Improving distributed reasoning with privacy using tree decomposition

Vincent Armant

LAMIA Seminar

Insight

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Outline

- Introduction
  - A distributed Reasoning Problem
  - Graphical Tree Decomposition

- Distributed tree decomposition
  Preserve network structure
  Keep local information local

- Centralized tree decomp. VS concurrent approaches

- ***Token elimination***

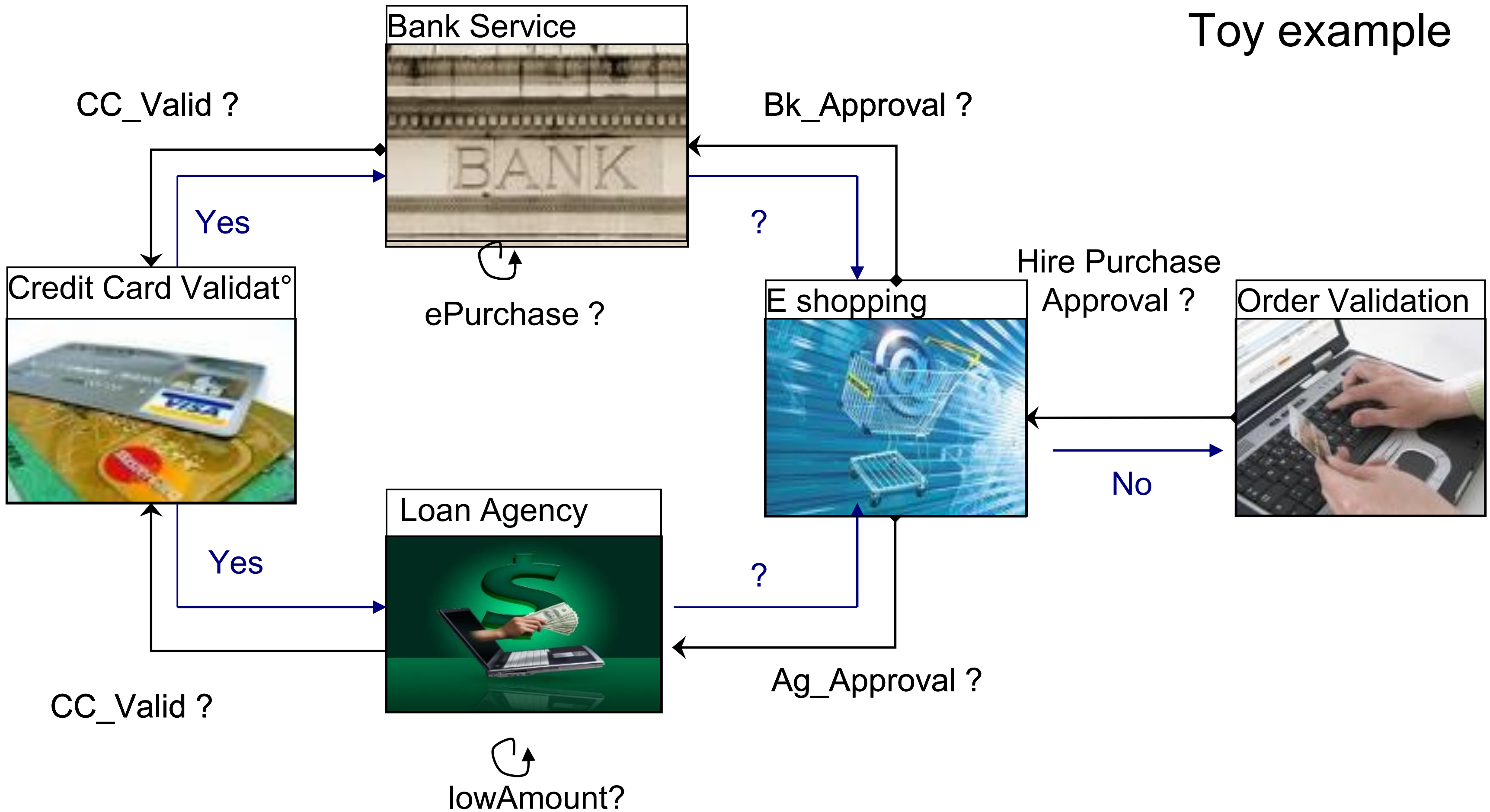
- Experimental results on small-world graphs

