Parameterized Complexity Results for Probabilistic Network Structure Learning

Stefan Szeider
Vienna University of Technology, Austria

Workshop on Applications of
Parameterized Algorithms and Complexity
APAC 2012
Warwick, UK, July 8, 2012

Joint work with:

Serge Gaspers
UNSW, Sydney

Mikko Koivisto
U Helsinki

Mathieu Liedloff
U d'Orleans

Sebastian Ordyniak
Masaryk U Brno

Papers:

- Algorithms and Complexity Results for Exact Bayesian Structure Learning. Ordyniak and Szeider. Conference on Uncertainty in Artificial Intelligence (**UAI 2010**).
- An Improved Dynamic Progamming Algorithm for Exact Bayesian Network Structure Learning. Ordyniak and Szeider. NIPS Workshop on Discrete Optimization in Machine Learning (**DISCML 2011**).
- On Finding Optimal Polytrees. Gaspers, Koivisto, Liedloff, Ordyniak, and Szeider. Conference on Artificial Intelligence (**AAAI 2012**).

Probabilistic Networks

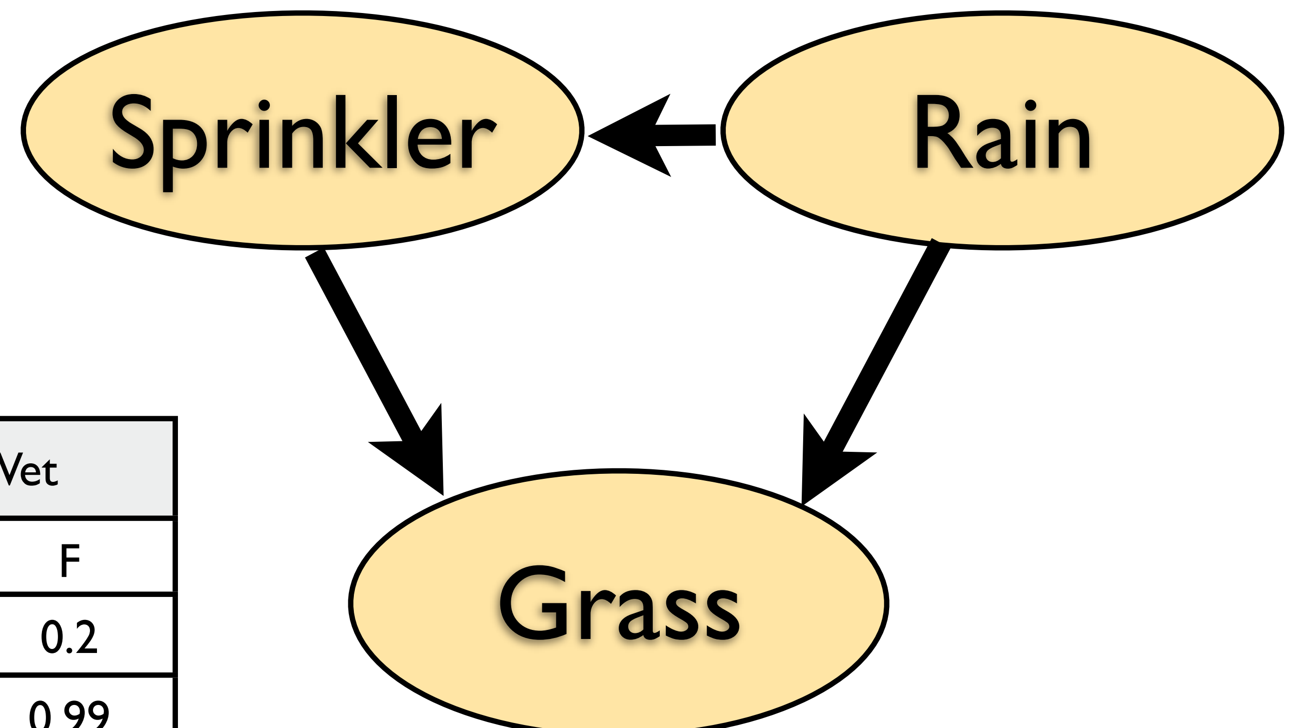
- Bayesian Networks (BNs) were introduced by Judea Perl in 1985 (2011 Turing Award Winner)
- A BN is a DAG D=(V,A) plus tables associated with the nodes of the network
- In addition to Bayesian networks, also other probabilistic networks have been considered: *Markov Random Fields, Factor Graphs*, etc.

Judea Pearl

Example: A Bayesian Network

Rain	Sprinkler	
	T	F
F	0.4	0.6
T	0.01	0.99

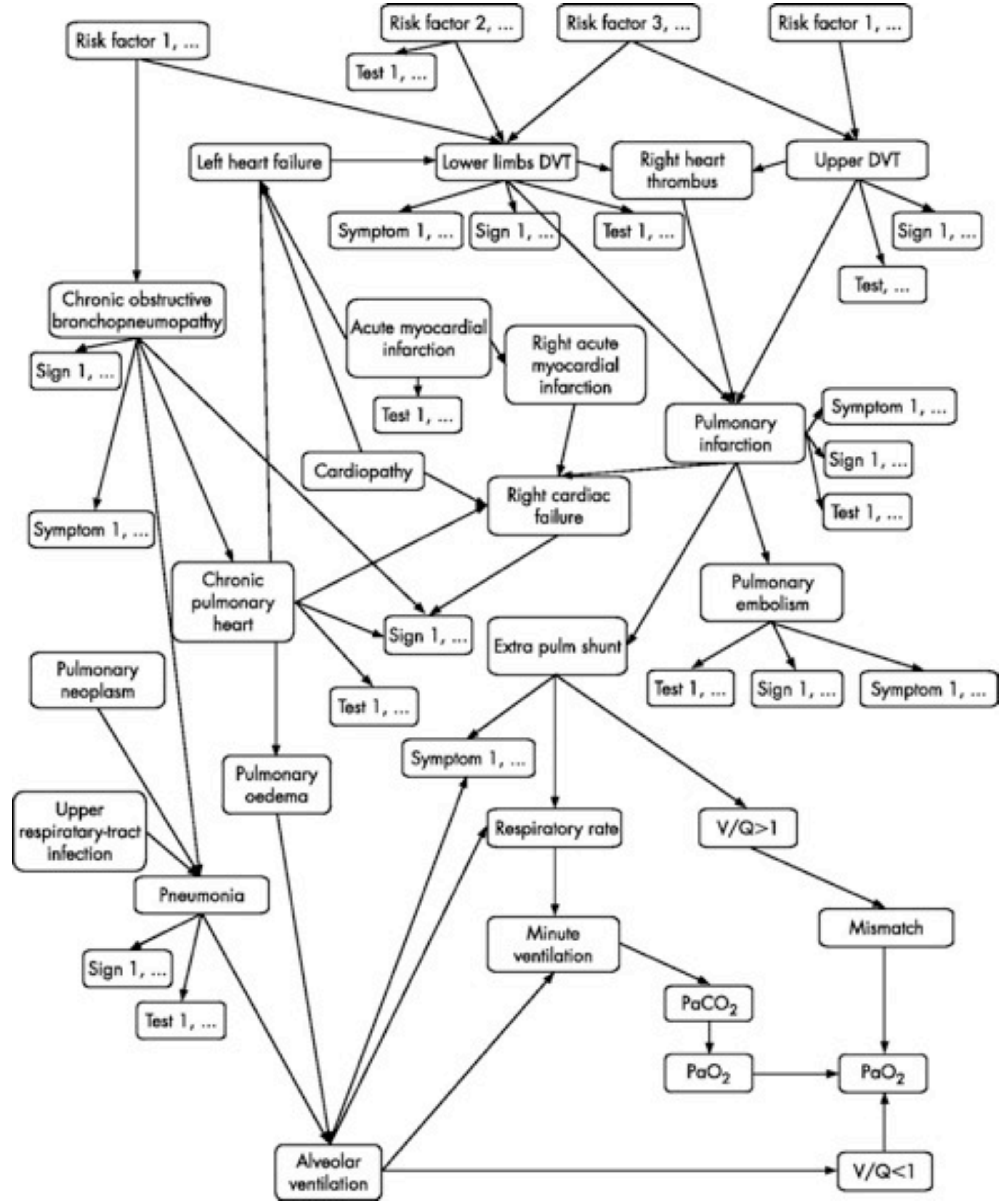
Rain	
T	F
0.8	0.2



Sprinkler	Rain	Grass Wet	
		T	F
F	F	0.8	0.2
F	T	0.01	0.99
T	F	0.9	0.1
T	T	0.99	0.01

Applications

- diagnosis
- computational biology
- document classification
- information retrieval
- image processing
- decision support
- etc.



