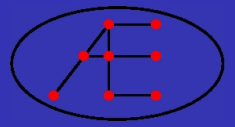
How to Attack the NP-Complete Dag Realization Problem in Practice

Annabell Berger and Matthias Müller-Hannemann

Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg
http://www.informatik.uni-halle.de

June 7, 2012

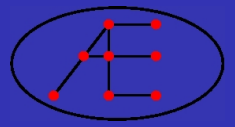
The Dag Realization Problem



Problem (dag realization problem)

Given is a finite sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ *with* $a_i, b_i \in \mathbb{Z}_0^+$. *Does there exist a dag (acyclic digraph without parallel arcs)* $G = (V, A)$ *with the labeled vertex set* $V := \{v_1, \ldots, v_n\}$ *such that we have indegree* $d_G^-(v_i) = a_i$ *and outdegree* $d_G^+(v_i) = b_i$ *for all* $v_i \in V$*?*

The Dag Realization Problem



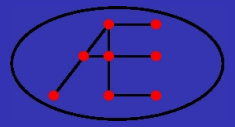
Problem (dag realization problem)

Given is a finite sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ *with* $a_i, b_i \in \mathbb{Z}_0^+$. *Does there exist a dag (acyclic digraph without parallel arcs)* $G = (V, A)$ *with the labeled vertex set* $V := \{v_1, \ldots, v_n\}$ *such that we have indegree* $d_G^-(v_i) = a_i$ *and outdegree* $d_G^+(v_i) = b_i$ *for all* $v_i \in V$*?*

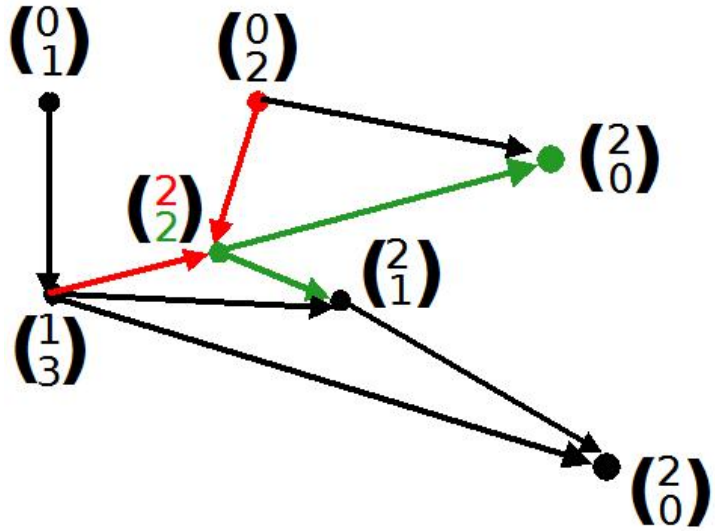
In case the answer is "yes" we call

- sequence S dag sequence
- dag G a dag realization

The Dag Realization Problem – an Example

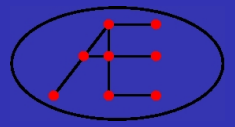


- Given is a sequence
  $\binom{0}{2}, \binom{0}{1}, \binom{1}{3}, \binom{2}{2}, \binom{2}{1}, \binom{2}{0}, \binom{2}{0}$.



Find an acyclic digraph with corresponding vertex degrees.

Terminology



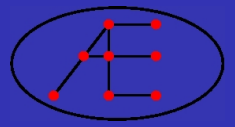
Classification of tuples $\binom{a_i}{b_i}$

- source tuple: $a_i = 0$ and $b_i > 0$
- sink tuple: $a_i > 0$ and $b_i = 0$
- stream tuple: $a_i > 0$ and $b_i > 0$

Assumptions:

- no *zero tuples* $\binom{0}{0}$
- $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} b_i$ (necessary for realization)

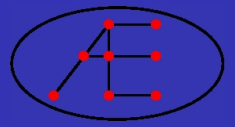
Digraph Realization is Easy



Problem (digraph realization problem)

Given is a finite sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ *with* $a_i, b_i \in \mathbb{Z}_0^+$. *Does there exist a digraph (without parallel arcs)* $G = (V, A)$ *with the labeled vertex set* $V := \{v_1, \ldots, v_n\}$ *such that we have indegree* $d_G^-(v_i) = a_i$ *and outdegree* $d_G^+(v_i) = b_i$ *for all* $v_i \in V$*?*

Digraph Realization is Easy



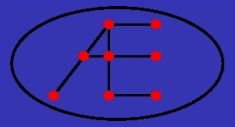
Problem (digraph realization problem)

Given is a finite sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ *with* $a_i, b_i \in \mathbb{Z}_0^+$. *Does there exist a digraph (without parallel arcs)* $G = (V, A)$ *with the labeled vertex set* $V := \{v_1, \ldots, v_n\}$ *such that we have indegree* $d_G^-(v_i) = a_i$ *and outdegree* $d_G^+(v_i) = b_i$ *for all* $v_i \in V$*?*

Two different approaches with polynomial running time:

1. recursive algorithms (KLEITMAN, WANG 1973) — choose an arbitrary tuple $\binom{a_i}{b_i}$ and reduce from b_i lexicographical largest tuples the a_j by "one"

Digraph Realization is Easy



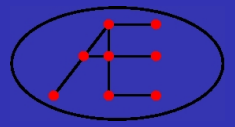
Problem (digraph realization problem)

Given is a finite sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ *with* $a_i, b_i \in \mathbb{Z}_0^+$. *Does there exist a digraph (without parallel arcs)* $G = (V, A)$ *with the labeled vertex set* $V := \{v_1, \ldots, v_n\}$ *such that we have indegree* $d_G^-(v_i) = a_i$ *and outdegree* $d_G^+(v_i) = b_i$ *for all* $v_i \in V$*?*

Two different approaches with polynomial running time:

1. recursive algorithms (KLEITMAN, WANG 1973) — choose an arbitrary tuple $\binom{a_i}{b_i}$ and reduce from b_i lexicographical largest tuples the a_j by "one"
2. complete characterization of digraph sequences (GALE 1957, RYSER 1957, FULKERSON 1960, CHEN 1966) — check a polynomial number of inequalities (in the size n)

Complexity of Dag Realization

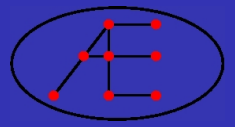


Theorem (Nichterlein 2011)

The dag realization problem is (strongly) NP-complete.

Proof: by reduction from 3-PARTITION

Complexity of Dag Realization



Theorem (Nichterlein 2011)

The dag realization problem is (strongly) NP-complete.

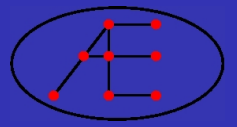
Proof: by reduction from 3-PARTITION

Theorem (Hartung and Nichterlein 2012)

The dag realization problem is fixed parameter tractable with respect to the parameter maximum degree Δ.

Note: This is a mere classification result. The running time of their FPT algorithm is $\Delta^{\Delta^{\Delta^{O(\Delta)}}} \cdot n!$

Realization with a Fixed Topological Order



Realization with a prescribed topological order

Input: sequence $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$
topological order $v_1 < v_2, < \cdots < v_n$

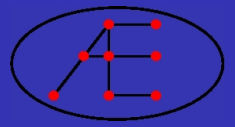
Task: Find a dag realization according to the given top. order