- Conclusion / perspectives

Introduction

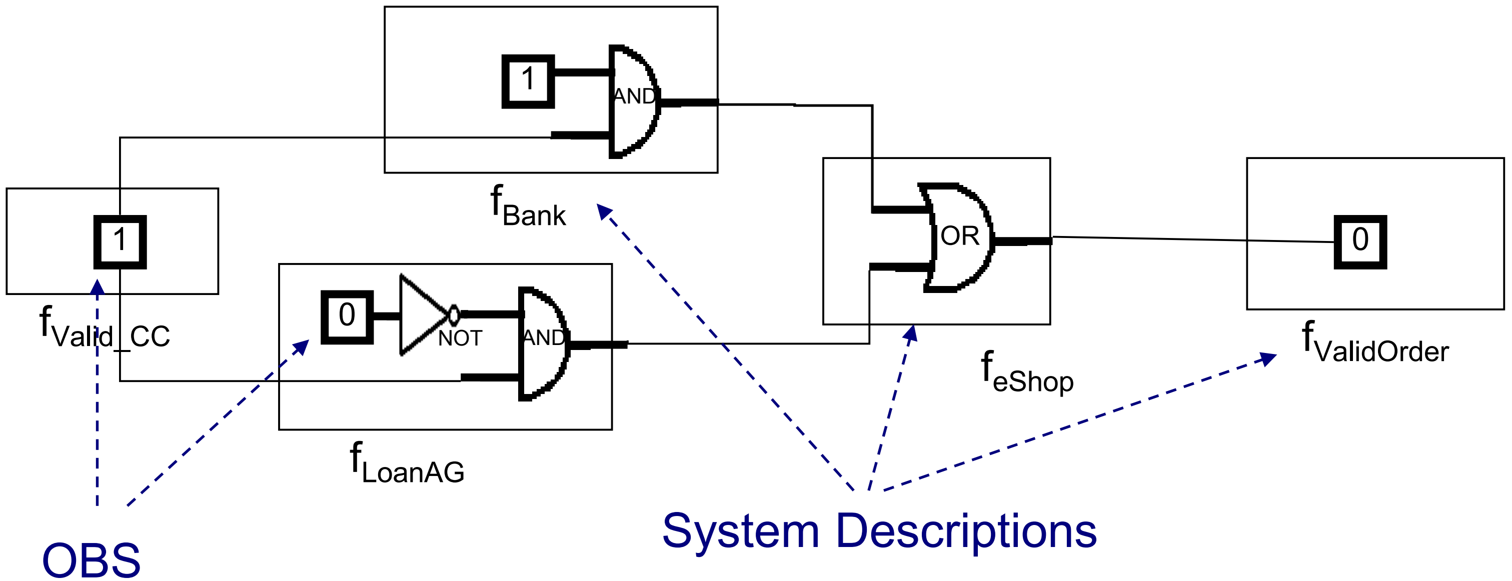
Three times web-payment certification

Toy example



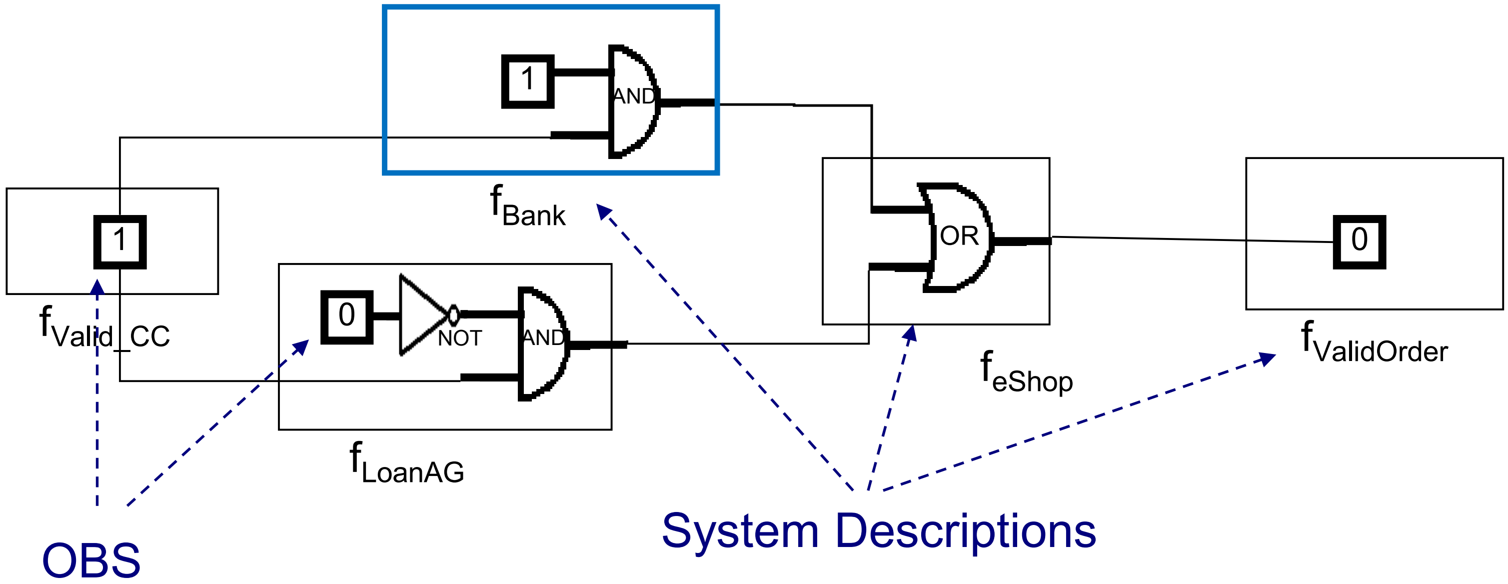
Introduction

Modeling the behaviors



Introduction

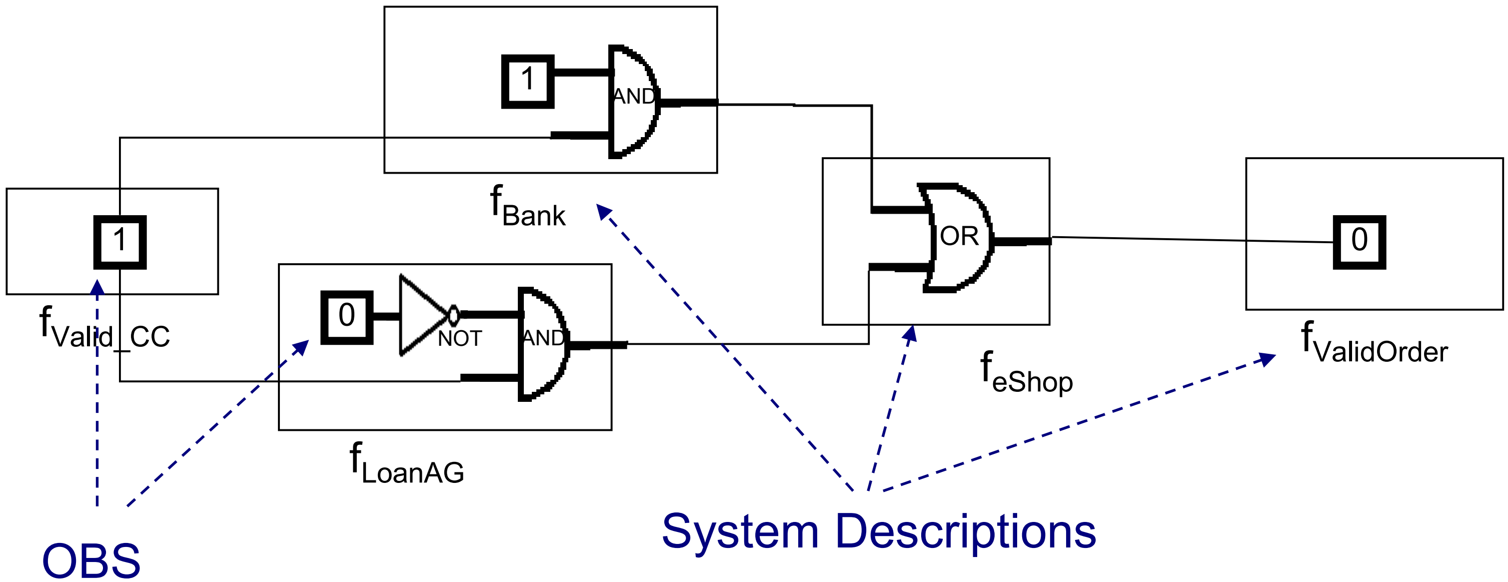
Modeling the behaviors



f_{Bank} : $\neg ab(Bank) \Rightarrow ((Cb\_valid \wedge \neg e\_purchase) \Leftrightarrow bk\_agreemt)$

Variables: Mode Shared Local

Modeling the behaviors



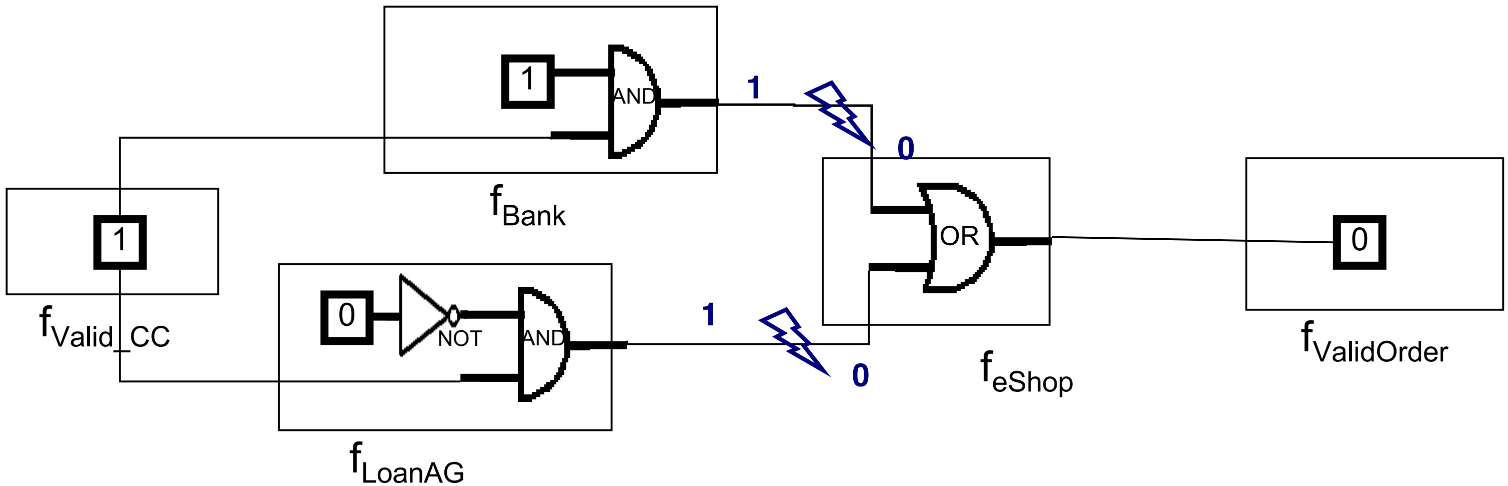
f_{Bank} : $\neg ab(Bank) \Rightarrow ((Cb\_valid \wedge \neg e\_purchase) \Leftrightarrow bk\_agreemt)$

Variables: Mode Shared Local

Global model : set of observations and local system descriptions

$$f_{global} = \bigwedge_i f_i \bigwedge_i OBS$$

Minimal conflicts



Minimal Conflict :

are components that are together inconsistent with observations

$\wedge f_i \wedge OBS_i \models C$ s.t. $\forall$ C' conflict, if C' $\Rightarrow$ C then C' = C