A BN showing the main pathophysiological relationships in the diagnostic reasoning focused on a suspected pulmonary embolism event

Computational Problems

Computational Problems

- BN Reasoning: Given a BN, compute the probability of a variable taking a specific value

Computational Problems

- BN Reasoning: Given a BN, compute the probability of a variable taking a specific value
- BN Learning: Given a set of sample data, find a BN that fits the data best.

Computational Problems

- BN Reasoning: Given a BN, compute the probability of a variable taking a specific value
- BN Learning: Given a set of sample data, find a BN that fits the data best.
    - BN *Structure* Learning: Given sample data, find the best DAG

Computational Problems

- BN Reasoning: Given a BN, compute the probability of a variable taking a specific value
- BN Learning: Given a set of sample data, find a BN that fits the data best.
  - BN *Structure* Learning: Given sample data, find the best DAG
  - BN *Parameter* Learning: Given sample data and a DAG, find the best probability tables

Computational Problems

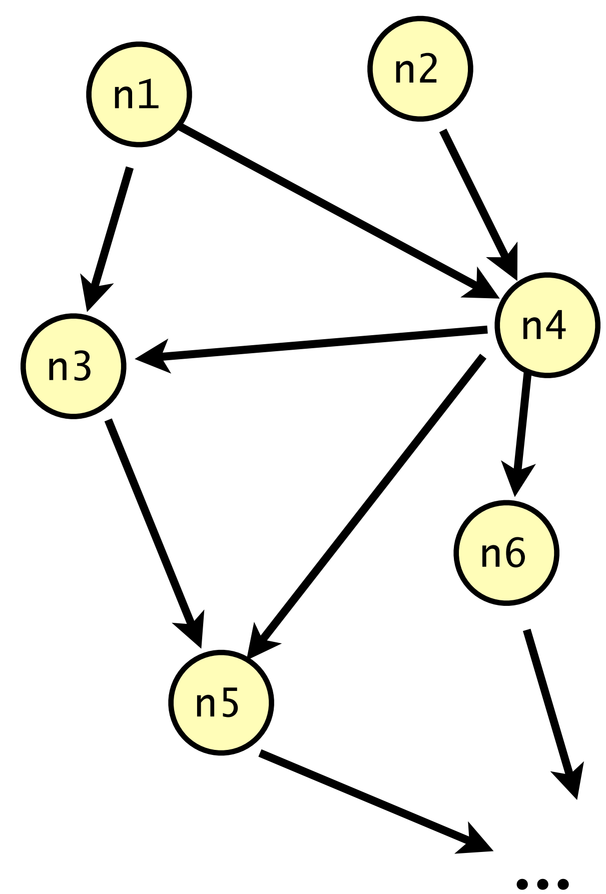
- BN Reasoning: Given a BN, compute the probability of a variable taking a specific value
- BN Learning: Given a set of sample data, find a BN that fits the data best.
  - BN *Structure* Learning: Given sample data, find the best DAG
  - BN *Parameter* Learning: Given sample data and a DAG, find the best probability tables

BN Learning and Local Scores

Sample Data

```
n1  n2  n3  n4  n5  n6  ....
0   1   0   1   1   0   ....
1   1   0   1   1   0   ....
*   1   0   0   *   1   ....
0   *   1   1   *   1   ....
1   0   1   1   0   1   ....
....
....
```

BN



BN Learning and Local Scores

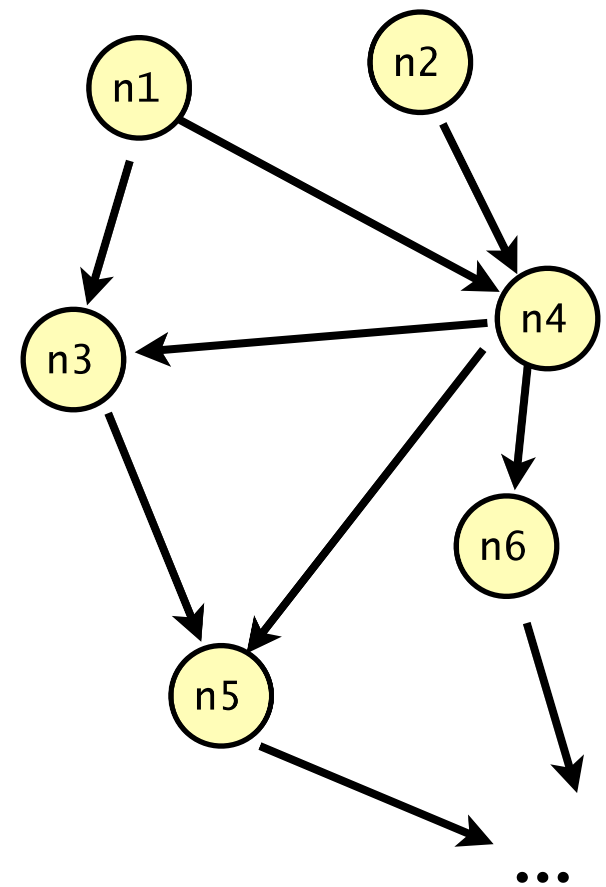
Local Score Function

f(n,P)=score of node n with P ⊆V
as its in-neighbors (parents)

Sample Data

```
n1 n2 n3 n4 n5 n6 ....
0  1  0  1  1  0  ....
1  1  0  1  1  0  ....
*  1  0  0  *  1  ....
0  *  1  1  *  1  ....
1  0  1  1  0  1  ....
....
....
```

BN



Our Combinatorial Model

- BN Structure Learning:

Input: a set N of nodes and a local sore function f (by explicit listing of nonzero tuples)

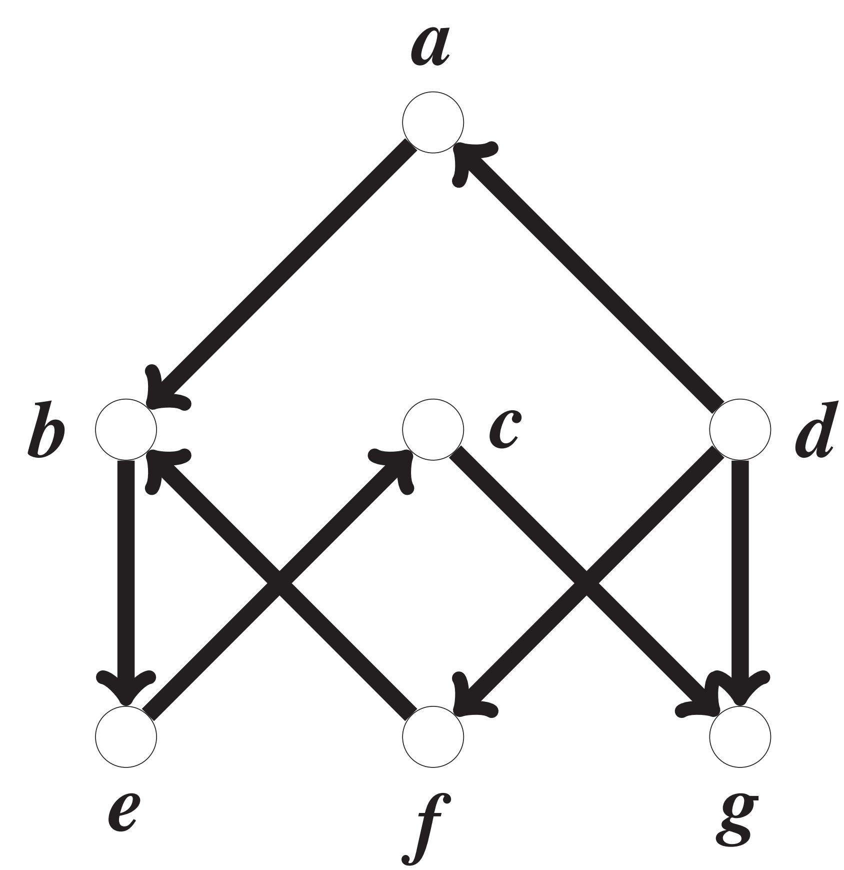
Task: find a DAG D=(N,A) such that the sum of $f(n,P_D(n))$ over all nodes n is maximum.