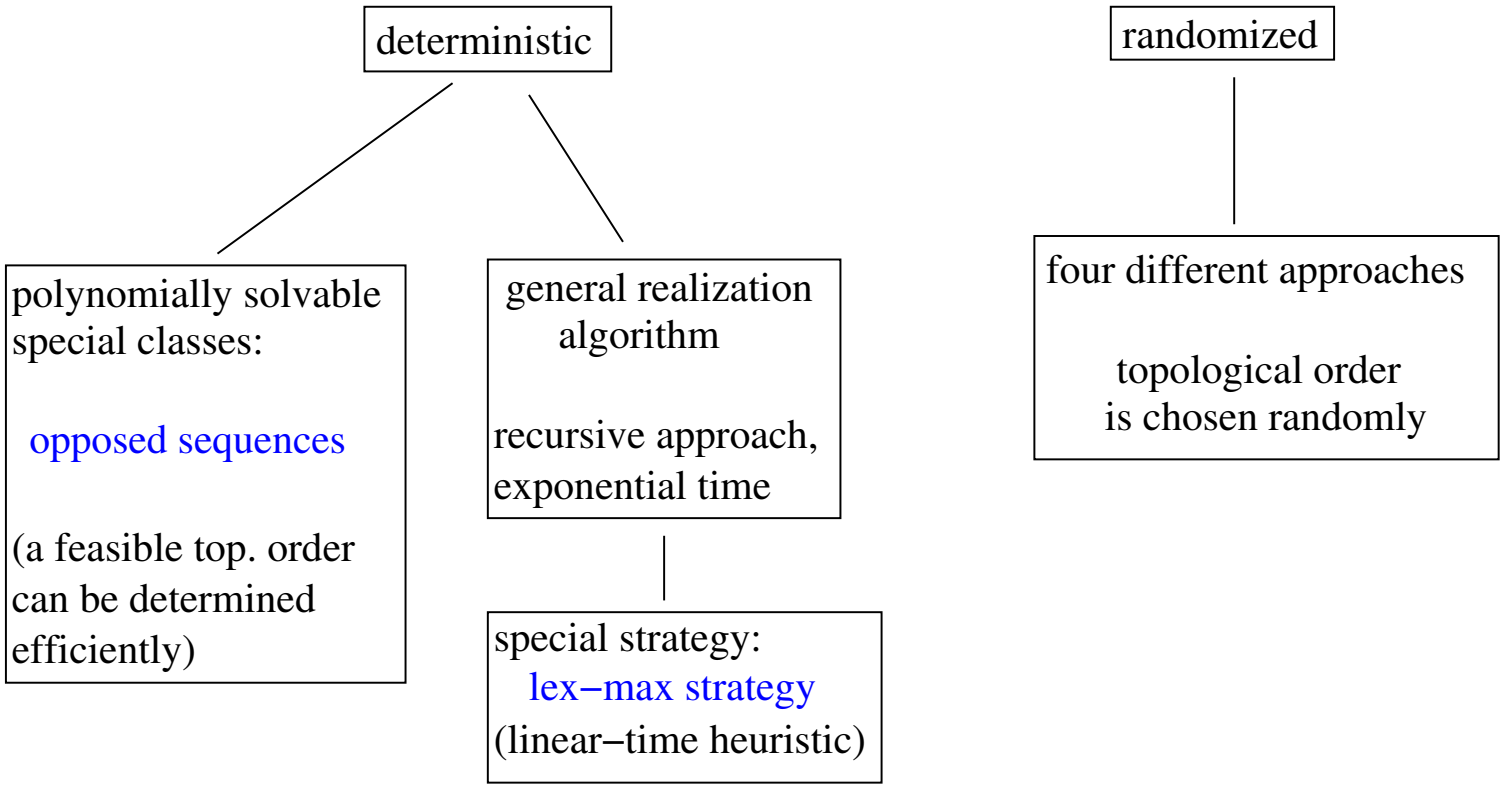
Greedy works (linear-time algorithm):

- connect first non-source vertex v_i with vertex degree $\binom{a_i}{b_i}$ with the a_i largest sources
- reduce $\binom{a_i}{b_i}$ to $\binom{0}{b_i}$, and the source out-degrees by one $\rightarrow$ yields new sequence S'
- we proved:
  if and only if these steps fail, the sequence is not realizable

This shows: Hardness lies in finding a feasible topological order

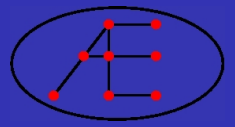
Overview: Our Contribution





- we made experiments for all these variants
- experiments show: it is hard to find sequences which we cannot solve in polynomial time

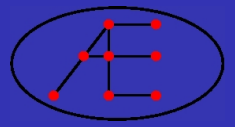
Opposed Relation



Definition (opposed relation)

Given are $c_1 := \binom{a_1}{b_1} \in \mathbb{Z}^2$ and $c_2 := \binom{a_2}{b_2} \in \mathbb{Z}^2$. We define:
$c_1 \leq_{opp} c_2 \Leftrightarrow (a_1 \leq a_2 \wedge b_1 \geq b_2)$.

Opposed Relation



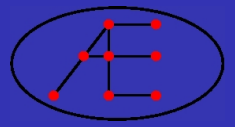
Definition (opposed relation)

Given are $c_1 := \binom{a_1}{b_1} \in \mathbb{Z}^2$ and $c_2 := \binom{a_2}{b_2} \in \mathbb{Z}^2$. We define:
$c_1 \leq_{opp} c_2 \Leftrightarrow (a_1 \leq a_2 \wedge b_1 \geq b_2)$.

Opposed relation defines a partial order

1. reflexive, transitive and antisymmetric relation
2. it is not possible to compare all tuples c_1 and c_2.

Opposed Relation



Definition (opposed relation)

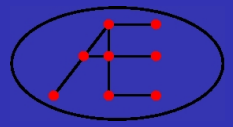
Given are $c_1 := \binom{a_1}{b_1} \in \mathbb{Z}^2$ and $c_2 := \binom{a_2}{b_2} \in \mathbb{Z}^2$. We define:
$c_1 \leq_{opp} c_2 \Leftrightarrow (a_1 \leq a_2 \wedge b_1 \geq b_2)$.

Opposed relation defines a partial order

1. reflexive, transitive and antisymmetric relation
2. it is not possible to compare all tuples c_1 and c_2.

Example: $\binom{2}{3} <_{opp} \binom{3}{1}$ but $\binom{2}{2}$, $\binom{3}{3}$ are not comparable

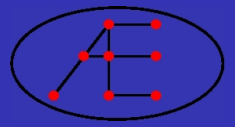
Opposed Sequences



Definition (opposed sequence)

We denote a sequence as opposed sequence, when it is possible to number all tuples (except for "sinks" and "sources") in a chain such that we have $\binom{a_i}{b_i} \leq_{opp} \binom{a_{i+1}}{b_{i+1}}$.

Opposed Sequences



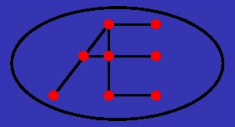
Definition (opposed sequence)

We denote a sequence as opposed sequence, when it is possible to number all tuples (except for "sinks" and "sources") in a chain such that we have $\binom{a_i}{b_i} \leq_{opp} \binom{a_{i+1}}{b_{i+1}}$.

Example: $\underbrace{\binom{0}{2}, \binom{0}{1}}_{\text{"sources"}}, \binom{1}{3} \leq_{opp} \binom{2}{2} \leq_{opp} \binom{2}{1}, \underbrace{\binom{2}{0}, \binom{2}{0}}_{\text{"sinks"}}$

Note: It is possible to sort all tuples (except for "sinks" and "sources") so that we have $a_i \leq a_{i+1}$ and $b_i \geq b_{i+1}$ for all indices i.

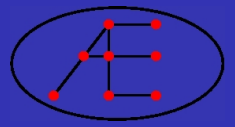
Realization of Opposed Sequences



We order an opposed sequence S containing at least one tuple (non-sink, non-source) such that:

1. at the beginning all source tuples build a decreasing sequence with respect to their b_i,
2. at the end all sink tuples build an increasing sequence with respect to their a_i,
3. number all remaining tuples (non-sinks and non-sources) in a chain such that we have $\binom{a_i}{b_i} \leq_{opp} \binom{a_{i+1}}{b_{i+1}}$, let $\binom{a_{i_{min}}}{b_{i_{min}}}$ be the first of them

Realization of Opposed Sequences



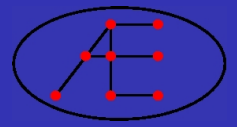
We order an opposed sequence S containing at least one tuple (non-sink, non-source) such that:

1. at the beginning all source tuples build a decreasing sequence with respect to their b_i,
2. at the end all sink tuples build an increasing sequence with respect to their a_i,
3. number all remaining tuples (non-sinks and non-sources) in a chain such that we have $\binom{a_i}{b_i} \leq_{opp} \binom{a_{i+1}}{b_{i+1}}$, let $\binom{a_{i_{min}}}{b_{i_{min}}}$ be the first of them

Theorem (opposed sequences, FCT 2011)

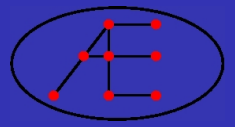
An opposed sequence S is a dag sequence if and only if there exist at least $a_{i_{min}}$ source tuples in S and if
$S' := \binom{0}{b_1-1}, \ldots, \binom{0}{b_{a_{i_{min}}}-1}, \binom{0}{b_{a_{i_{min}}}+1}, \ldots, \binom{0}{b_{i_{min}-1}}, \binom{0}{b_{i_{min}}}, \binom{a_{i_{min}+1}}{b_{i_{min}+1}}, \ldots, \binom{a_n}{b_n}$
is a dag sequence.