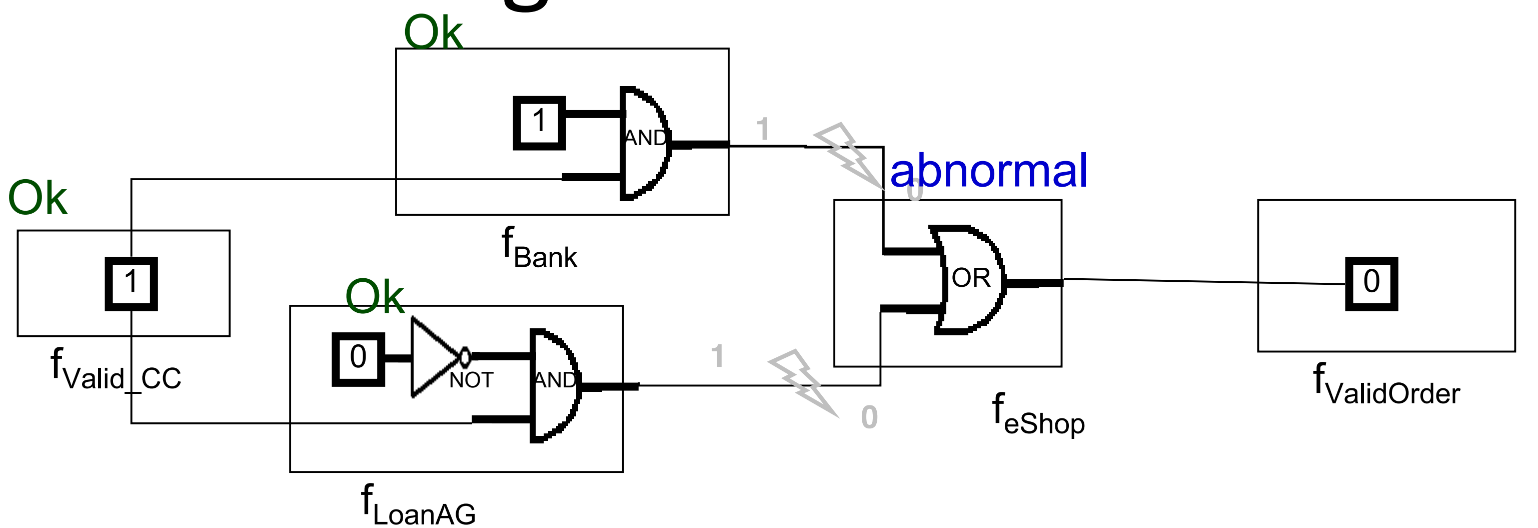
C $\subseteq$ AB, AB = {ab1, …,abn }

Example:

ab(Bank) $\vee$ ab(eShop)

ab(LoanAg) $\vee$ ab(eShop) $\wedge$

Minimal diagnoses



Minimal Diagnosis Δ :

Is a minimal explanation which cover **all** minimal conflicts

$\wedge f_i \wedge OBS_i \wedge \Delta \wedge \overline{AB\backslash\Delta} \not\models \bot$ s.t. $\forall$ Δ' diagnosis, if $\Delta' \Rightarrow \Delta$ then $\Delta' = \Delta$

$\Delta \subseteq F$, F = {ab1, …,abn }

Example:

ab(Bank) $\wedge$ ab(LoanAg)

ab(eShop) $\vee$

Challenge of distributed Reasoning

- Context : Distributed Algorithm
  - Each peer performs the same algorithm
  - A peer only know :
    - Its acquaintance
    - Its own description
  - A peer do not want to share some private knowledge
    - But must share any local knowledge that is “interesting” for the task
  - The network incrementally returns solutions (i.e. diagnoses)

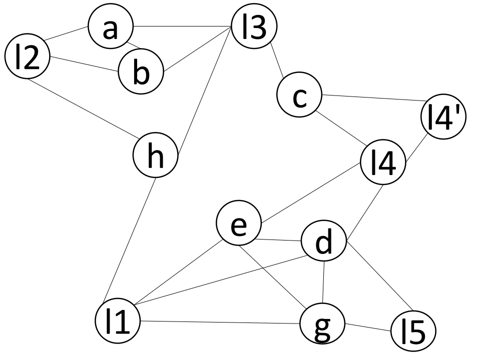
How to solve efficiently a distributed reasoning problem ?

Introduction

Primal graph

Pb :

f_1(l1, h) ∧
f_2(l1, d , e, g) ∧
f_3(l2, a, b) ∧
f_4(l2, h) ∧
f_5(l3, a, b) ∧
f_6(l3, c) ∧
f_7(l3, h) ∧
f_8(l4, l4', c) ∧
f_9(l4, e, d) ∧
f_{10}(l5, d , g)



centralized problem description

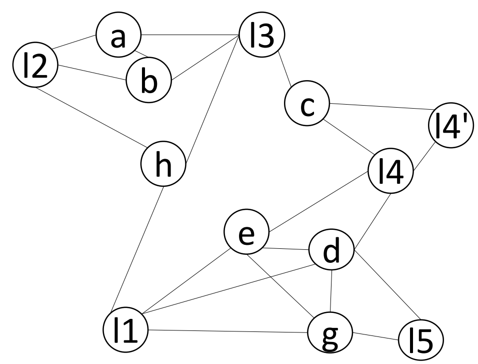
Its primal graph

Introduction

Primal graph

Pb :

$f_1(l1, h)$ ∧
$f_2(l1, d, e, g)$ ∧
$f_3(l2, a, b)$ ∧
$f_4(l2, h)$ ∧
$f_5(l3, a, b)$ ∧
$f_6(l3, c)$ ∧
$f_7(l3, h)$ ∧
$f_8(l4, l4', c)$ ∧
$f_9(l4, e, d)$ ∧
$f_{10}(l5, d, g)$



centralized problem description

Its primal graph

Generalization

Pb = join of databases relation (Primal Graph ~ Data Base Schema)

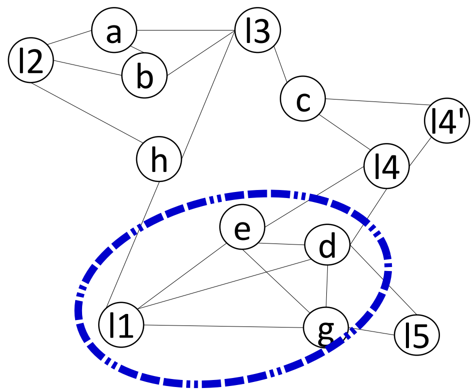
"A new approach to database logic Kuper,Vardi 1984"

Pb = Bayesian Inference (Primal Graph ~ Variable dependencies)

Primal graph

Pb :

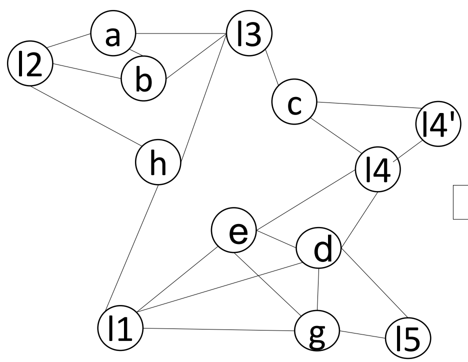
f_1(l1, h) ∧
f_2(l1, d , e, g) ∧
f_3(l2, a, b) ∧
f_4(l2, h) ∧
f_5(l3, a, b) ∧
f_6(l3, c) ∧
f_7(l3, h) ∧
f_8(l4, l4', c) ∧
f_9(l4, e, d) ∧
f_{10}(l5, d , g)



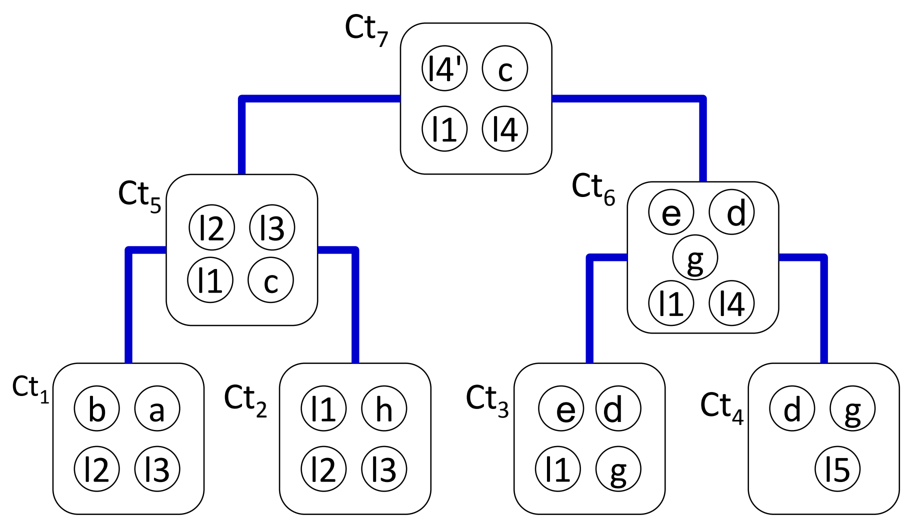
- Each variable labels exactly one node
- All variables contained in the scope of a formula in the problem description are neighbors in the primal graph