Example

A local score function

n	P	$f(n,P)$
a	$\{d\}$	1
a	$\{b,c,d\}$	0.5
b	$\{a,f\}$	1
c	$\{e\}$	1
d	$\emptyset$	1
e	$\{b\}$	1
f	$\{d\}$	1
g	$\{c,d\}$	1

An optimal BN structure

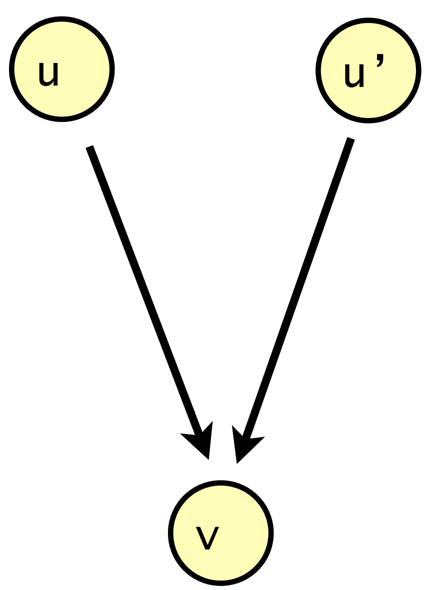


Heuristic Algorithms: find first an undirected graph, then greedily orient the edges, then do local search for improving the orientation

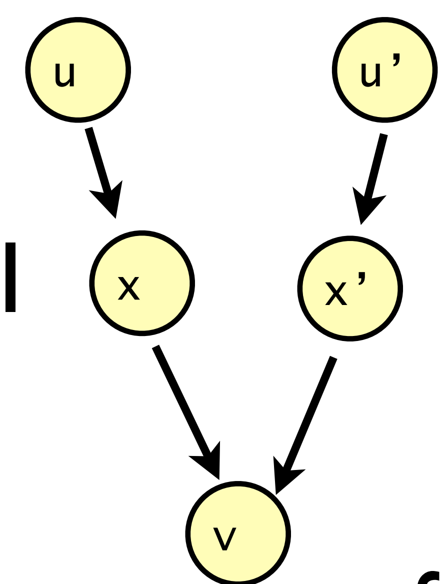
NP-Hardness

Theorem: BN Structure Learning is NP-hard.
[Chickering 1996]

- Proof: By reduction from FAS
  (NP-h for max deg 4, [Karp 1972])



$f(x,\{u\})=1$



$f(x',\{u'\})=1$

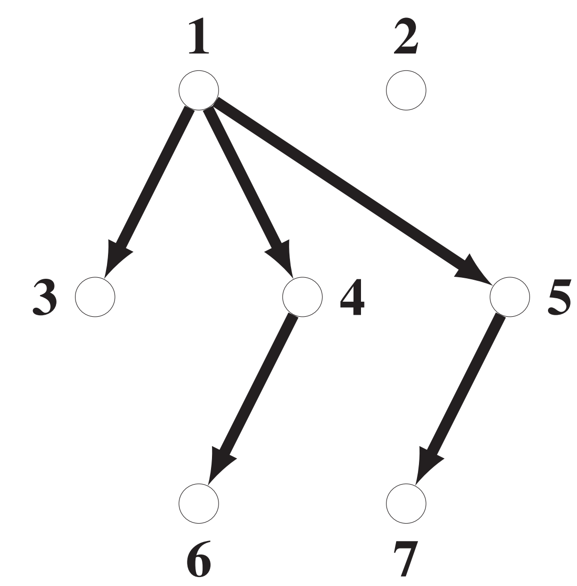
$f(v,\{x\})=1$
$f(v,\{x'\})=1$
$f(v,\{x,x'\})=2$

Very Few Polynomial Cases

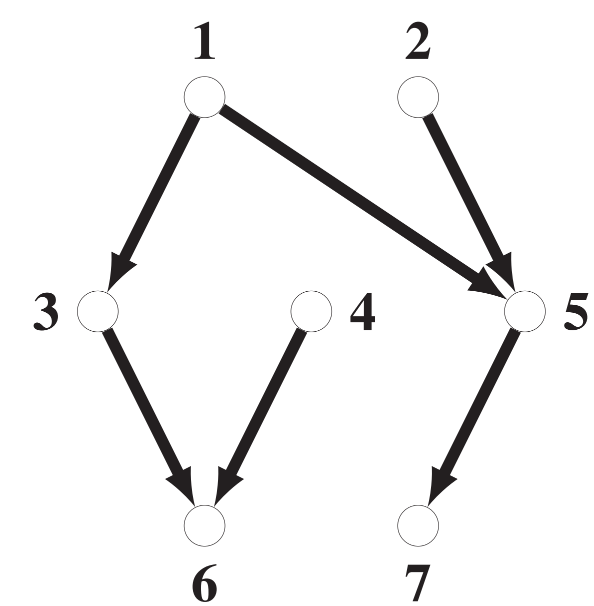
Theorem: An optimal *branching* can be found in polynomial time. (Chu and Liu 1965, Edmonds 1967)

Theorem: Finding an optimal *polytree* is NP-hard (Dasgupta 1999)

Can parameters make the problem tractable?



A branching



A polytree

Parameterized Complexity

Downey and Fellows: Parameterized Complexity Springer 1999

- Identify a parameter k of the problem
- Aim: solve the problem in time
  $f(k)n^c$
  fixed-parameter tractable (FPT)
- XP: solvable in time $n^{f(k)}$
- W-classes:
  $FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq XP$

Parameters for BN Structure Learning?

1. Parameterized by the treewidth (of the super structure)
2. Parameterized Local Search
3. Parameterized by distance from being a Branching

1.

BN Structure learning

parameterized by the treewidth

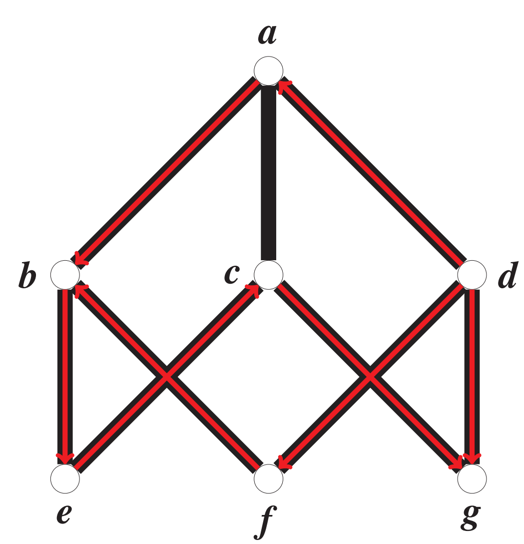
(of the super structure)

