumeration of preimages

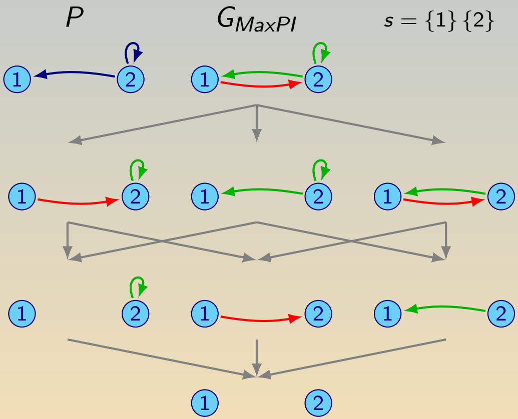
Proposition

Let s be a block-sequential schedule and G and G'' two digraphs such that $G'' \subseteq G$. If $\mathcal{P}(G, s) = \mathcal{P}(G'', s)$, then
$\forall G', G'' \subseteq G' \subseteq G, \mathcal{P}(G', s) = \mathcal{P}(G, s) = \mathcal{P}(G'', s)$.

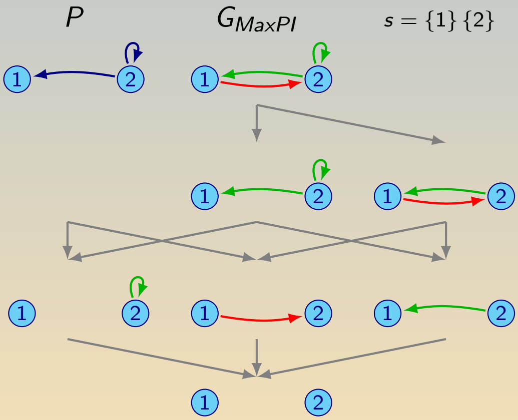
Proof

Since $G'' \subseteq G' \subseteq G$, then $\mathcal{P}(G'', s) \subseteq \mathcal{P}(G', s) \subseteq \mathcal{P}(G, s)$. Since $\mathcal{P}(G'', s) = \mathcal{P}(G, s)$, then $\mathcal{P}(G'', s) = \mathcal{P}(G', s) = \mathcal{P}(G, s)$.

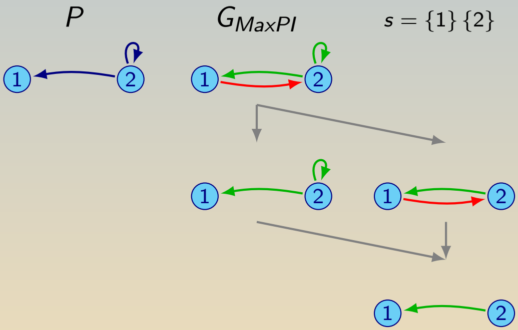
Algorithm for enumeration of preimages



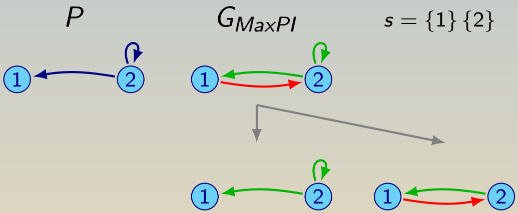
Algorithm for enumeration of preimages



Algorithm for enumeration of preimages



Algorithm for enumeration of preimages



Algorithm for enumeration of preimages

Complexity

The complexity of this algorithm has a **polynomial delay**.

Algorithm for enumeration of preimages

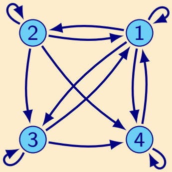
Complexity

The complexity of this algorithm has a **polynomial delay**.
Since there are cases with an exponential number of pre-images, listing all the pre-images has an **exponential cost**

Algorithm for enumeration of preimages

Complexity

The complexity of this algorithm has a **polynomial delay**.
Since there are cases with an exponential number of pre-images, listing all the pre-images has an **exponential cost**
For example, this digraph with the block-sequential schedule $\{2\}\{3\}\{4\}\{1\}$ has 8 preimages, corresponding to $2^{\frac{(n-2)(n-1)}{2}}$.