Introduction

Tree Decomposition



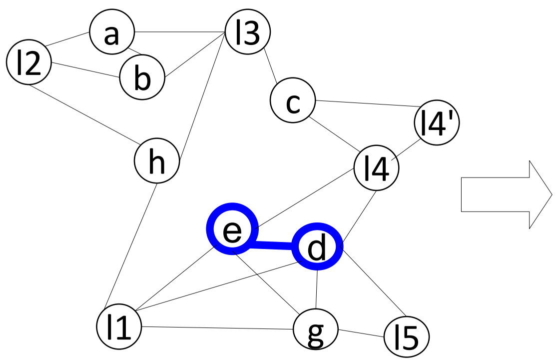
Primal graph

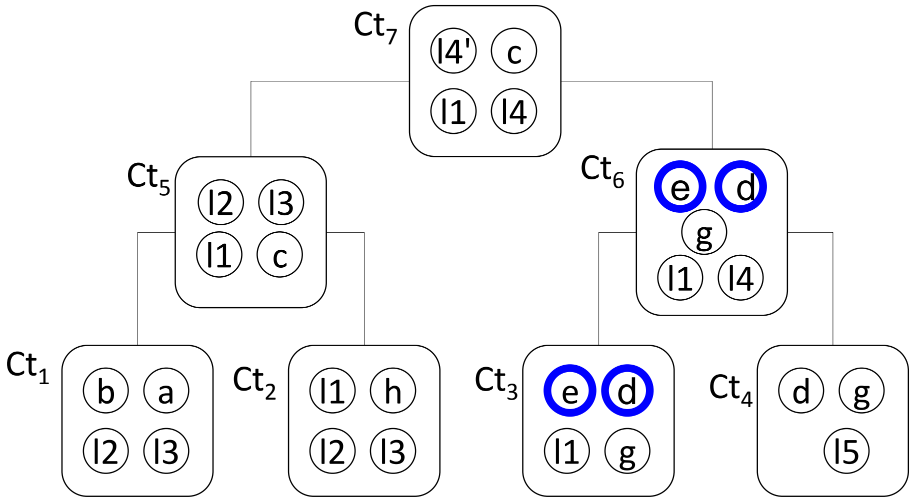


A tree decomposition

1) is a tree of clusters
2) preserves variables dependency
3) ensures running intersection

Tree Decomposition





Primal graph

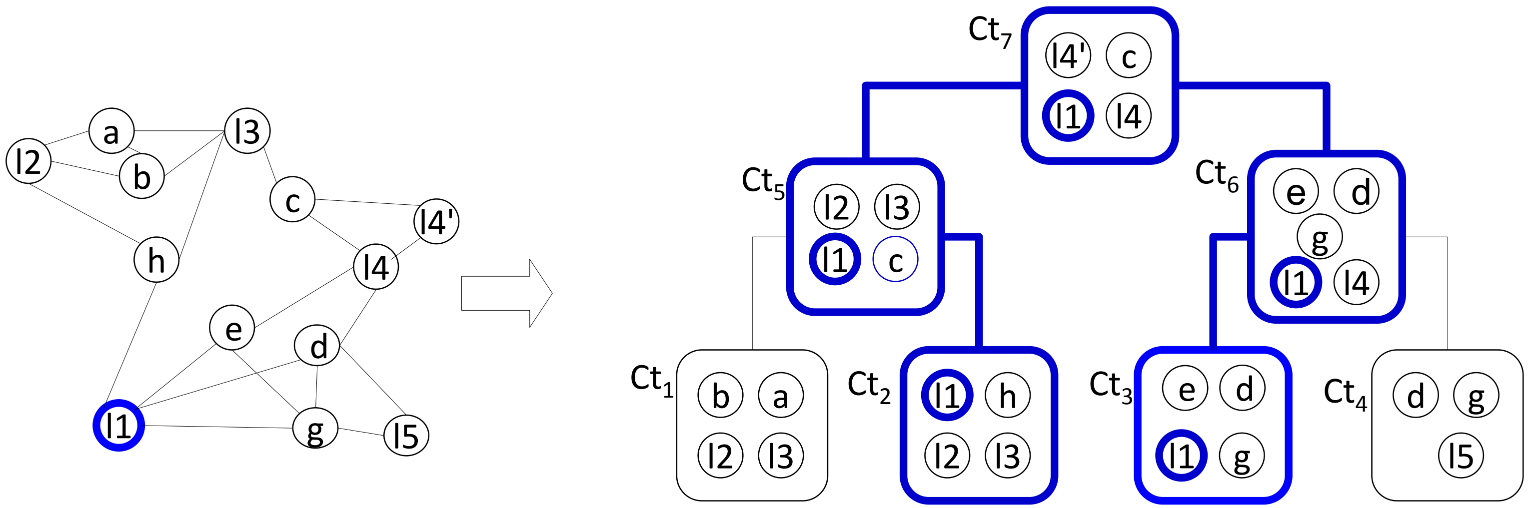
A tree decomposition

1) is a tree of clusters

2) preserves variables dependency

3) ensures running intersection

Tree Decomposition



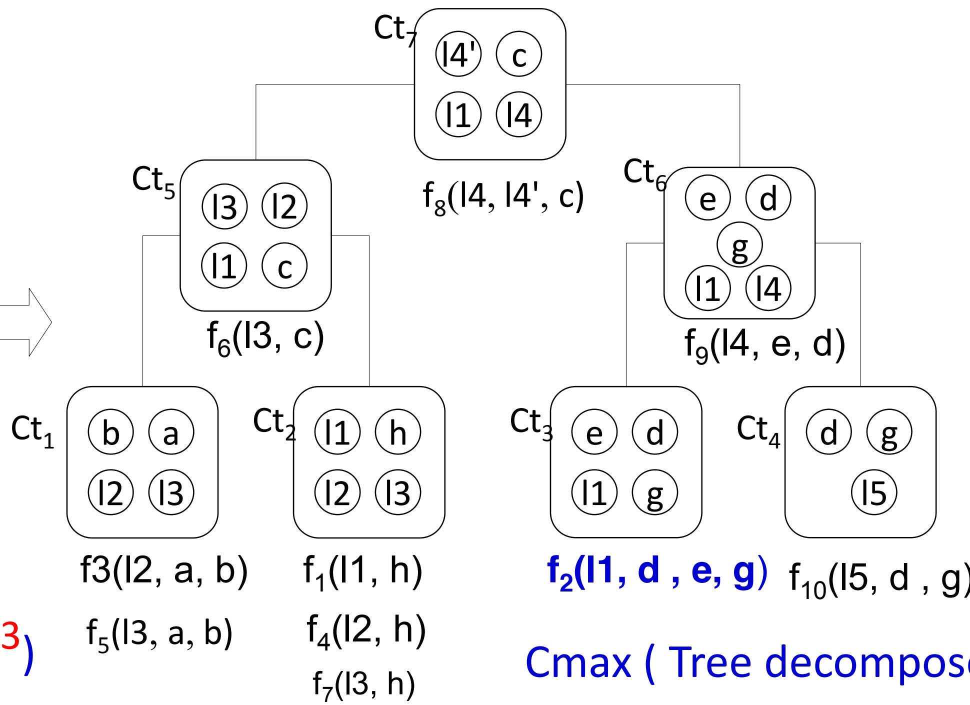
Primal graph

A tree decomposition
1) is a tree of clusters
2) preserves variables dependency
3) ensures running intersection

Why is it useful ?