The Super Structure

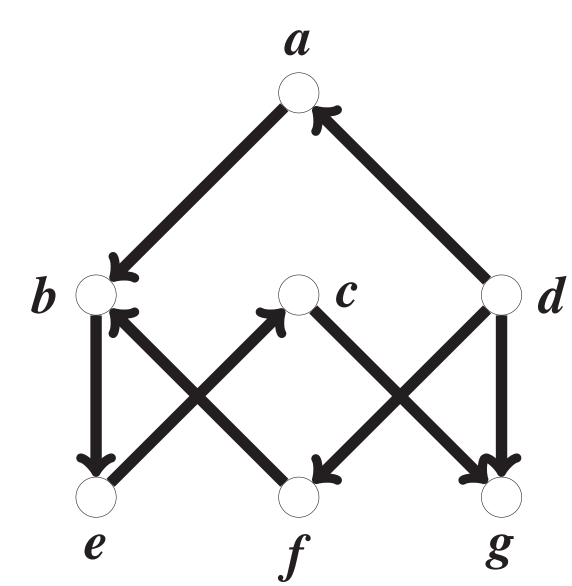
- *Super structure* S_f of a local score function f is the undirected graph S_f=(N,E) containing all edges that can participate in an optimal DAG [Perrier, Imoto, Miyano 2008]

n	P	$f(n,P)$
a	$\{d\}$	1
a	$\{b,c,d\}$	0.5
b	$\{a,f\}$	1
c	$\{e\}$	1
d	$\emptyset$	1
e	$\{b\}$	1
f	$\{d\}$	1
g	$\{c,d\}$	1

local score function



super structure



BN

Hardness

Theorem: BN Structure Learning is W[1]-hard when parameterized by the treewidth of the super structure.

Hardness

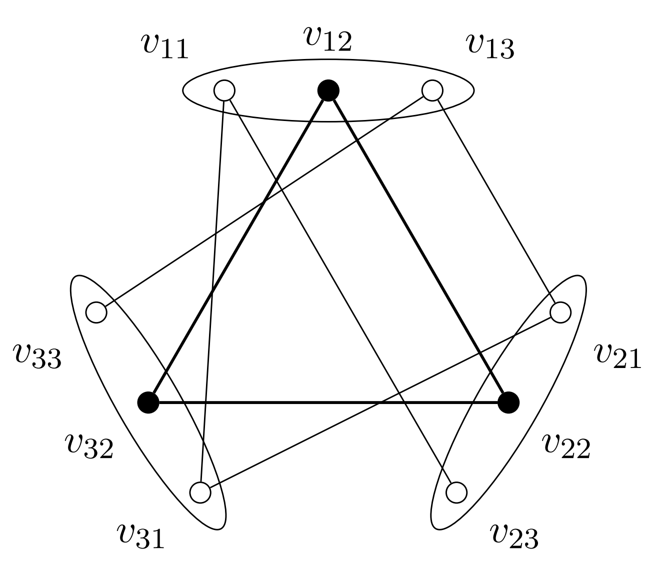
Theorem: BN Structure Learning is W[1]-hard when parameterized by the treewidth of the super structure.

Proof: by reduction from multi-colored clique (MCC).

Hardness

Theorem: BN Structure Learning is W[1]-hard when parameterized by the treewidth of the super structure.

Proof: by reduction from multi-colored clique (MCC).

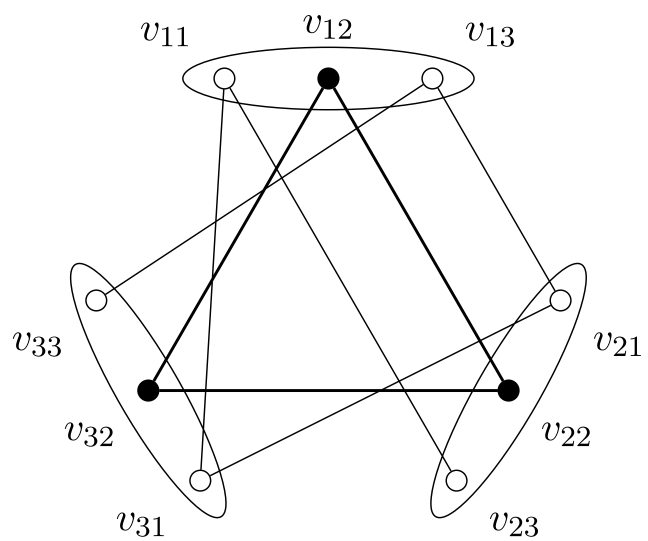


MCC instance, k=3

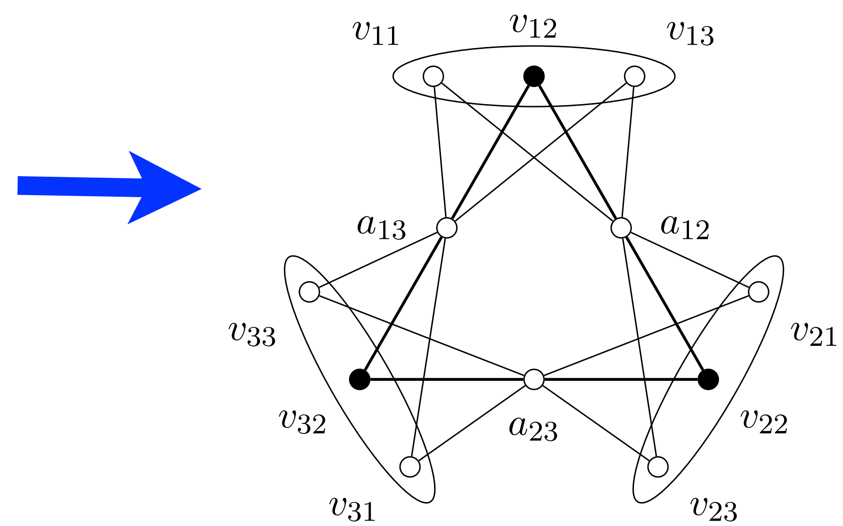
Hardness

Theorem: BN Structure Learning is W[1]-hard when parameterized by the treewidth of the super structure.

Proof: by reduction from multi-colored clique (MCC).



MCC instance, k=3

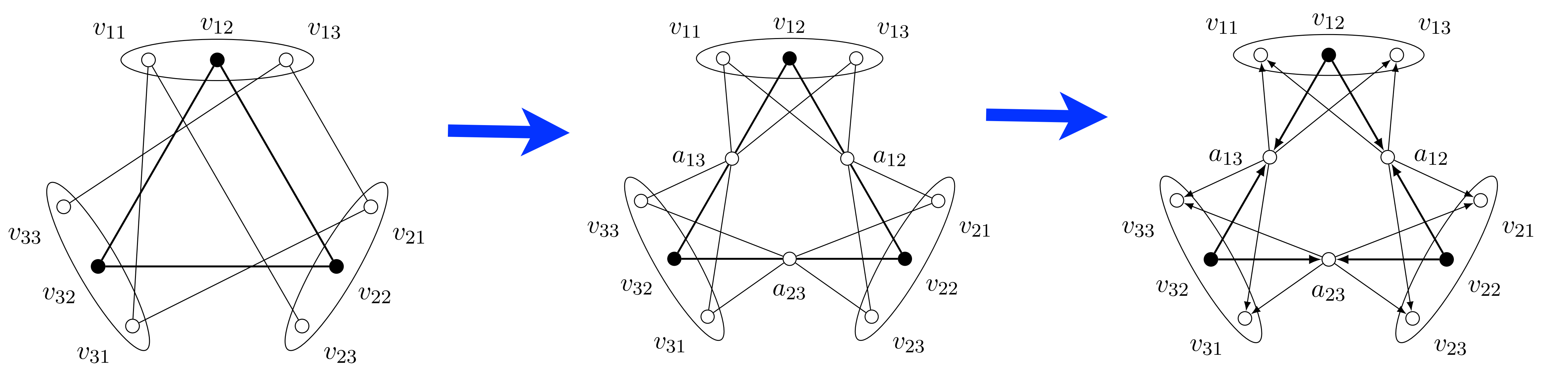


Super Structure

Hardness

Theorem: BN Structure Learning is W[1]-hard when parameterized by the treewidth of the super structure.