An Algorithmic Example



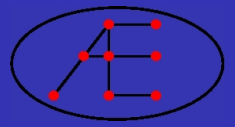
step 0 $\binom{0}{2}$, $\binom{0}{1}$, $\binom{2}{3}$, $\binom{2}{2}$, $\binom{2}{1}$, $\binom{1}{0}$, $\binom{2}{0}$

An Algorithmic Example



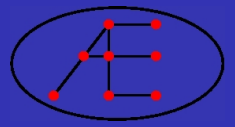
step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

An Algorithmic Example



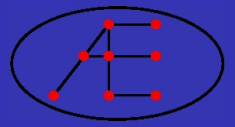
step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 2	$\binom{0}{1}$,	X	$\binom{0}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

An Algorithmic Example



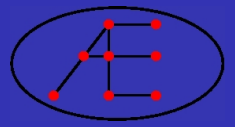
step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 2	$\binom{0}{1}$,	X	$\binom{0}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 3	$\binom{0}{1-1}$,	X	$\binom{0}{3-1}$,	$\binom{2-2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

An Algorithmic Example



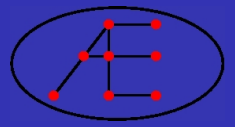
step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 2	$\binom{0}{1}$,	X	$\binom{0}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 3	$\binom{0}{1-1}$,	X	$\binom{0}{3-1}$,	$\binom{2-2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 4	X	X	$\binom{0}{2}$,	$\binom{0}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

An Algorithmic Example



step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 2	$\binom{0}{1}$,	X	$\binom{0}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 3	$\binom{0}{1-1}$,	X	$\binom{0}{3-1}$,	$\binom{2-2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 4	X	X	$\binom{0}{2}$,	$\binom{0}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 5	X	X	$\binom{0}{2-1}$,	$\binom{0}{2-1}$,	$\binom{2-2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

An Algorithmic Example

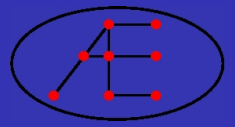


step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 2	$\binom{0}{1}$,	X	$\binom{0}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 3	$\binom{0}{1-1}$,	X	$\binom{0}{3-1}$,	$\binom{2-2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 4	X	X	$\binom{0}{2}$,	$\binom{0}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 5	X	X	$\binom{0}{2-1}$,	$\binom{0}{2-1}$,	$\binom{2-2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 6	X	X	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

– source-sink-sequence!

We apply an algorithm for digraphs (Kleitman, Wang 1973):

An Algorithmic Example



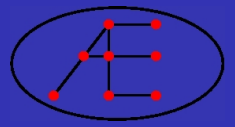
step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 2	$\binom{0}{1}$,	X	$\binom{0}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 3	$\binom{0}{1-1}$,	X	$\binom{0}{3-1}$,	$\binom{2-2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 4	X	X	$\binom{0}{2}$,	$\binom{0}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 5	X	X	$\binom{0}{2-1}$,	$\binom{0}{2-1}$,	$\binom{2-2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 6	X	X	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

– source-sink-sequence!

We apply an algorithm for digraphs (Kleitman, Wang 1973):

step 7	X	X	X	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{1}{0}$,	$\binom{1}{0}$

An Algorithmic Example



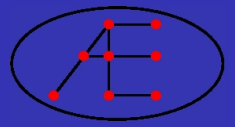
step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 2	$\binom{0}{1}$,	X	$\binom{0}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 3	$\binom{0}{1-1}$,	X	$\binom{0}{3-1}$,	$\binom{2-2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 4	X	X	$\binom{0}{2}$,	$\binom{0}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 5	X	X	$\binom{0}{2-1}$,	$\binom{0}{2-1}$,	$\binom{2-2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 6	X	X	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

– source-sink-sequence!

We apply an algorithm for digraphs (Kleitman, Wang 1973):

step 7	X	X	X	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{1}{0}$,	$\binom{1}{0}$
step 8	X	X	X	X	$\binom{0}{1}$,	X	$\binom{1}{0}$

An Algorithmic Example



step 0	$\binom{0}{2}$,	$\binom{0}{1}$,	$\binom{2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 1	$\binom{0}{2-1}$,	$\binom{0}{1-1}$,	$\binom{2-2}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 2	$\binom{0}{1}$,	X	$\binom{0}{3}$,	$\binom{2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 3	$\binom{0}{1-1}$,	X	$\binom{0}{3-1}$,	$\binom{2-2}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 4	X	X	$\binom{0}{2}$,	$\binom{0}{2}$,	$\binom{2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 5	X	X	$\binom{0}{2-1}$,	$\binom{0}{2-1}$,	$\binom{2-2}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$
step 6	X	X	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{1}{0}$,	$\binom{2}{0}$

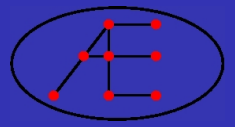
– source-sink-sequence!

We apply an algorithm for digraphs (Kleitman, Wang 1973):

step 7	X	X	X	$\binom{0}{1}$,	$\binom{0}{1}$,	$\binom{1}{0}$,	$\binom{1}{0}$
step 8	X	X	X	X	$\binom{0}{1}$,	X	$\binom{1}{0}$

Note: This algorithm can be implemented to run in time $O(m+n)$ using a "bucket" technique.

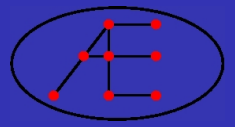
General Realization Algorithm



We order a sequence S containing at least one stream tuple such that:

1. at the beginning all source tuples, say q many, build a decreasing sequence with respect to their b_i,
2. at the end all sink tuples build an increasing sequence with respect to their a_i,

General Realization Algorithm



We order a sequence S containing at least one stream tuple such that:

1. at the beginning all source tuples, say q many, build a decreasing sequence with respect to their b_i,
2. at the end all sink tuples build an increasing sequence with respect to their a_i,

candidate set V_{min}: all stream tuples which satisfy

1. $a_i \leq q$ (indegree does not exceed # available sources) and
2. there does not exist a smaller stream tuple with respect to the opposed relation $<_{opp}$.

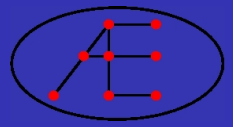
Theorem (FCT 2011)

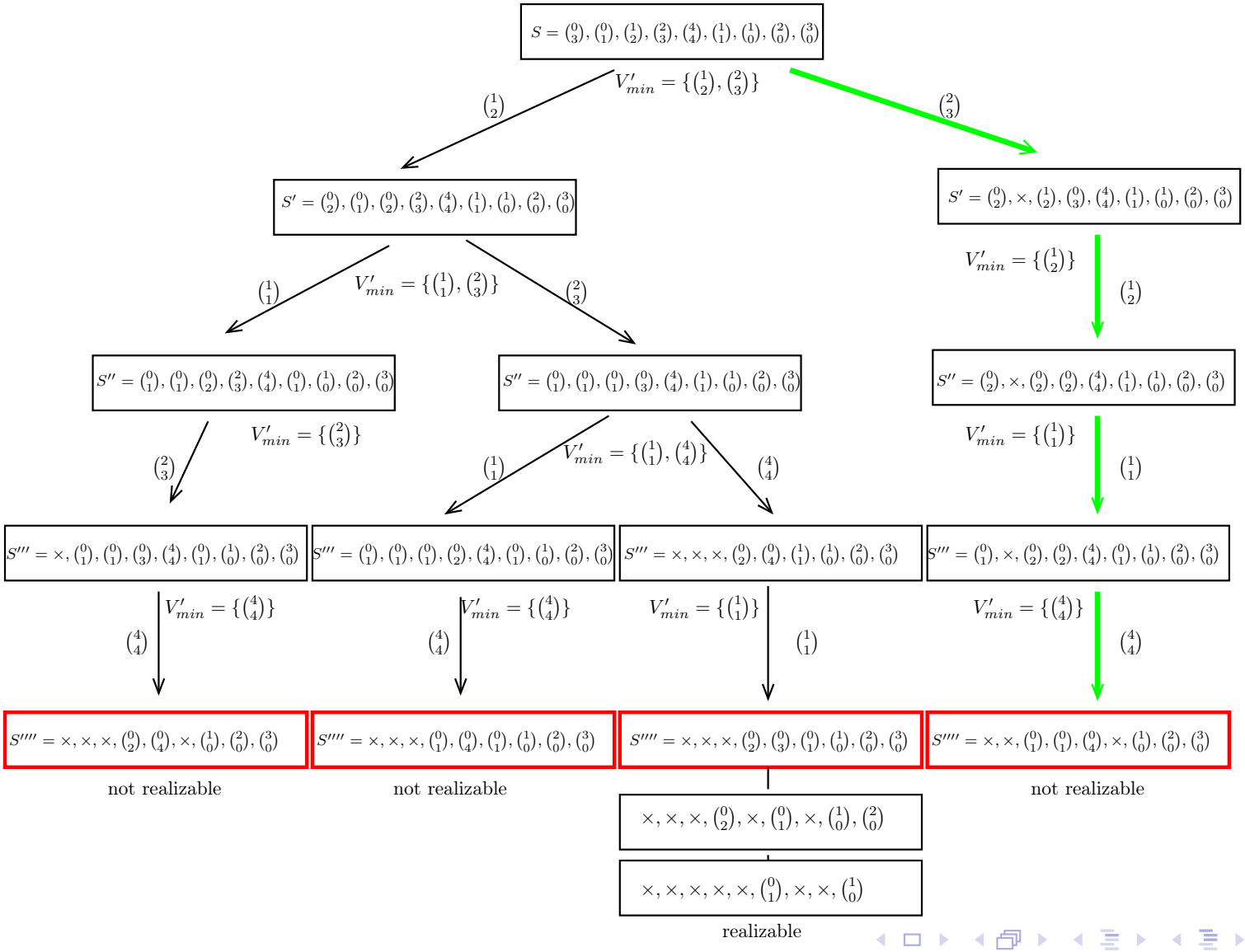
S is a dag sequence if and only if $V_{min} \neq \emptyset$ and there exists an element $\binom{a_{i_{min}}}{b_{i_{min}}} \in V_{min}$ such that $S' :=$