Pb :

$f_1(l1, h) \wedge$
$f_2(l1, d, e, g)$ $\wedge$
$f_3(l2, a, b) \wedge$
$f_4(l2, h) \wedge$
$f_5(l3, a, b) \wedge$
$f_6(l3, c) \wedge$
$f_7(l3, h) \wedge$
$f_8(l4, l4', c) \wedge$
$f_9(l4, e, d) \wedge$
$f_{10}(l5, d, g)$



Cmax(initial pb)= $O(d^{13})$

Cmax (Tree decomposed pb) = $O(d^4)$

1) Good points:
   - divides the initial problem into sub-problems organized in a tree structure
   - allows concurrent resolution and /or backtrack free search
   - bounds time and space complexity by the size of the largest cluster (width)
     e.g. allows succinct representation (OBDD, MDD, DNNF, ..)

2) Limitations:
   - finding an optimal tree-decomposition is NP-Hard

Outline

- Introduction

- Distributed tree decomposition
  Preserve network structure
  Keep local information local

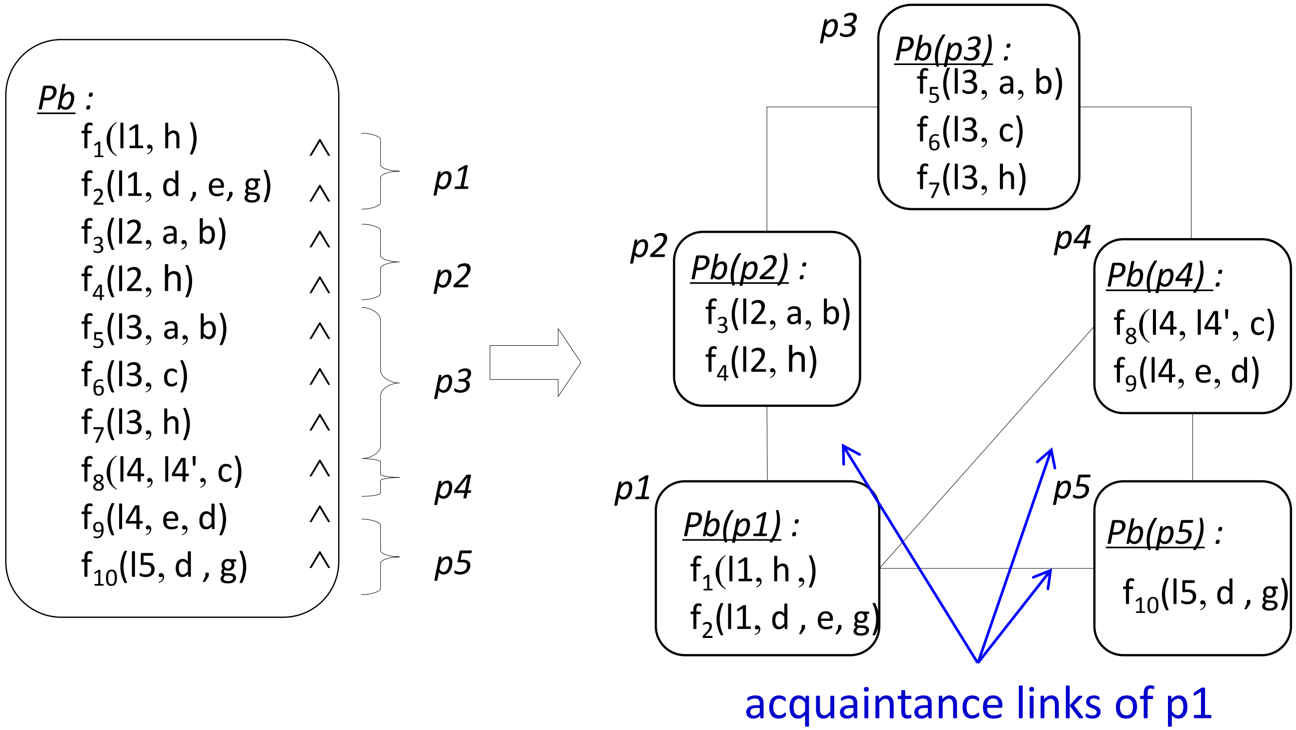
- Centralized tree decomp. VS concurrent approaches

- ***Token elimination***

- Experimental results on small-world graphs

- Conclusion / perspectives

Distributed system

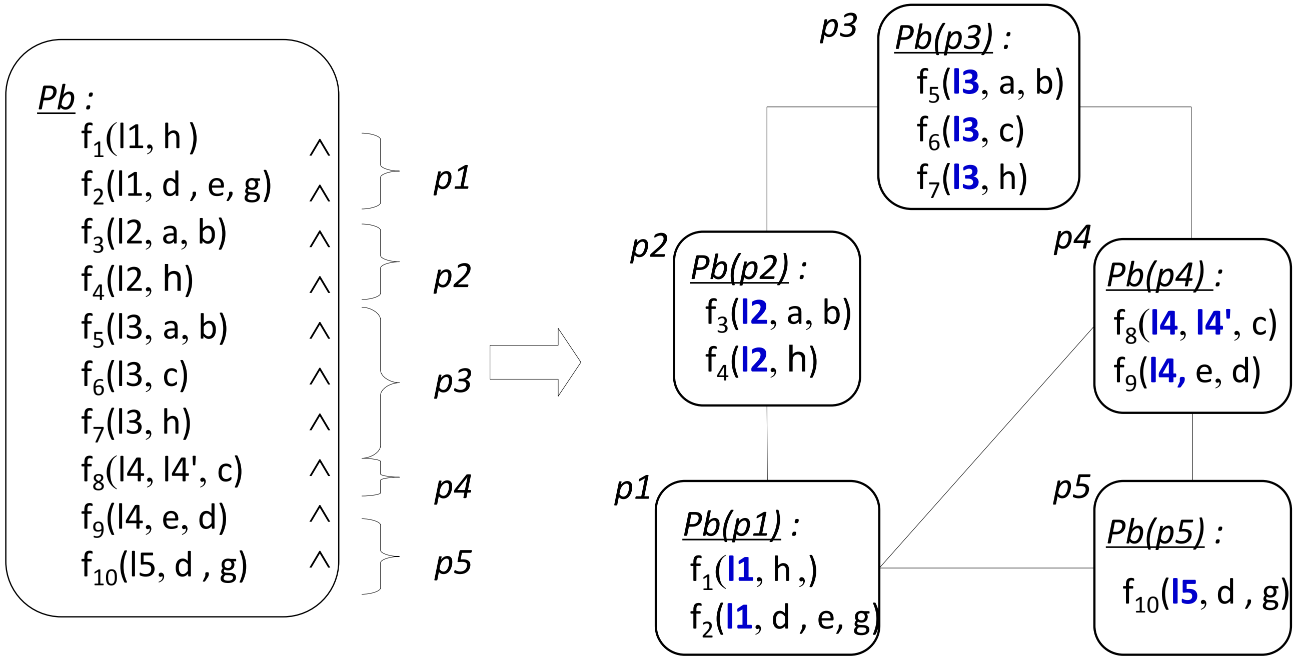


Initial problem setting is distributed among a set of peers

1) each peer can only interact with its neighbors by acquaintance links

2) local variables remain local

Distributed system



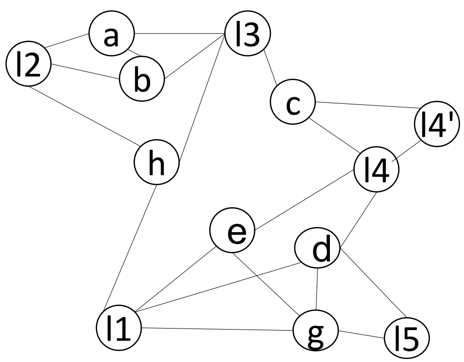
each « **li** » represents a local variable of pi