Proof: by reduction from multi-colored clique (MCC).



MCC instance, k=3

Super Structure

BN corresponding to clique

XP-tractability

Theorem: BN Structure Learning is in XP when parameterized by the treewidth of the super structure.

XP-tractability

Theorem: BN Structure Learning is in XP when parameterized by the treewidth of the super structure.

Proof: dynamic programming along a nice tree decomposition
For each node U of the tree decomp. we compute a table T(U).

T(U) gives the the highest possible score any partial solution (below U) can achieve

- for a given chosen parents of the nodes in the bag, and
- for a given reachability between nodes in the bag by directed paths.

XP-tractability

Theorem: BN Structure Learning is in XP when parameterized by the treewidth of the super structure.

Proof: dynamic programming along a nice tree decomposition
For each node U of the tree decomp. we compute a table T(U).

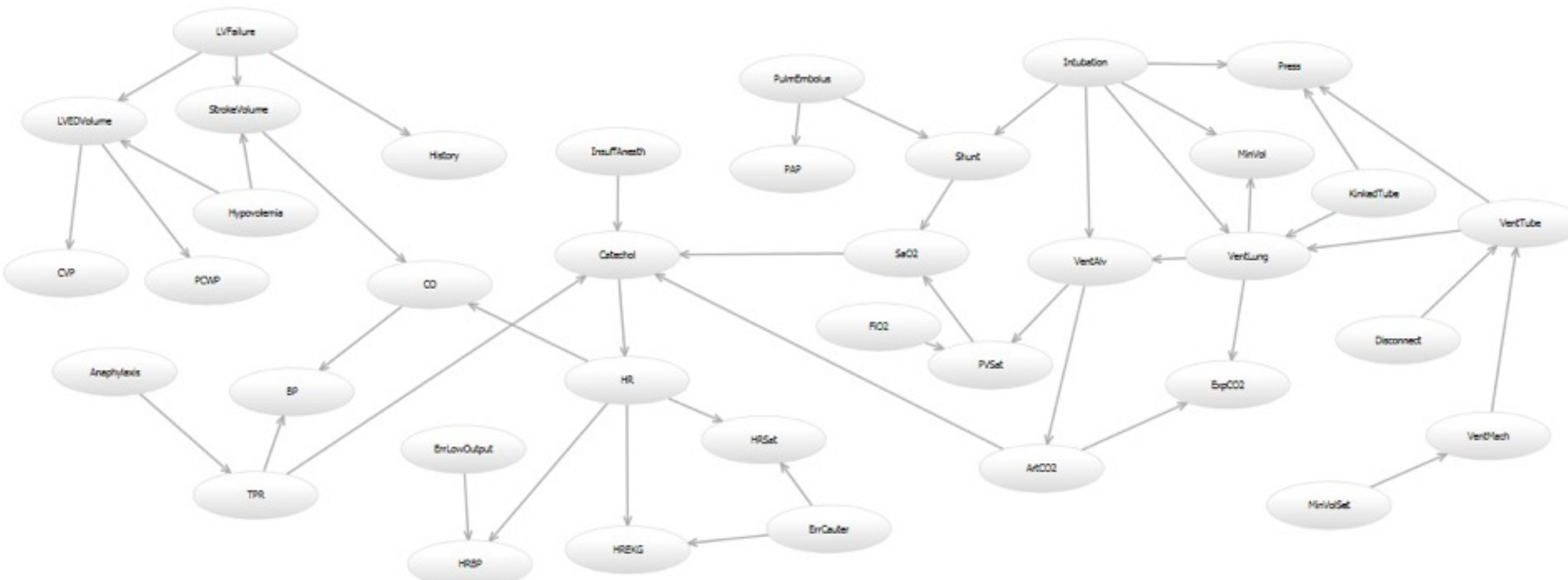
T(U) gives the the highest possible score any partial solution (below U) can achieve

- for a given chosen parents of the nodes in the bag, and
- for a given reachability between nodes in the bag by directed paths.

Corollary: If the super-structure has bounded degree, then we get fixed-parameter tractability

Experiments

- Alarm Network: 37 nodes



Experiments

- Bottleneck: huge tables:
- SharpLib for dynamic programming
- Dynamic Lower Bound approach (LB)
- Ongoing improvements

	Characteristics of S_f				running time (s)		memory usage (MB)	
S_f	$\|V(S_f)\|$	$\|E(S_f)\|$	$\Delta(S_f)$	$\mathrm{tw}(S_f)$	no LB	LB	no LB	LB
$\mathcal{S}_{\mathrm{SKEL}}$	37	46	6	3	14	7	337	180
$\mathcal{S}_{\mathrm{TIL}}(0.01)$	37	62	7	5	20785	6393	15679	5346
$\mathcal{S}_{\mathrm{TIL}}(0.05)$	37	63	7	5	46560	16712	38525	18786
$\mathcal{S}_{\mathrm{TIL}}(0.1)$	37	65	7	5	44554	16928	38520	18918

2.

BN Structure Learning

parameterized local search

Local Search

- Once we have a candidate BN, we can try to find a better one by local editing.
- We consider three local editing operations:
  - ADD - add an arc
  - DEL - delete an arc
  - REV - reversing an arc
- We parameterize by the number k of edits we can make in one step.

k-Local Search

- Parameterizing LS problems by the number of changes in one step has been suggested by Fellows 2003
- Since then a series of paper on that parameterized local search have been published, considering, among others: *TSP, Max-SAT, Max-Cut, Vertex Cover, Weighted SAT, Cycle Traversals, Stable Matchings*, etc.

Results

Theorem: Except for {ADD} and {DEL} all problems are W[1]-hard.

Hardness even holds for instances of bounded degree.

operations	complexity
ADD	PTIME
DEL	PTIME
REV	W[1]-h
ADD,DEL	W[1]-h
ADD,REV	W[1]-h
DEL,REV	W[1]-h
ADD,DEL,REV	W[1]-h

Results

Theorem: Except for {ADD} and {DEL} all problems are W[1]-hard.

Hardness even holds for instances of bounded degree.

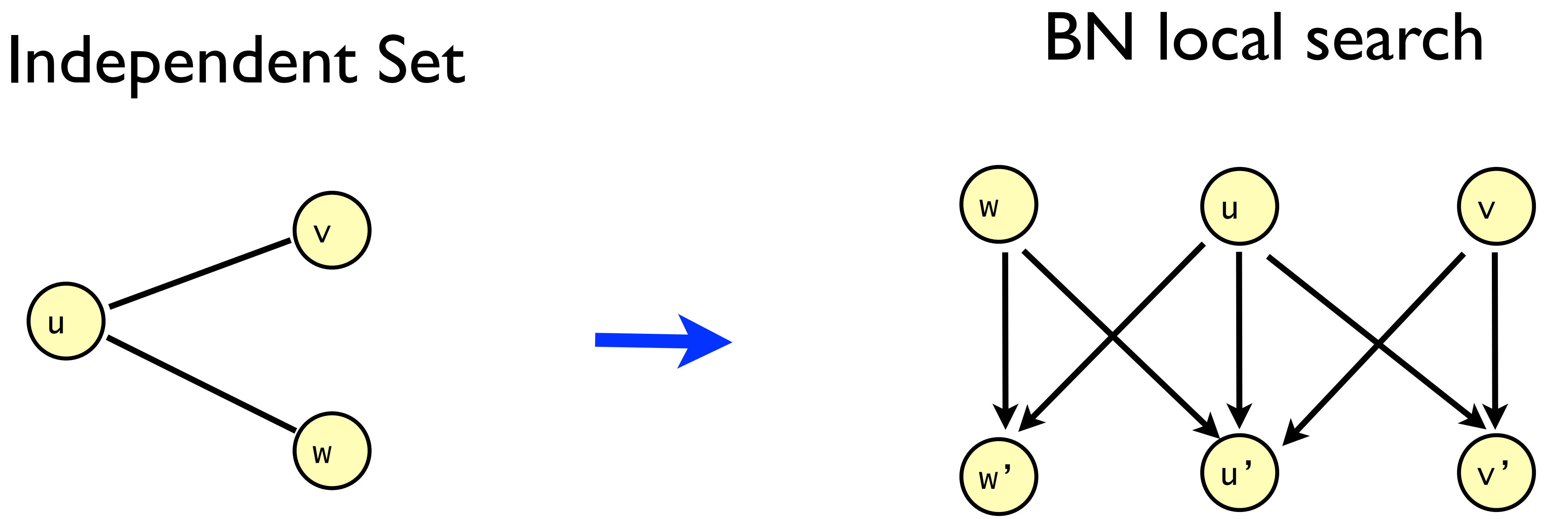
operations	complexity
ADD	PTIME
DEL	PTIME
REV	W[1]-h
ADD,DEL	W[1]-h
ADD,REV	W[1]-h
DEL,REV	W[1]-h
ADD,DEL,REV	W[1]-h