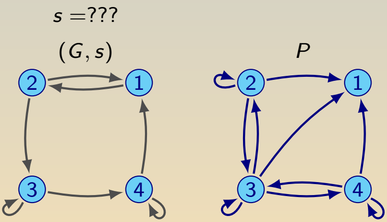


Contents

Work in progress...

Problem

Given two digraphs G and P, does there exist a block-sequential schedule s such that $\mathcal{P}(G, s) = P$?



Work in progress...

Theorem

If G and P are acyclic digraphs, it is possible to decide if there is labeling function lab such that $\mathcal{P}(G, lab) = P$ in polynomial time.

Work in progress...

Theorem

If G and P are acyclic digraphs, it is possible to decide if there is labeling function *lab* such that $\mathcal{P}(G, lab) = P$ in polynomial time.

How?

With an algorithm that label the arcs of G, according the following rules:

Work in progress...

Theorem

If G and P are acyclic digraphs, it is possible to decide if there is labeling function *lab* such that $\mathcal{P}(G, lab) = P$ in polynomial time.

How?

With an algorithm that label the arcs of G, according the following rules:

- "Transitive rule": If $\exists u, v, w \in V(G)$, such that $(u, v) \in G, (w, u) \in P$ and $(w, v) \notin P$, then $lab(u, v) = \oplus$.

Work in progress...

Theorem

If G and P are acyclic digraphs, it is possible to decide if there is labeling function *lab* such that $\mathcal{P}(G, lab) = P$ in polynomial time.

How?

With an algorithm that label the arcs of G, according the following rules:

- "Transitive rule": If $\exists u, v, w \in V(G)$, such that $(u, v) \in G, (w, u) \in P$ and $(w, v) \notin P$, then $lab(u, v) = \oplus$.
- "$\overline{P}$ rule": If $\exists u, v \in V(G)$, such that $(u, v) \in G$ and $(u, v) \notin P$, then $lab(u, v) = \ominus$.

Work in progress...

Theorem

If G and P are acyclic digraphs, it is possible to decide if there is labeling function lab such that $\mathcal{P}(G, lab) = P$ in polynomial time.

How?

With an algorithm that label the arcs of G, according the following rules:

- "Transitive rule": If $\exists u, v, w \in V(G)$, such that $(u, v) \in G, (w, u) \in P$ and $(w, v) \notin P$, then $lab(u, v) = \oplus$.
- "$\overline{P}$ rule": If $\exists u, v \in V(G)$, such that $(u, v) \in G$ and $(u, v) \notin P$, then $lab(u, v) = \ominus$.
- If $\exists u, v \in V(G)$, such that if (u, v) is labeled $\ominus$, then an arc that is not in P is formed, then $lab(u, v) = \oplus$.

Work in progress...

Theorem

If G and P are acyclic digraphs, it is possible to decide if there is labeling function lab such that $\mathcal{P}(G, lab) = P$ in polynomial time.

How?

With an algorithm that label the arcs of G, according the following rules:

- "Transitive rule": If $\exists u, v, w \in V(G)$, such that $(u, v) \in G, (w, u) \in P$ and $(w, v) \notin P$, then $lab(u, v) = \oplus$.
- "$\overline{P}$ rule": If $\exists u, v \in V(G)$, such that $(u, v) \in G$ and $(u, v) \notin P$, then $lab(u, v) = \ominus$.
- If $\exists u, v \in V(G)$, such that if (u, v) is labeled $\ominus$, then an arc that is not in P is formed, then $lab(u, v) = \oplus$.
- If $\exists u, v \in V(G)$, such that if (u, v) is labeled $\oplus$, then an arc that is in P cannot be formed, then $lab(u, v) = \ominus$.

Work in progress...

Theorem

If G and P are acyclic digraphs, it is possible to decide if there is labeling function *lab* such that $\mathcal{P}(G, lab) = P$ in polynomial time.

How?