$$\binom{0}{b_1-1}, \ldots, \binom{0}{b_{a_{i_{min}}}-1}, \binom{0}{b_{a_{i_{min}}}+1}, \ldots, \binom{0}{b_q}, \binom{a_{q+1}}{b_{q+1}}, \ldots, \binom{a_{i_{min}}-1}{b_{i_{min}}-1}, \binom{0}{b_{i_{min}}}, \binom{a_{i_{min}}+1}{b_{i_{min}}+1}, \ldots, \binom{a_n}{b_n}$$

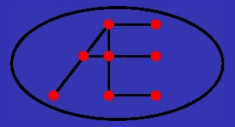
is a dag sequence.

Example: Recursion Tree





Lex Max Strategy



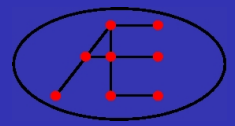
Observations:

1. bottleneck is the cardinality of V_{min}
2. for opposed sequences we have a smallest tuple resulting in $|V_{min}| = 1$

"lex max strategy":
choose always the lexicographical largest tuple in V_{min}

Early conjecture:
Lex max strategy works

Story of the Lex Max Strategy And Why We Became Curious

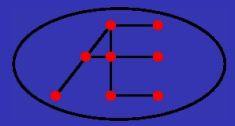


Note: when we started our work, the complexity status of dag realization was still open

Initial experiments:

1. we generated two million dag sequences randomly (for various sequence sizes)
2. observed success in each case for the lex max strategy

Story of the Lex Max Strategy And Why We Became Curious



Note: when we started our work, the complexity status of dag realization was still open

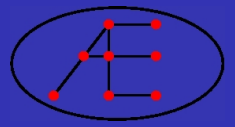
Initial experiments:

1. we generated two million dag sequences randomly (for various sequence sizes)
2. observed success in each case for the lex max strategy

But: When we tried to prove "correctness" of the strategy, we finally managed to construct counter-example(s)

Lesson:
randomly generated instances turn out to be easy instances

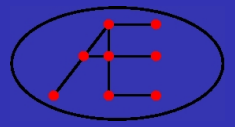
Three Types of Test Instances



1. generation of "random sequences"
   - sample uniformly dags with n vertices and m arcs
   - take the corresponding dag sequence

Note: We sample uniformly dags, but not sequences.

Three Types of Test Instances

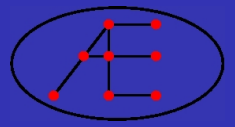


1. generation of "random sequences"
   - sample uniformly dags with n vertices and m arcs
   - take the corresponding dag sequence

   Note: We sample uniformly dags, but not sequences.

2. systematic generation of dag sequences
   - generate all non-isomorphic dag sequences with $7, 8, 9$ tuples
   - Note: this is infeasible for $n \geq 10$!
   - Ignore all "trivial sequences" (with ≤ 1 stream tuples)

Three Types of Test Instances

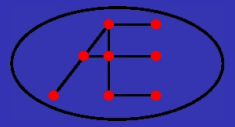


1. generation of "random sequences"
   - sample uniformly dags with n vertices and m arcs
   - take the corresponding dag sequence

   Note: We sample uniformly dags, but not sequences.

2. systematic generation of dag sequences
   - generate all non-isomorphic dag sequences with $7, 8, 9$ tuples
   - Note: this is infeasible for $n \geq 10$!
   - Ignore all "trivial sequences" (with ≤ 1 stream tuples)
3. degree sequences derived from real-world dags

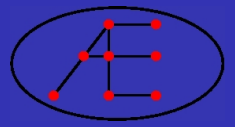
Experiments I



First questions:

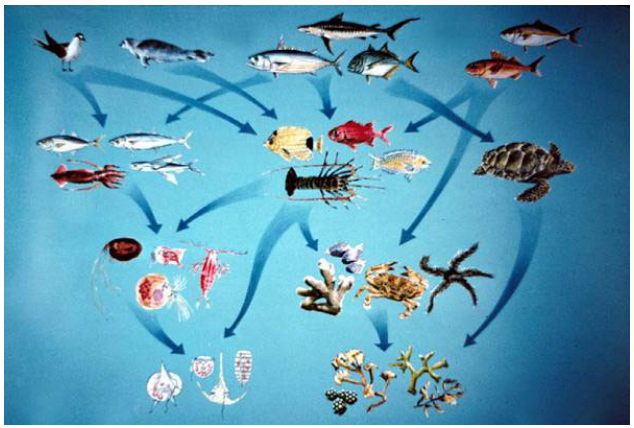
1. How relevant are opposed sequences?
2. How large is the fraction of dag sequences which are realizable by using the lex max strategy?
3. How difficult are degree sequences derived from real-world dags?

Acyclic Real World Networks



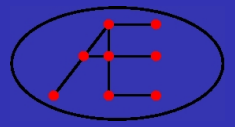
We considered:

1. OBDDs (ordered binary decision diagrams)
2. public train transport schedule (20000 tuples)
3. flight schedules (37800 tuples)
4. several food webs (40 to 150 tuples)



Our observations: All instances are realizable by the lex max strategy.

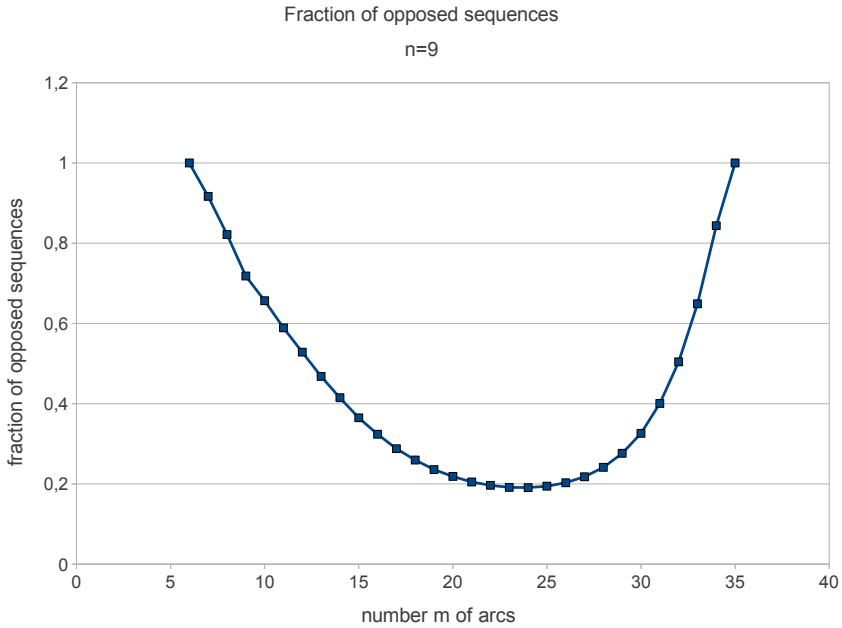
Opposed Sequences



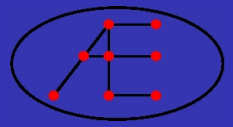
Which fraction of sequences are opposed sequences?