Does it help to bound the degree?

W[1]-hardness for bounded degree

- All known parameterized LS problems are FPT for instances of bounded degree.
- Almost all W[1]-hard problems are FPT for graphs of bounded degree.
- Red/Blue Nonblocker isn't an exception.
- We have now a new proof by reduction from Independent Set (we don't need the reduced graph to have bounded degree).

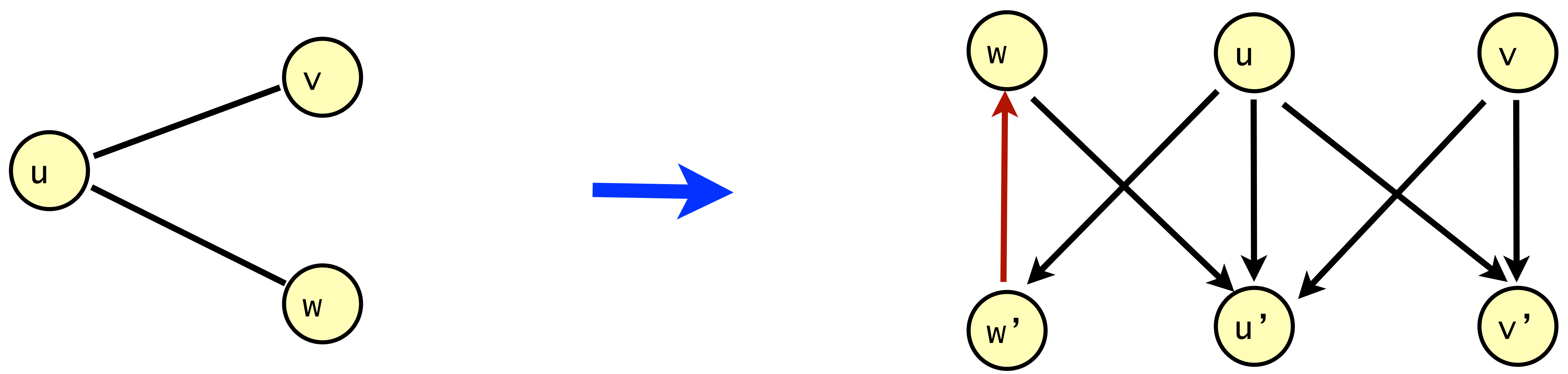
Reduction from IS for {REV}



Reduction from IS for {REV}

Independent Set

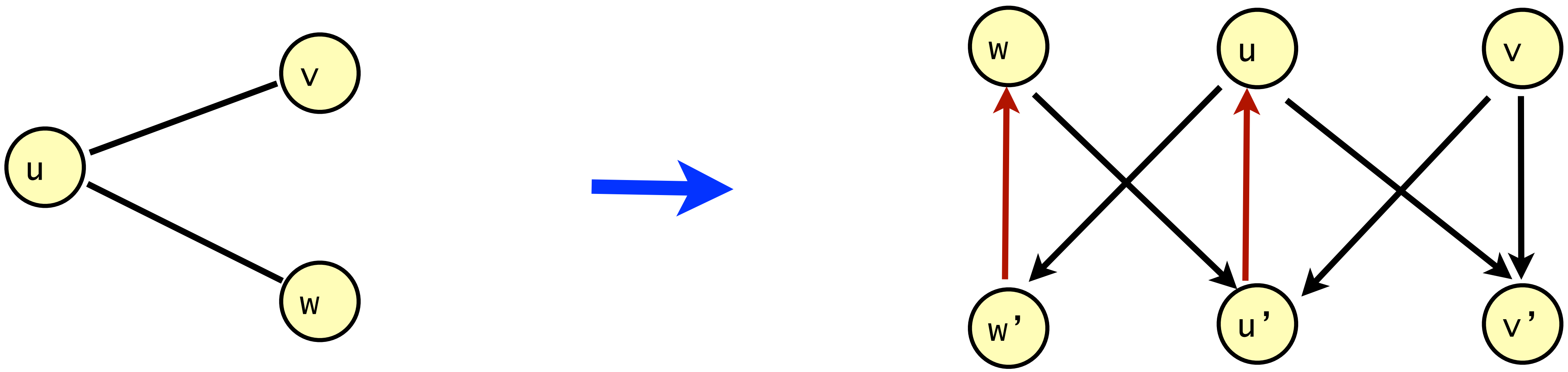
BN local search



Reduction from IS for {REV}

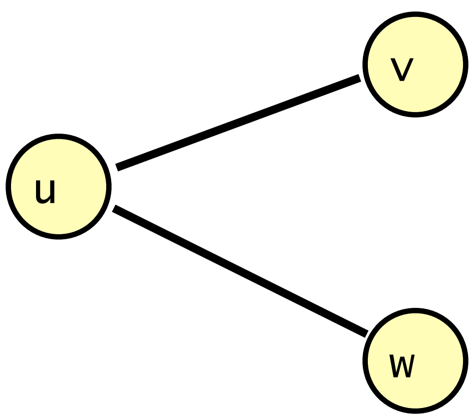
Independent Set

BN local search

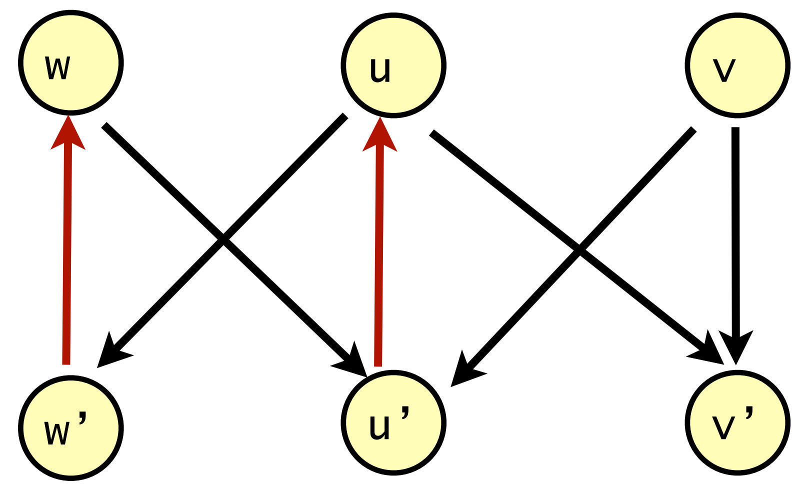


Reduction from IS for {REV}

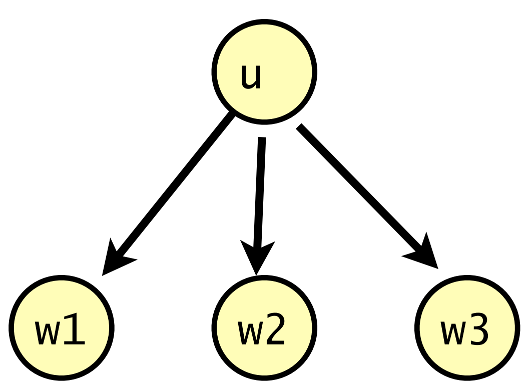
Independent Set

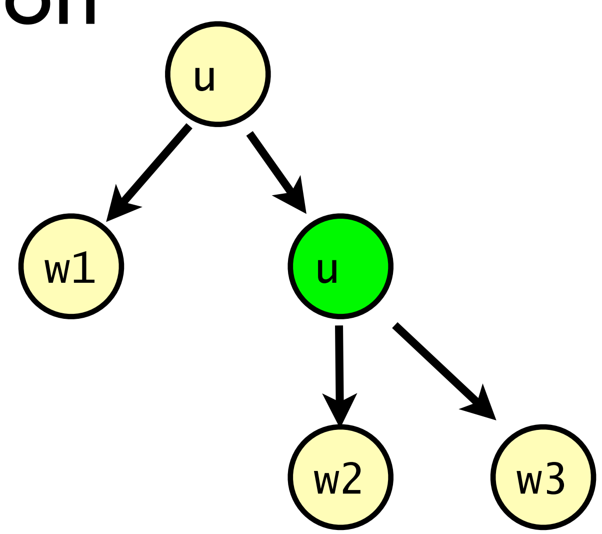


BN local search



degree reduction