With an algorithm that label the arcs of G, according the following rules:

- "Transitive rule": If $\exists u, v, w \in V(G)$, such that $(u, v) \in G, (w, u) \in P$ and $(w, v) \notin P$, then $lab(u, v) = \oplus$.
- "$\overline{P}$ rule": If $\exists u, v \in V(G)$, such that $(u, v) \in G$ and $(u, v) \notin P$, then $lab(u, v) = \ominus$.
- If $\exists u, v \in V(G)$, such that if (u, v) is labeled $\ominus$, then an arc that is not in P is formed, then $lab(u, v) = \oplus$.
- If $\exists u, v \in V(G)$, such that if (u, v) is labeled $\oplus$, then an arc that is in P cannot be formed, then $lab(u, v) = \ominus$.
- If $\exists u, v \in V(G)$, such that if (u, v) is labeled $\ominus$, then an arc that is in P cannot be formed, then $lab(u, v) = \oplus$.

Work in progress...

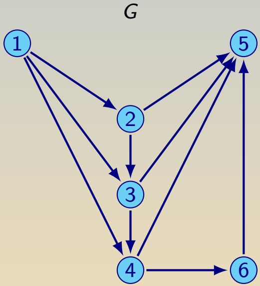
If one arc is labeled $\oplus$ and $\ominus$ by different rules, then the decision problem answer is "There is no labeling function *lab* such that $\mathcal{P}(G, lab) = P$"..

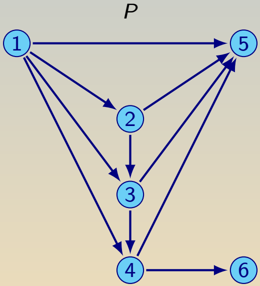
Work in progress...

If one arc is labeled $\oplus$ and $\ominus$ by different rules, then the decision problem answer is "There is no labeling function *lab* such that $\mathcal{P}(G, lab) = P$"..

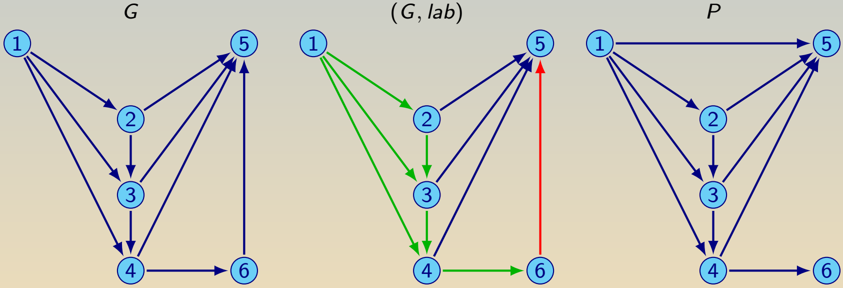
Otherwise, the labeling function formed by all the arcs labeled by the algorithm plus negative arcs (replacing the arcs not labeled by the algorithm) is a labeling function such that $\mathcal{P}(G, lab) = P$.

Work in progress...

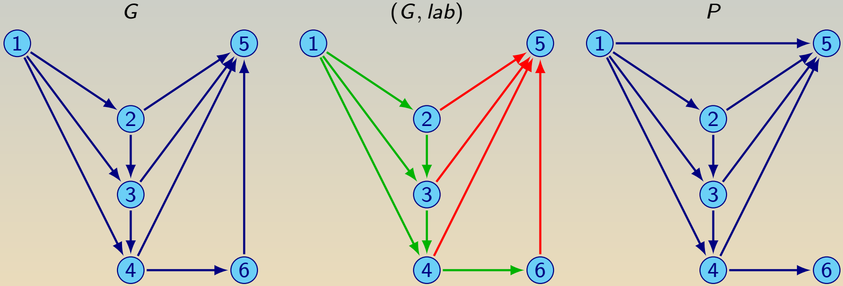




Work in progress...



Work in progress...



Work in progress...

- And for the acyclic case?

Work in progress...

- And for the acyclic case?

Still in progress

Work in progress...

- And for the acyclic case?

Still in progress

Thank You!