Initial setting is distributed among a set of peers

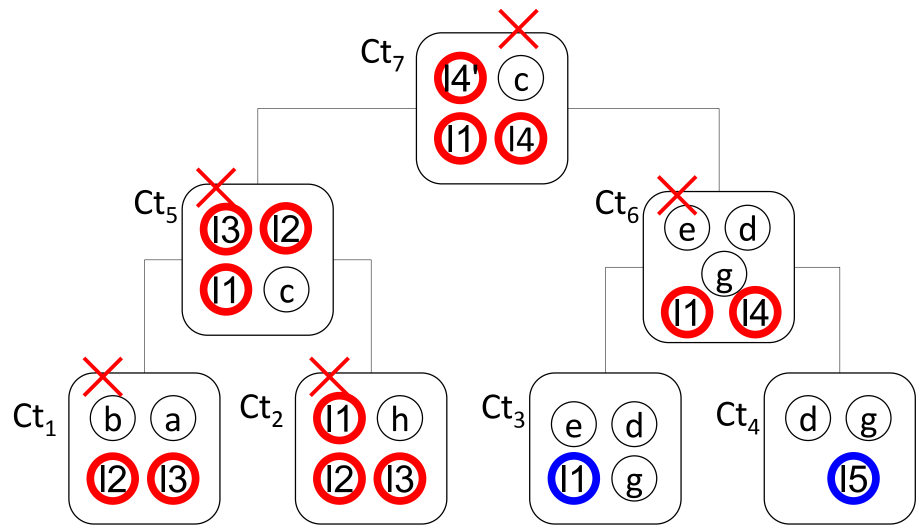
1) each peer can only interact with neighbors by acquaintance links

2) local variables remain local

Problematic: How to decompose a distributed system respecting privacy and the peer acquaintances ?



a primal graph



its tree decomposition

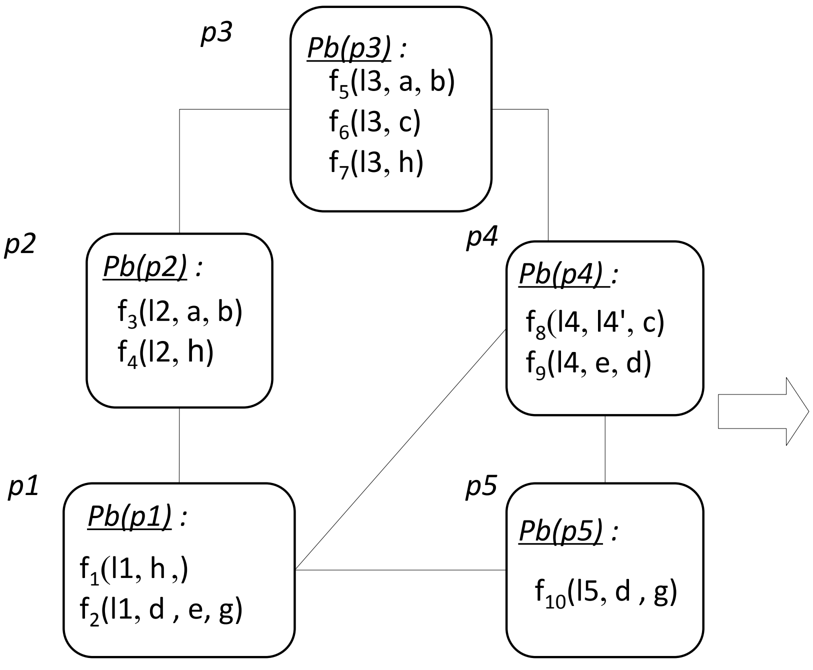
The classical notion of tree decomposition is not sufficient

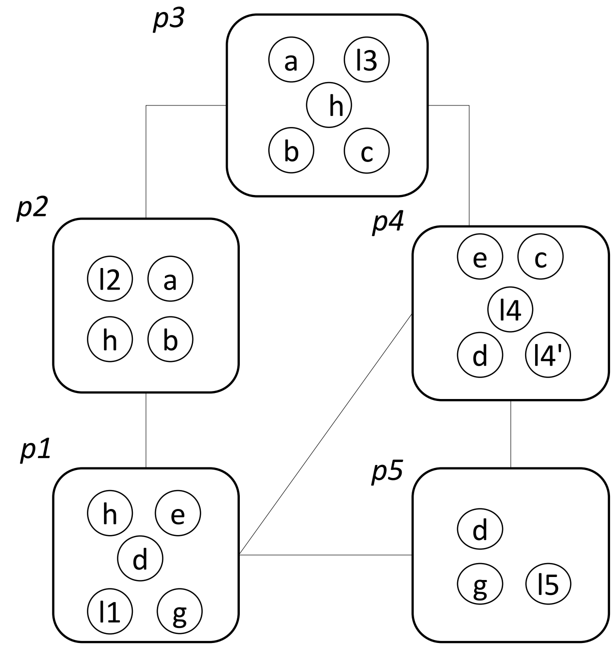
it does not respect the privacy of local variables

it does not preserve the peer acquaintances

Distributed Tree Decomposition

Acquaintance Graph



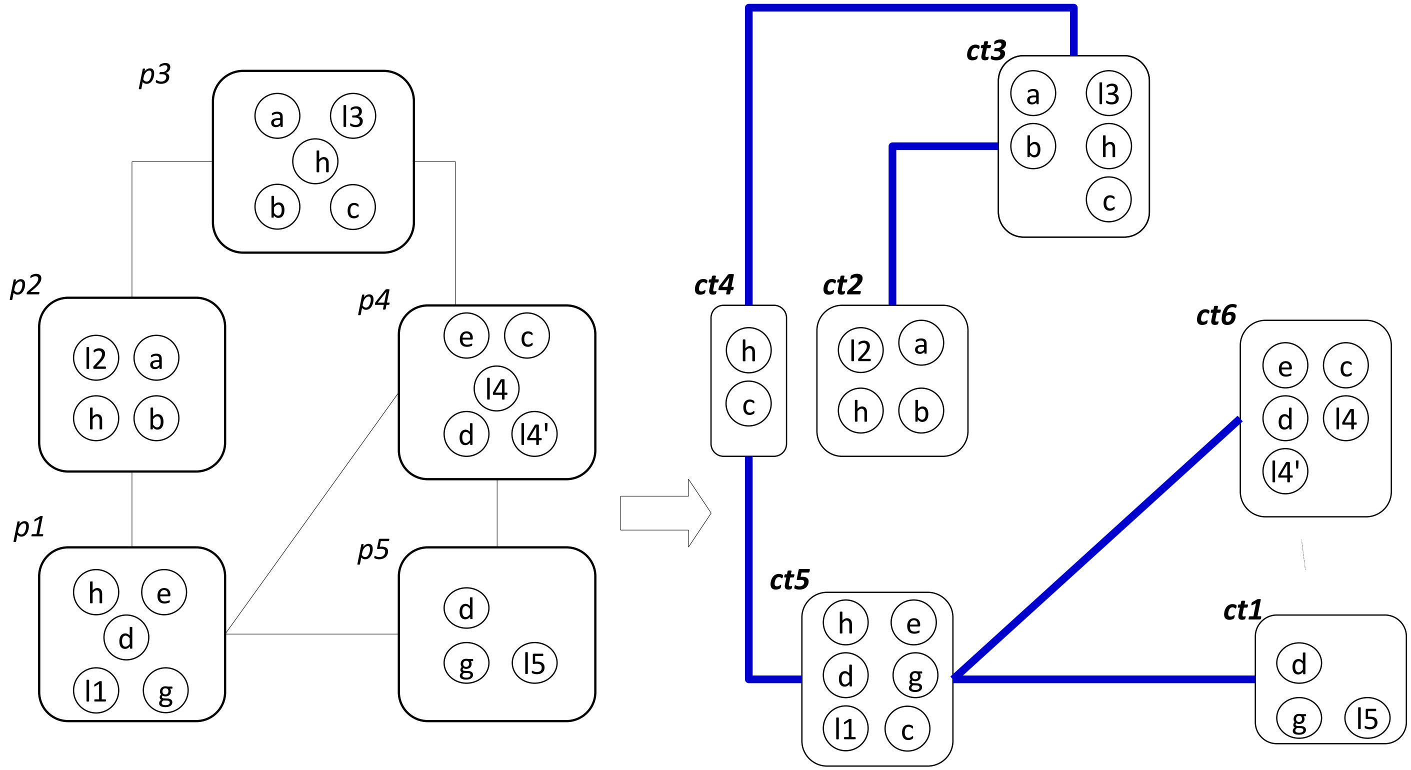


Distributed system

Acquaintance Graph G((P,V), ACQ)