Observations

1. sequences with a middle density have the smallest fraction of opposed sequences
2. opposed sequences are a relevant class of sequences



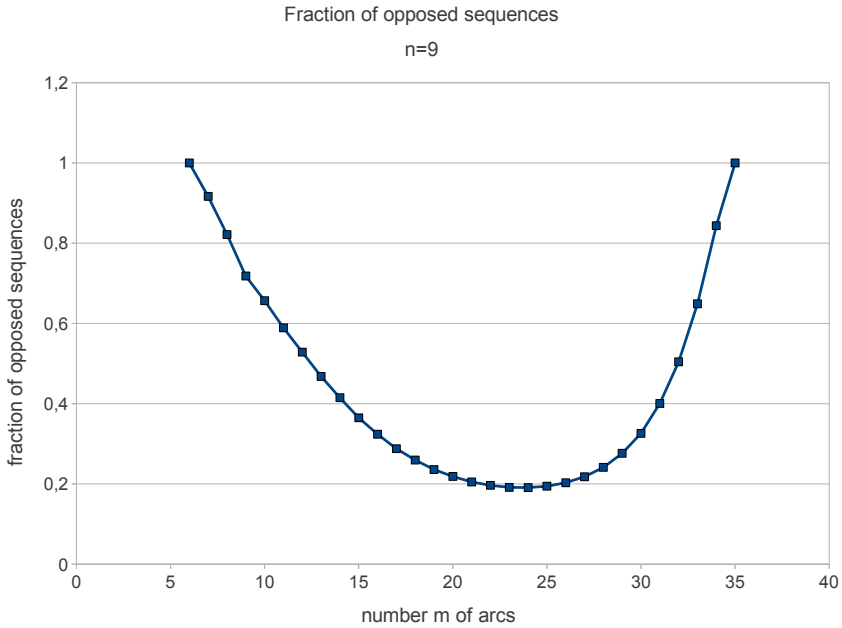
Opposed Sequences



Which fraction of sequences are opposed sequences?

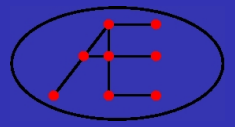
Observations

1. sequences with a middle density have the smallest fraction of opposed sequences
2. opposed sequences are a relevant class of sequences



Note: OBDDs (ordered binary decision diagrams) are dags with opposed dag sequences.

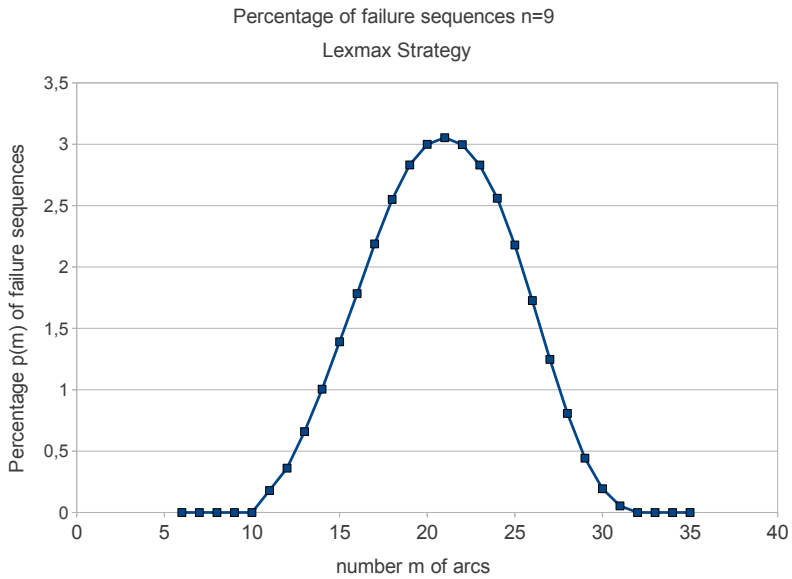
Lex Max Strategy



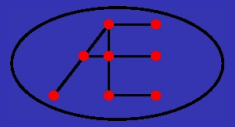
How often does the lex max strategy fail?

Observation

1. lex max strategy leads to a dag realization for at least 97% of all dag sequences with 9 tuples
2. a strong connection between the density of a sequence and the realizability



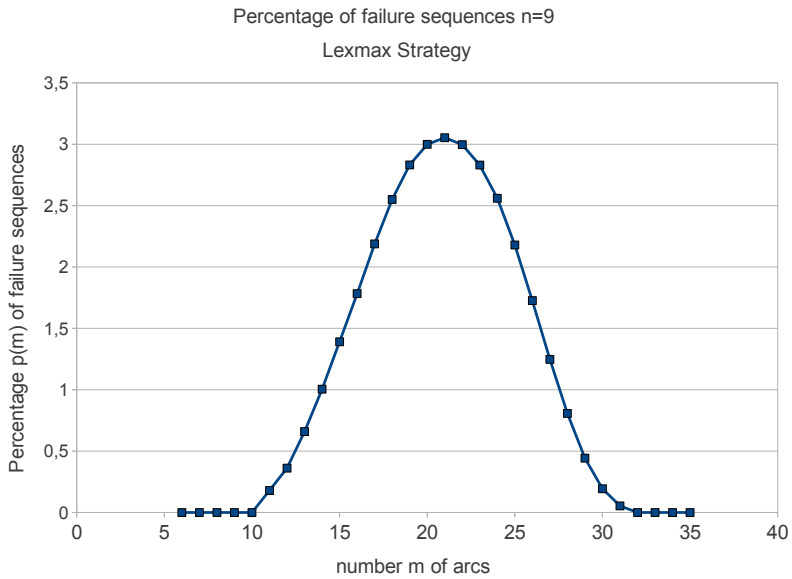
Lex Max Strategy



How often does the lex max strategy fail?

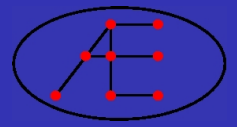
Observation

1. lex max strategy leads to a dag realization for at least 97% of all dag sequences with 9 tuples
2. a strong connection between the density of a sequence and the realizability



But: This result does not explain our observation at the beginning – "success for 2 million randomly chosen dag sequences" with ≥ 20 tuples.

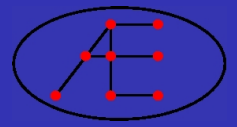
Distance to Opposed



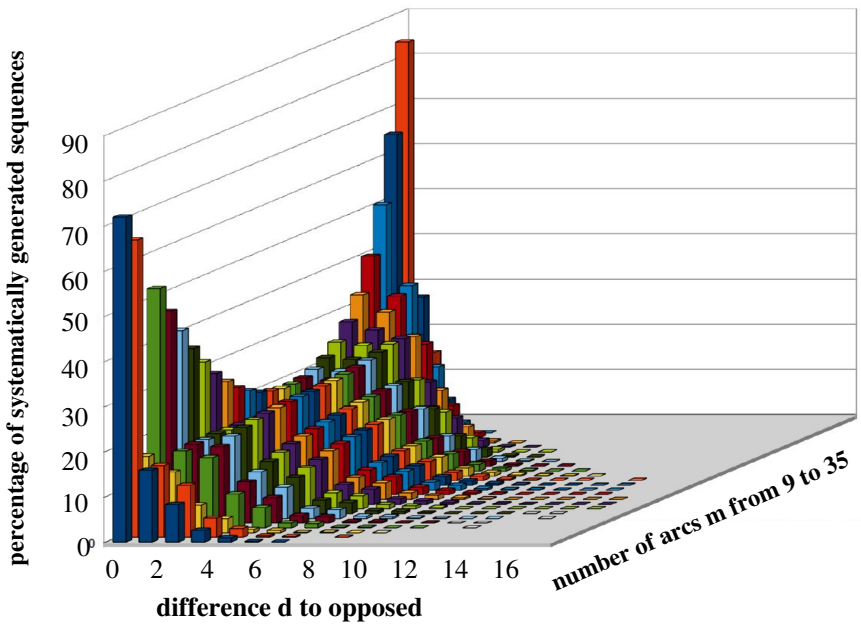
- opposed sequences are efficiently solvable
- would like to have a measure:
  how similar is a sequence to being an opposed sequence?
- distance to opposed $=$ # pairwise incomparable stream tuples with respect to some specific order

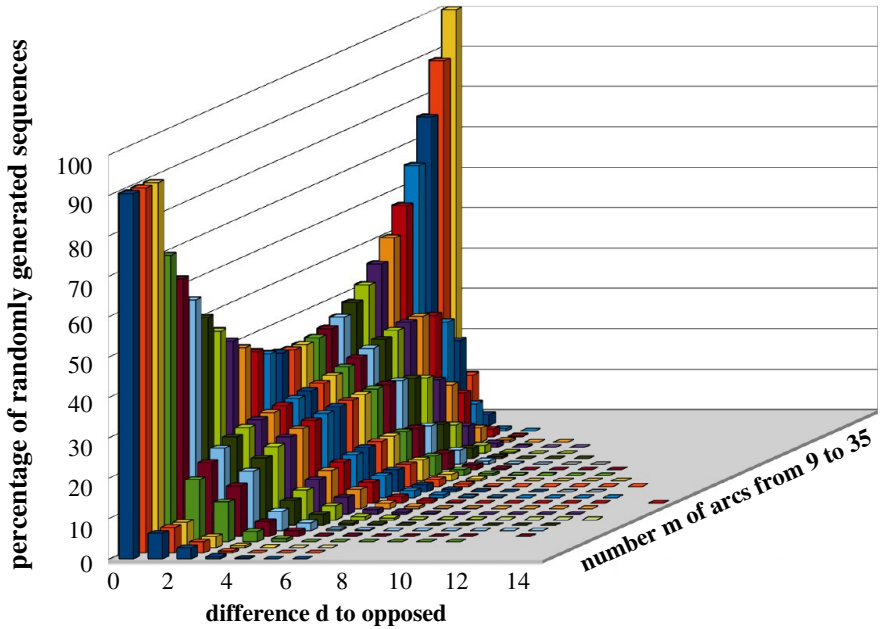
$$d(S) := \left| \left\{ \left(\begin{pmatrix} a_i \\ b_i \end{pmatrix}, \begin{pmatrix} a_j \\ b_j \end{pmatrix} \right) \mid \begin{pmatrix} a_i \\ b_i \end{pmatrix}, \begin{pmatrix} a_j \\ b_j \end{pmatrix} \text{ incomparable stream tuples w.r.t. } \leq_{opp} \text{ and } i < j \right\} \right|.$$

Distance to Opposed



Question: Do randomly generated sequences possess a preference to a "small" distance to opposed in comparison with systematically generated sequences?