3.

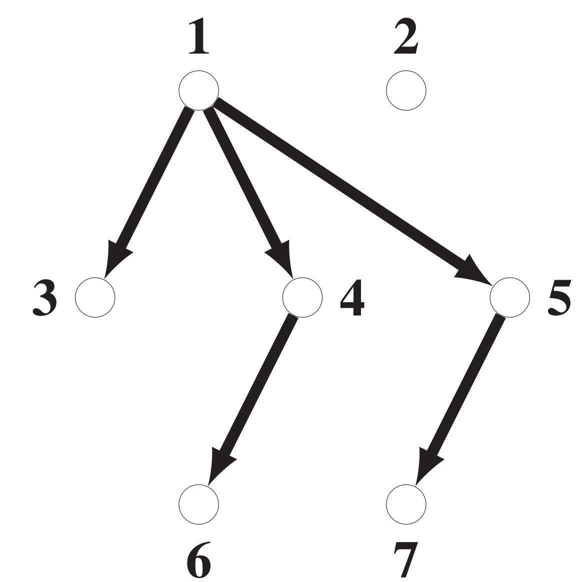
BN Structure Learning

parameterized by distance from being a branching

Beyond Branchings

Beyond Branchings

Theorem: An optimal *branching* can be found in polynomial time.
[Chu and Liu 1965, Edmonds 1967]

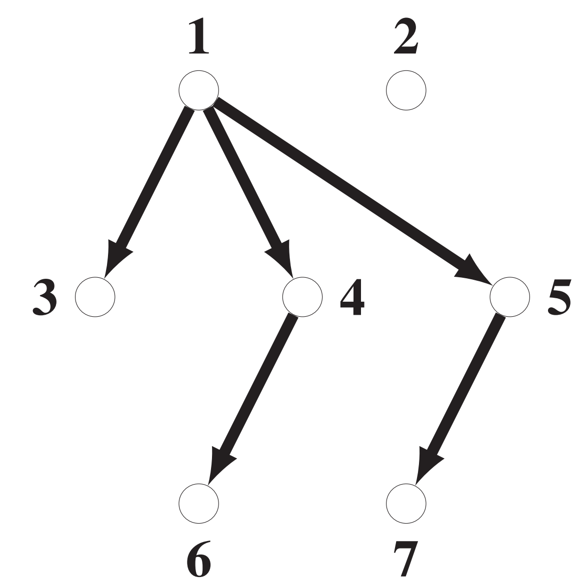


A branching

Beyond Branchings

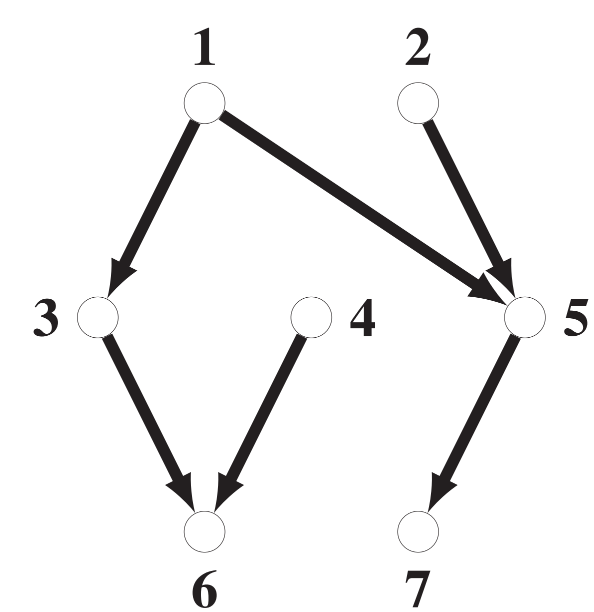
Theorem: An optimal *branching* can be found in polynomial time.

[Chu and Liu 1965, Edmonds 1967]



A branching

Theorem: Finding an optimal *polytree* is NP-hard [Dasgupta 1999]



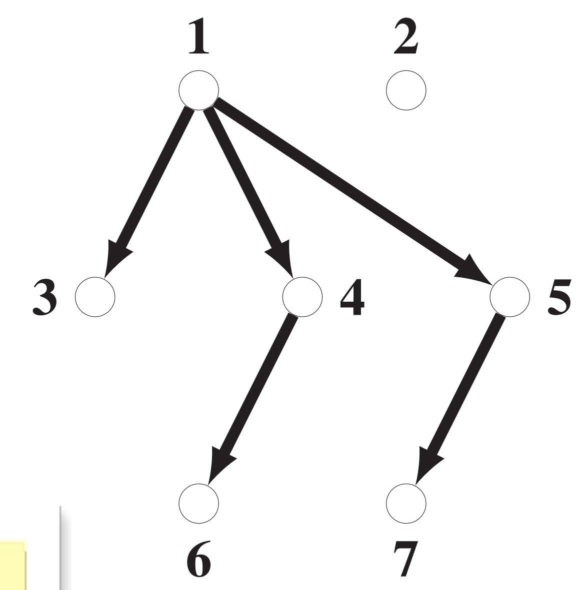
A polytree

Beyond Branchings

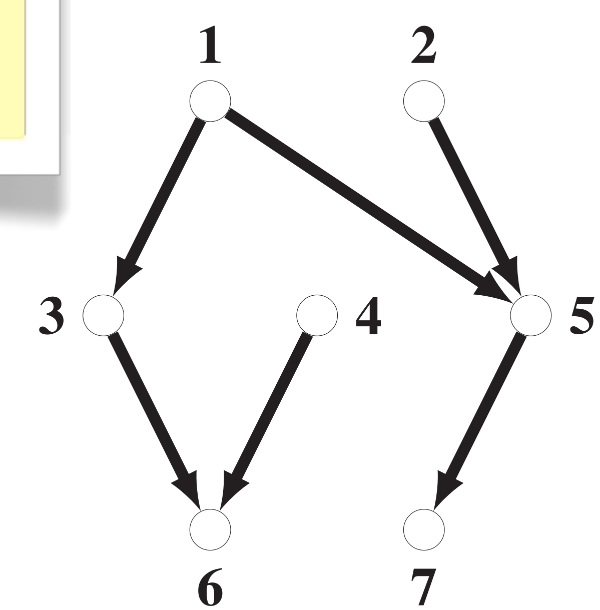
Theorem: An optimal *branching* can be found in polynomial time.
[Chu and Liu 1965, Edmonds 1967]

k-branching = polytree such that after deleting certain k arcs, every node has at most one parent.