1) P represents the set of peers
2) V labels each peer by its set of variables
3) ACQ $\subseteq$ P x P represents is acquaintance links

Distributed Tree Decomposition

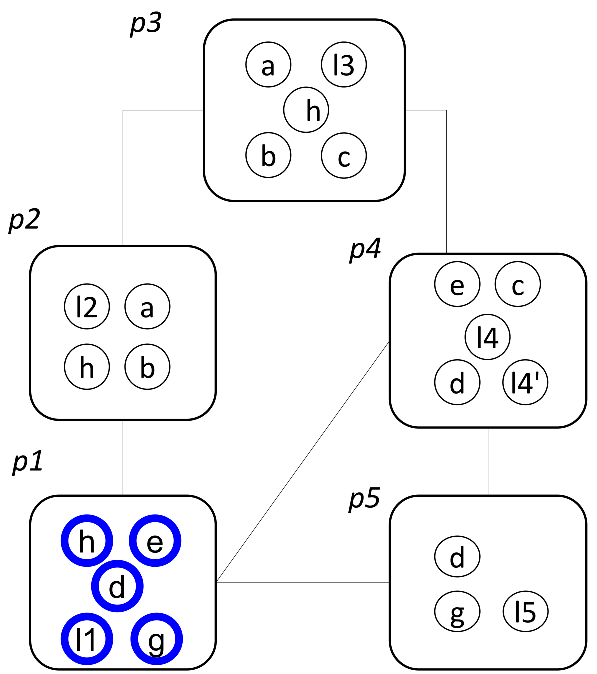


Acquaintance Graph

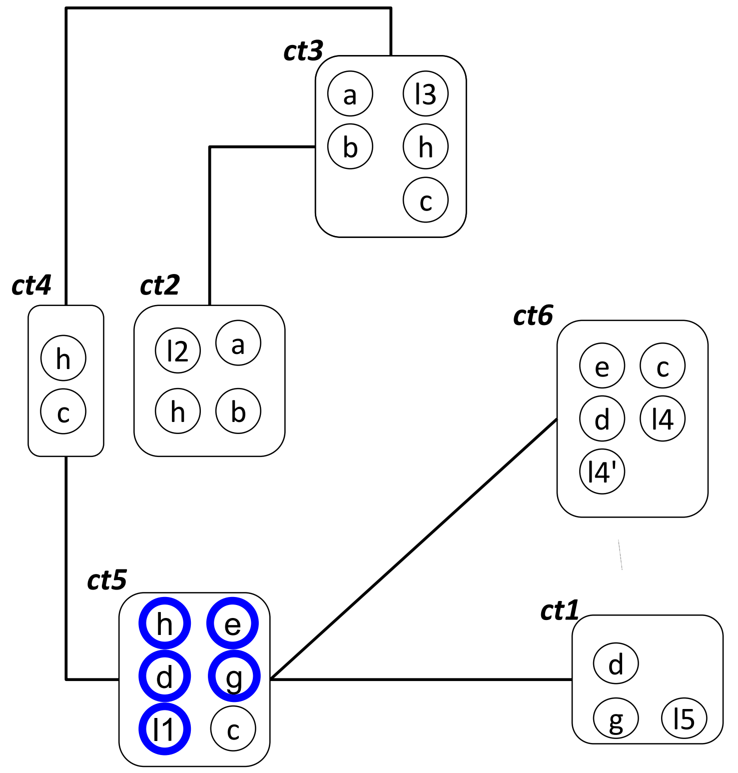
Distributed Tree Decomposition

1) is a tree of clusters
2) preserves the variables dependencies
3) respects the running intersection property
4) preserves the peers acquaintance
5) respectis the privacy of local variables

Distributed Tree Decomposition



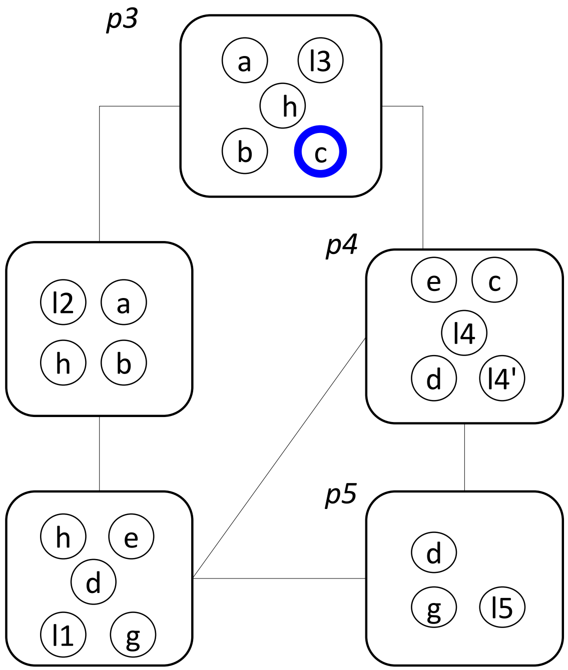
Acquaintance Graph



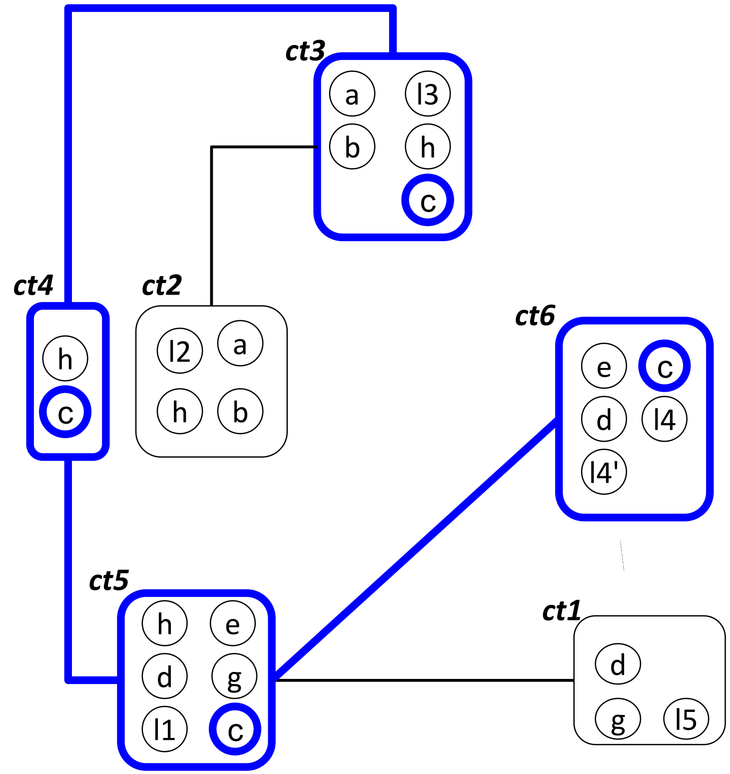
Distributed Tree Decomposition

1) is a tree of clusters
2) preserves the variables dependencies
3) respects the running intersection property
4) preserves the peers acquaintance
5) respectis the privacy of local variables