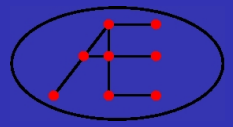




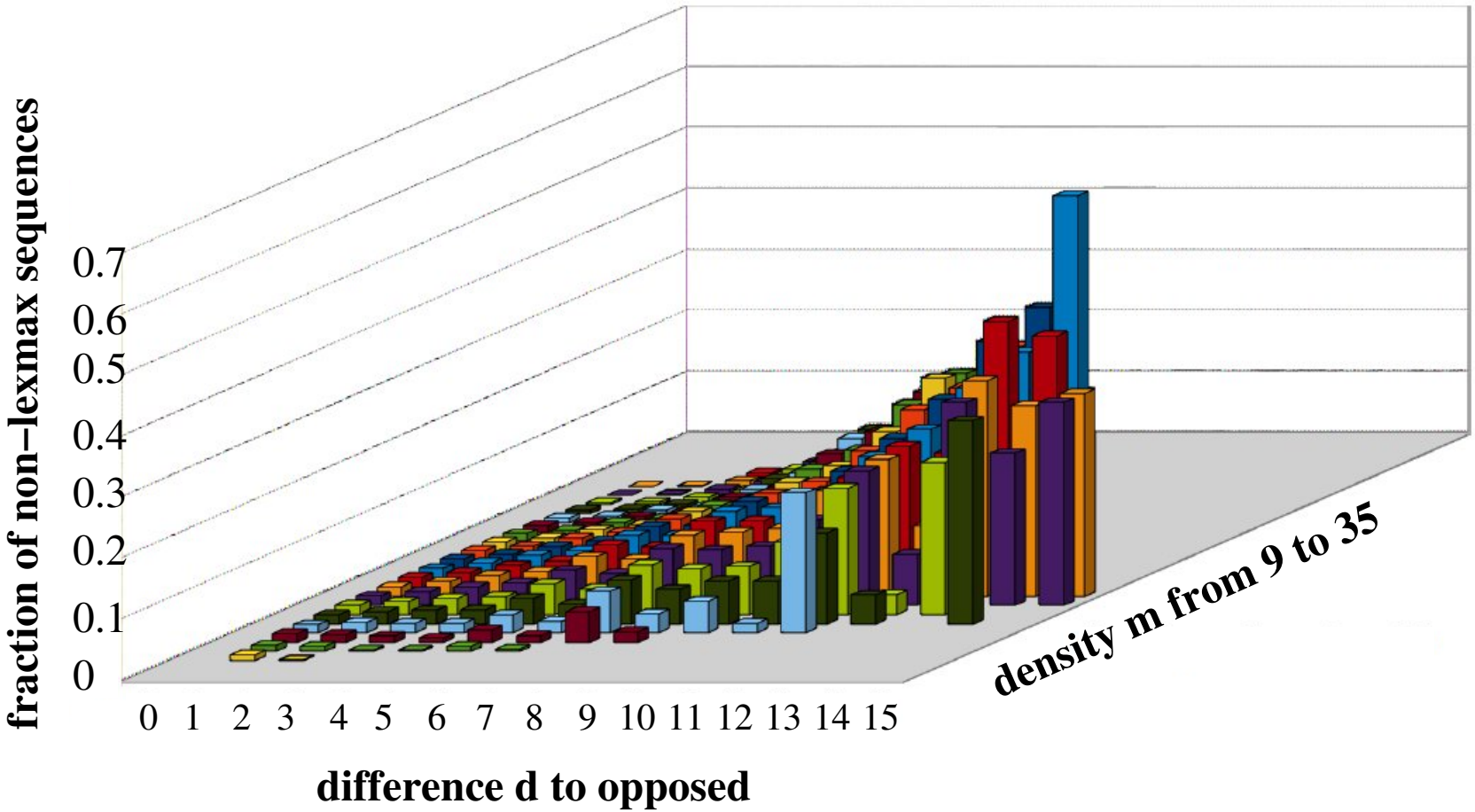
Systematic vs. randomized generation of sequences

YES, there is a clear bias towards smaller distance to opposed for random instances.

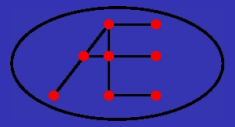
Distance to Opposed



Question: Do non-lexmax sequences possess a preference for large opposed distances?



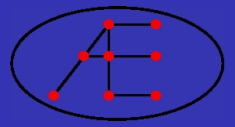
Back to Theory



Observation: very sparse instances ($m < n$) "forest dags" are always solvable by lex max strategy

Is there a theoretical explanation?

Back to Theory



Observation: very sparse instances ($m < n$) "forest dags" are always solvable by lex max strategy

Is there a theoretical explanation?

Yes, and even more: every choice of a tuple in V_{min} provably works!

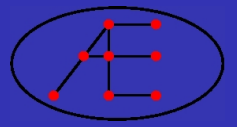
Theorem (Realization of forest dags in linear time)

Let $S := \binom{a_1}{b_1}, \ldots, \binom{a_n}{b_n}$ with $\sum_{i=1}^{n} a_i \leq n-1$ be a canonically sorted sequence containing $k > 0$ source tuples. Furthermore, we assume that S is not a source-sink-sequence. Consider an arbitrary stream tuple $\binom{a_i}{b_i}$ with $a_i \leq k$.
S is a dag sequence if and only if

$$S' := \binom{0}{b_1 - 1}, \ldots, \binom{0}{b_{a_i} - 1}, \binom{0}{b_{a_i+1}}, \ldots, \binom{0}{b_k}, \ldots, \binom{a_i - 1}{b_i - 1}, \binom{0}{b_i}, \binom{a_i+1}{b_{i+1}}, \ldots, \binom{a_n}{b_n}$$

is a dag sequence.

Randomized Strategy I (Rand I)



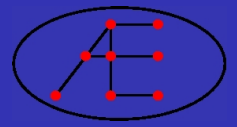
Rand I:

1. choose a random permutation of the (stream) tuples
2. apply the linear-time realization algorithm for prescribed topological orders

Note:

- considers **all** permutations of stream tuples
- has a high probability to fail

Rand II: Exploit Necessary Conditions



Let S be a dag sequence with n tuples.

q — number of source tuples in S
s — number of sink tuples in S

Lemma (necessary criterion for the realizability of dag sequences)

If a stream tuple $\binom{a}{b}$ occurs at position i in a topological order of a dag realization, then it follows that

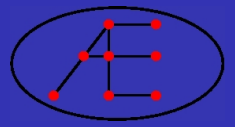
$$a \leq \min\{n - s, i - 1\}$$

and

$$b \leq \min\{n - q, n - i\}.$$

Our task: Find a topological order which fulfills these conditions for all stream tuples simultaneously.