Theorem: Finding an optimal *polytree* is NP-hard [Dasgupta 1999]



A branching



A polytree

XP-Result

Theorem: finding an optimal k-branching is polynomial for every constant k.

(i.e., the problem is in XP for parameter k.)

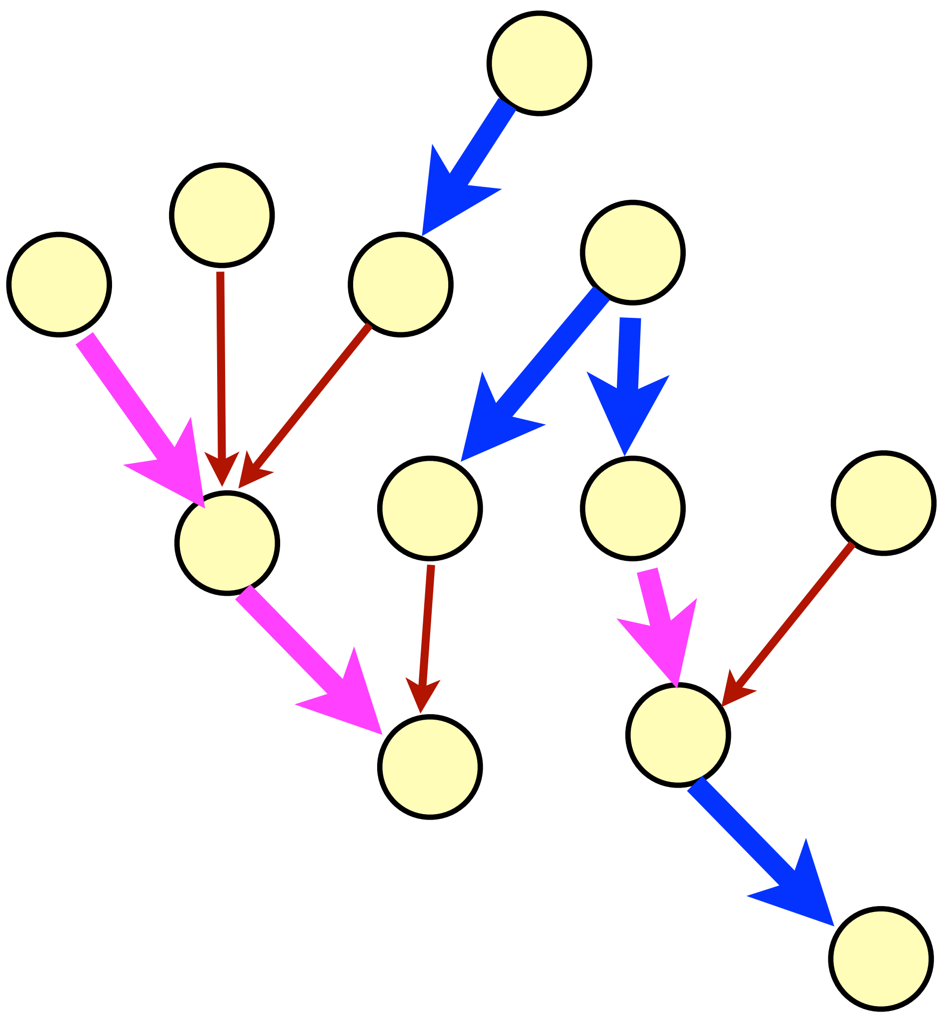
XP-Result

Theorem: finding an optimal k-branching is polynomial for every constant k.

(i.e., the problem is in XP for parameter k.)

Approach:
(1) guess the deleted arcs
(2) solve the induced problem for branchings
turns out to be quite tricky!

A k-branching (k=4)



+ + = D *k-branching*

= X *deletion set*

= X‘ *co-set*

= R *rest*

+ = R∪X’=D-X polytree with in-deg ≤ 1

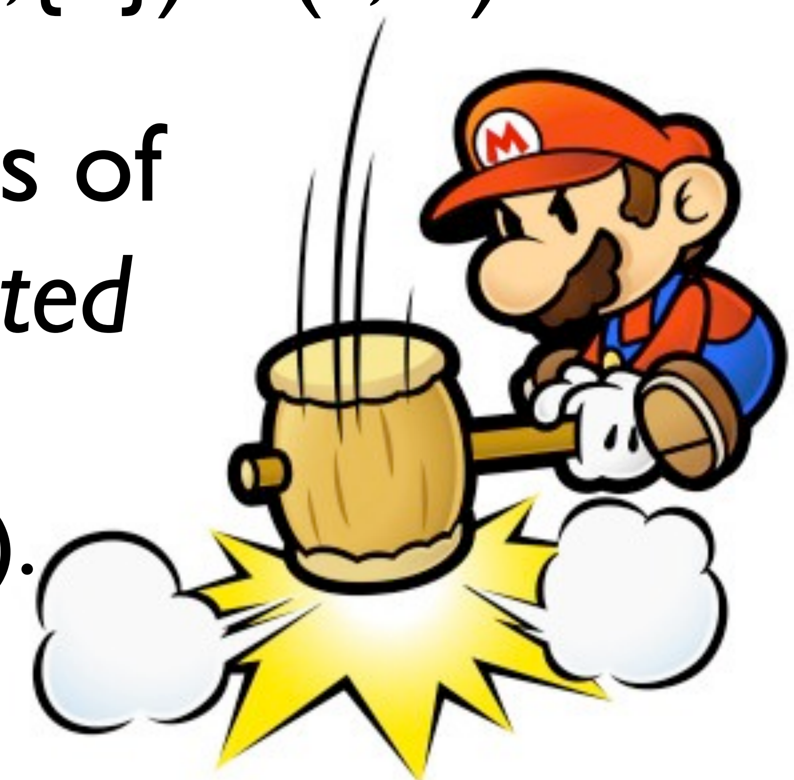
|X|≤k and |X’|≤k. Hence for constant k we can try all X and X‘ in polynomial time. For each choice of X∪X’ we compute the best set R such that R∪X’ is a polytree with in-deg ≤ 1.

Matroid Approach

- Let X, X' be two disjoint sets of arcs.
- We define define two matroids over N×N.
- The **in-degree matroid** where a set R is independent iff $R \cap (X \cup X') = \varnothing$ and R has indegree ≤ 1.
- The **acyclicity matroid** where a set R is independent iff $R \cup X \cup X'$ has no undirected cycles.
- *Lemma: D is a k-branching iff D-X is independent in both matroids.*

Matroid Intersection

- Hence, for each guessed set X we need to find an optimal set D that is independent in both matroids.
- Optimality can be expressed by considering weighted matroids: $w(u,v)= f(v,\{u\})-f(v,\emptyset)$.
- Solution can be found by means of a poly time algorithm for *weighted matroid intersection* (Brezovec, Cornuejols, Glover 1986).



FPT?

- We don’t know whether the k-branching problem is FPT.
- We have FPT algorithms for special cases whenever the guess-phase is FPT.
- For instance: *the arcs in* $X \cup X'$ *form an in-tree and each node has a bounded number of potential parent sets.*
- BN structure learning remains NP-hard under this restriction.

A slight generalization is W[1]-hard

Define a *k-node branching* as a polytree that becomes a branching by deletion of k *nodes*.

A slight generalization is W[1]-hard

Define a *k-node branching* as a polytree that becomes a branching by deletion of k *nodes*.

> Theorem: Finding an optimal k-node branching is W[1]-hard.

A slight generalization is W[1]-hard

Define a *k-node branching* as a polytree that becomes a branching by deletion of k *nodes*.

> Theorem: Finding an optimal k-node branching is W[1]-hard.

Proof: reduction from multicolored clique.

Conclusion

- BN Network Structure Learning
- important and notoriously difficult
- parameterized complexity approach:
    - *treewidth of super structure*
    - *parameterized local search*
    - *k-branchings and k-node branchings*
- Other parameters?