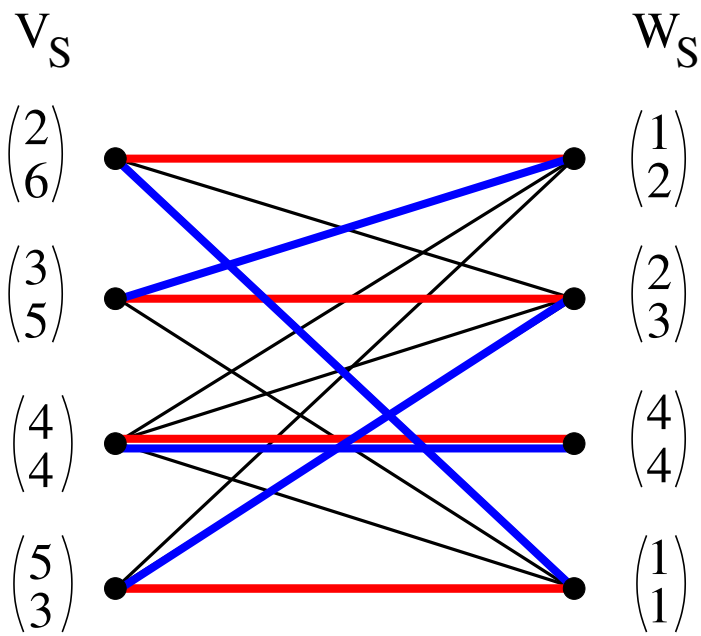
Rand II: Exploit Necessary Conditions



Reformulation as a perfect matching problem in a bipartite graph (the so-called bounding graph)

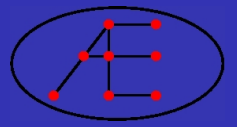
Example: sequence $S := \binom{0}{3}, \binom{0}{1}, \binom{1}{2}, \binom{2}{3}, \binom{4}{4}, \binom{1}{1}, \binom{1}{0}, \binom{2}{0}, \binom{3}{0}$

bounds for positions 3 to 6



stream tuples of S

Randomized Strategy II (Rand II)



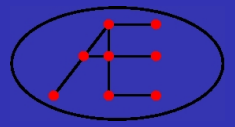
Rand II:

1. choose a random perfect matching in the bounding graph
2. let P be the corresponding permutation of tuples
3. apply a linear-time realization algorithm (subject to the fixed permutation P)

Note: a random perfect matching can be determined in polynomial time, $O(n^8(n \log n + \log \frac{1}{\varepsilon}) \log \frac{1}{\varepsilon})$, ε denotes deviation from uniform distribution
(Jerrum, Sinclair and Vigoda, 2004)

in our experiments:
we compute the average running time over all perfect matchings

Randomized Strategies III and IV



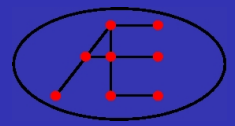
Rand III:

- recall our recursive approach:
  if the sequence is realizable, then the set V_{min} contains at least one element by which we can reduce the sequence
- our general realization algorithm branches over all elements of V_{min}
- instead of branching, we sample the next stream tuple uniformly at random from the set V_{min}

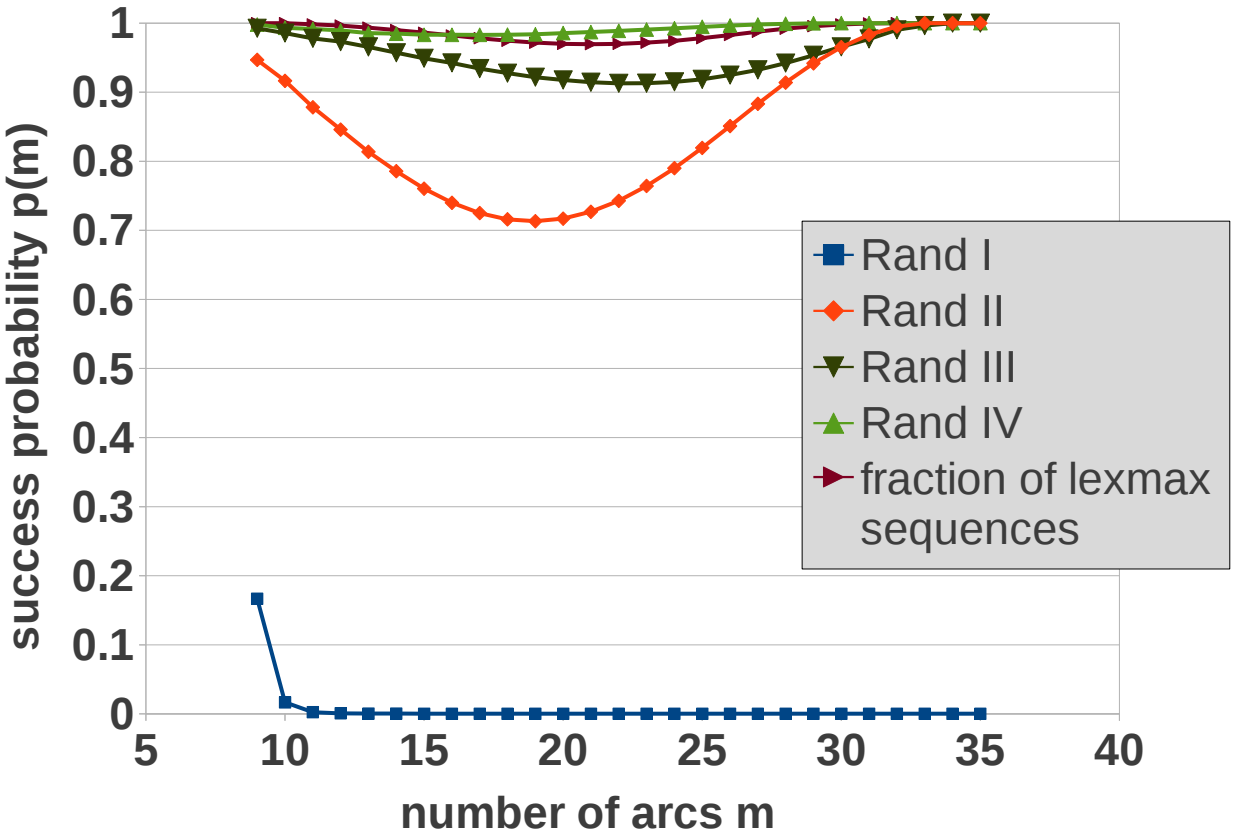
Rand IV:

- combine Rand III with reduction rules
- for details see full paper

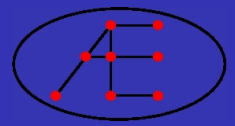
Success Probability of the Randomized Strategies



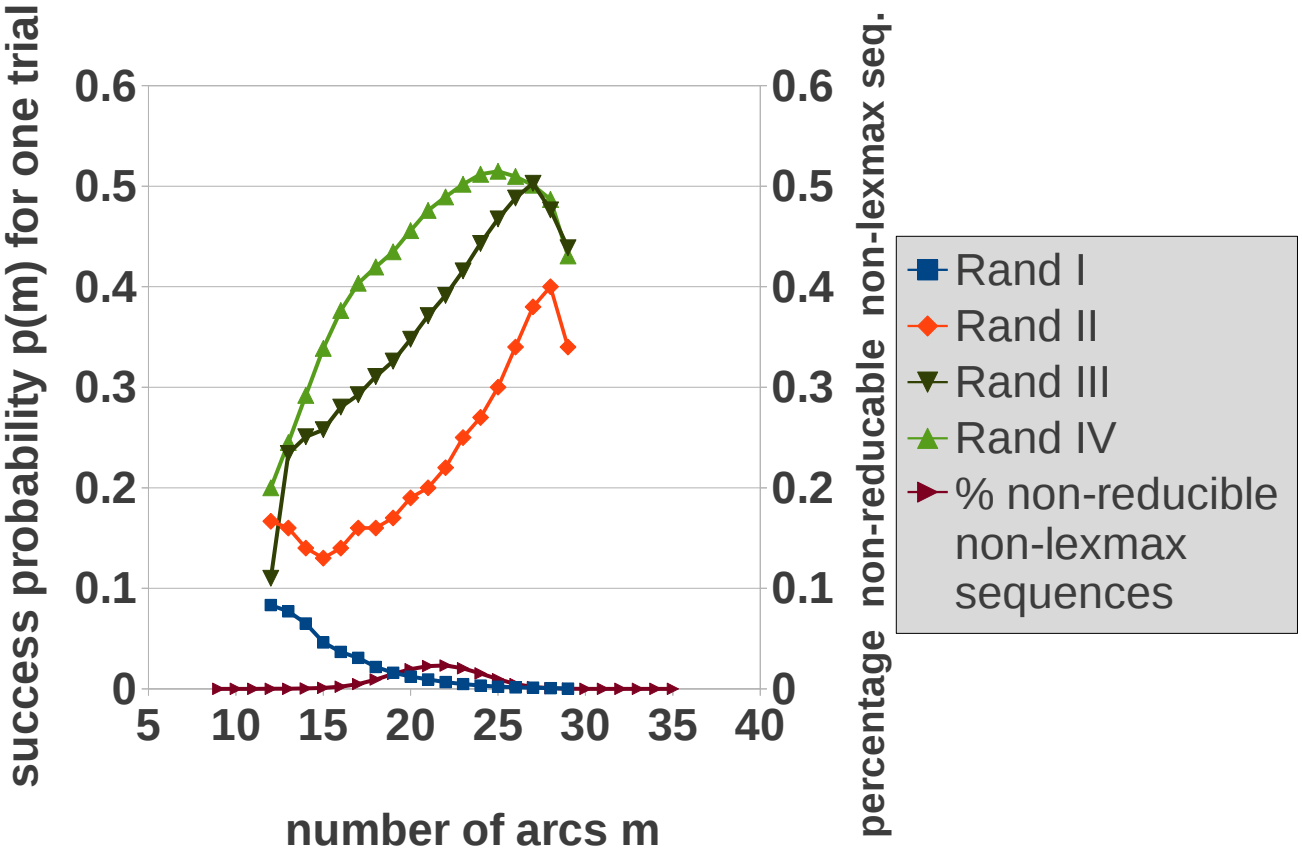
All non-trivial sequences on 9 tuples:



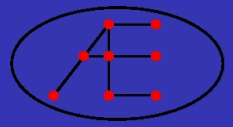
Success Probability of the Randomized Strategies

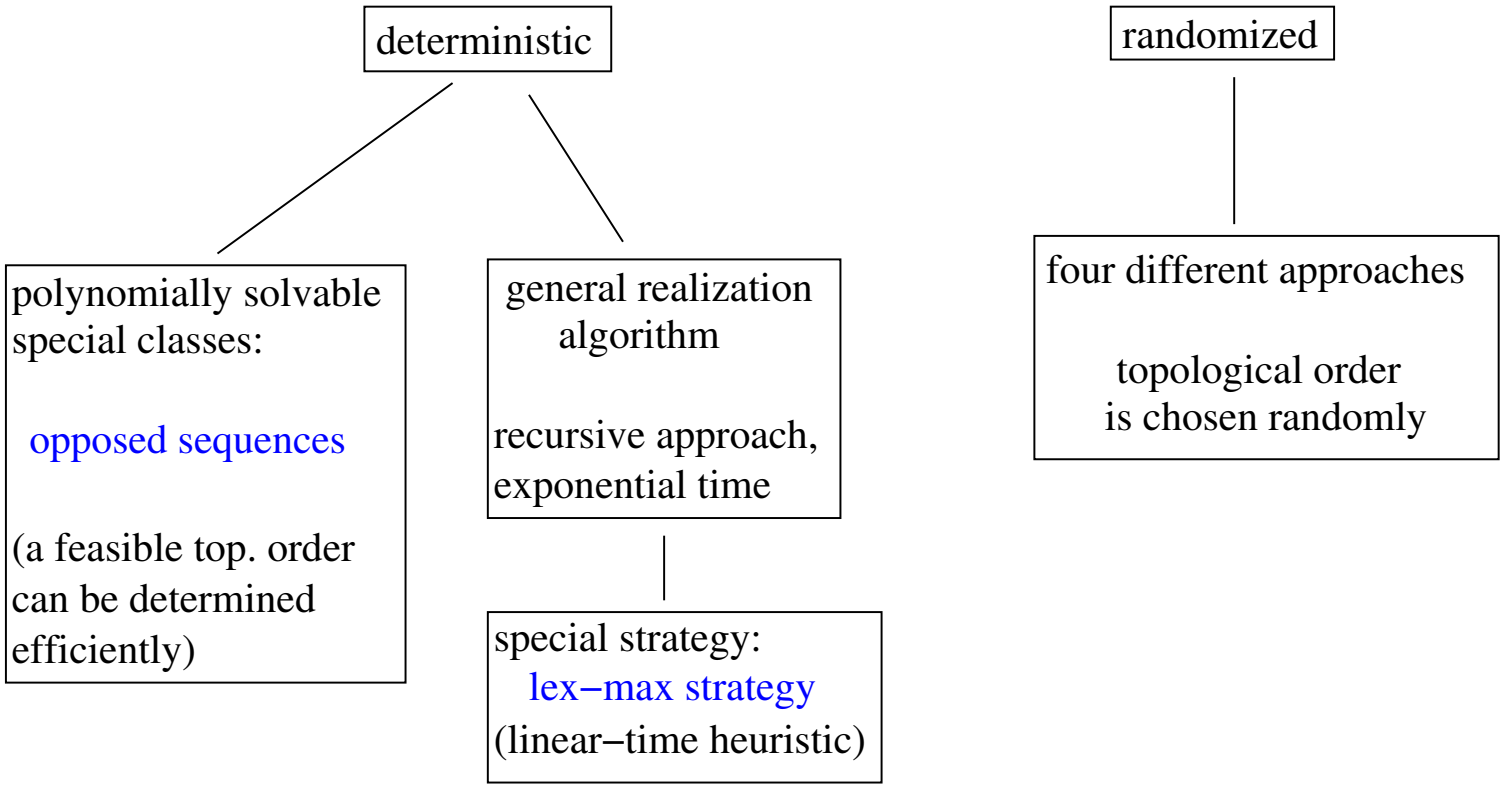


Restriction to non-reducible, non-lexmax sequences of 9 tuples:



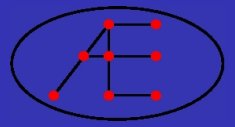
Summary: Our Contribution





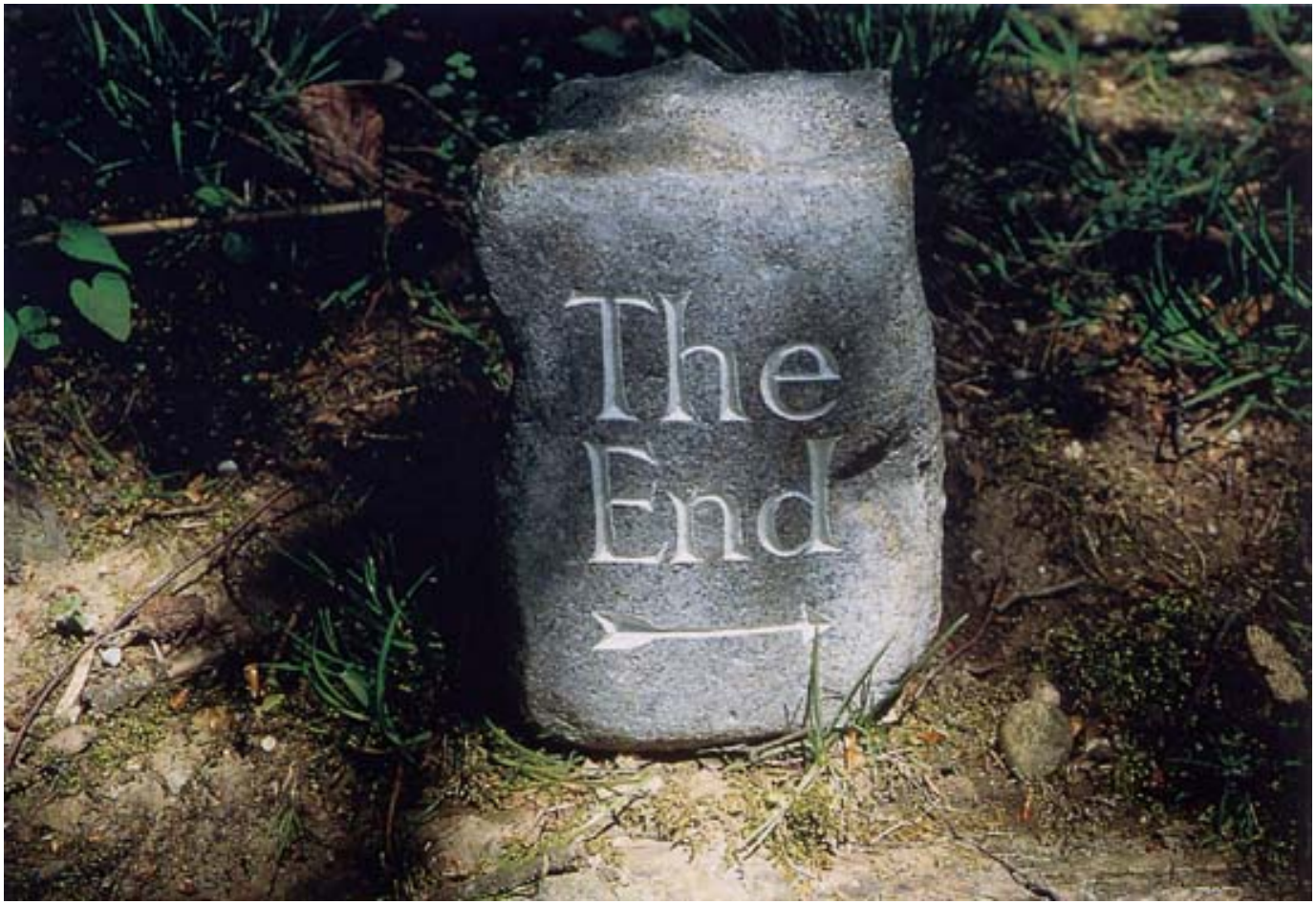
- lex max strategy and RAND IV are remarkably successful
- all real-world instances solved easily in linear time

Future Work



To do:

- characterize the class of instances for which the lex max strategy works provably correct
- identify other classes of instances which allow polynomial-time algorithms
- provide a theoretical analysis of the randomized approaches


The
End