Distributed Tree Decomposition



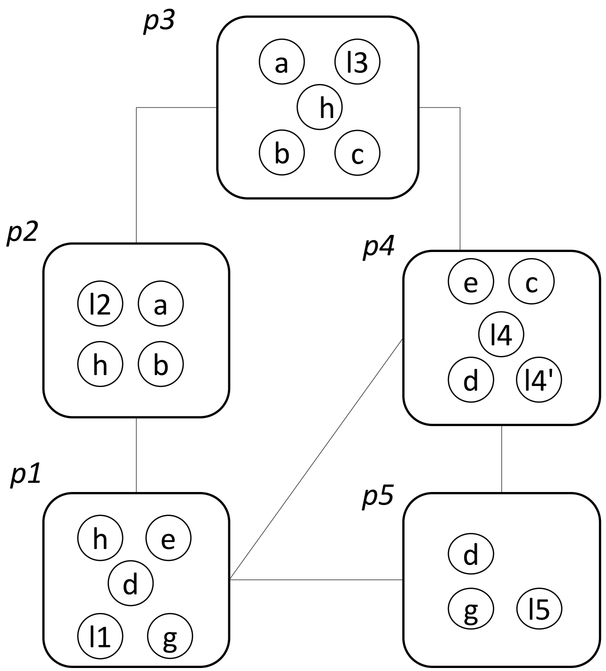
Acquaintance Graph

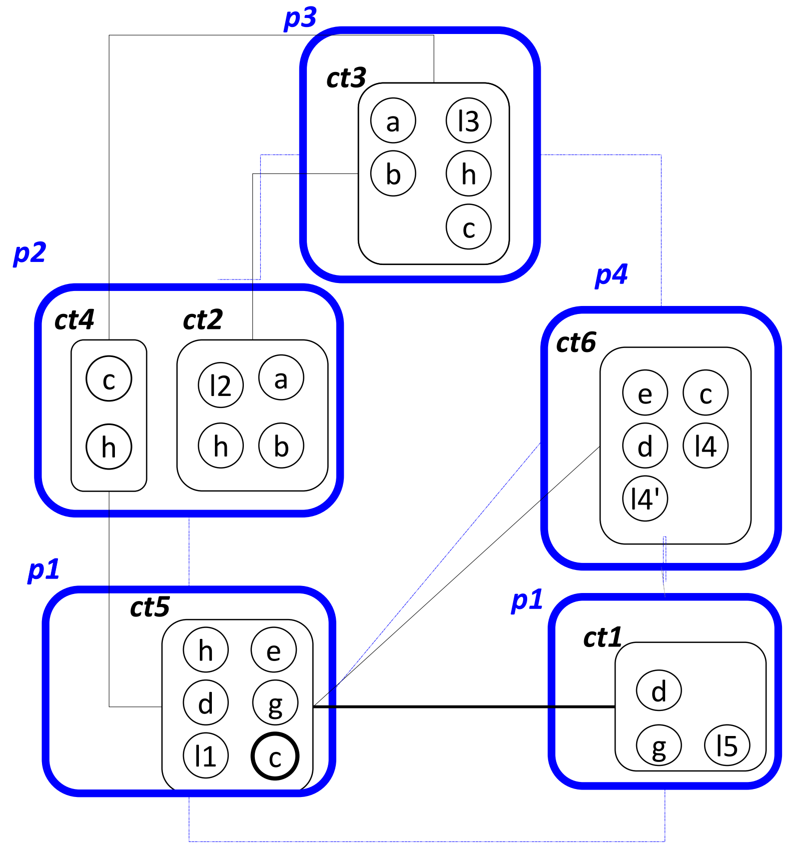


Distributed Tree Decomposition

1) is a tree of clusters
2) preserves the variables dependencies
3) respects the running intersection property
4) preserves the peers acquaintance
5) respectis the privacy of local variables

Distributed Tree Decomposition





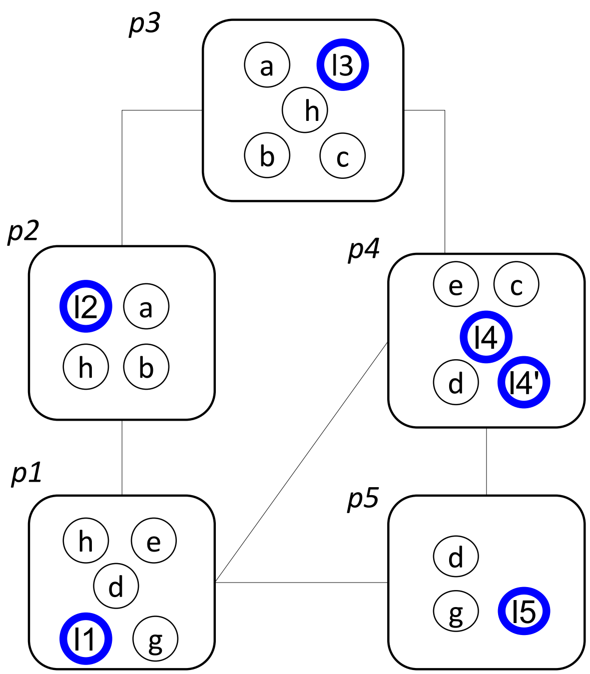
Acquaintance Graph

Distributed Tree Decomposition

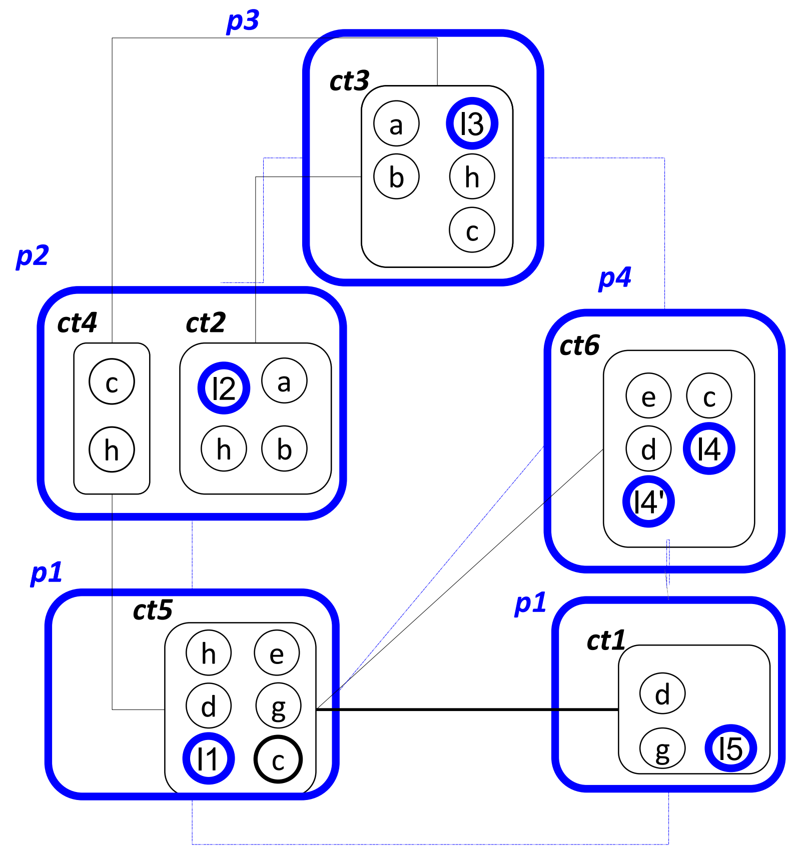
-a cluster is created by one peer
-2 neighboring clusters come from:
  - the same peer
  - neighboring peers

1) is a tree of clusters
2) preserves the variables dependencies
3) respects the running intersection property
4) preserves the peers acquaintance
5) respects the privacy of local variables

Distributed Tree Decomposition



Acquaintance Graph



Distributed Tree Decomposition

1) is a tree of clusters
2) preserves the variables dependencies
3) respects the running intersection property
4) preserves the peers acquaintance
5) respects the privacy of local variables

A local variable from pi can only appear in a cluster created by pi

Outline

- Introduction
- Distributed tree decomposition
  Preserve network structure
  Keep local information local
- Centralized tree decomp. VS concurrent approaches
- ***Token elimination***
- Experimental results on small-world graphs
- Conclusion / perspectives

Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

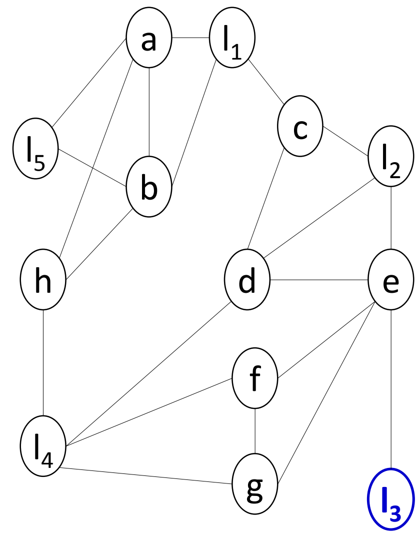
Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

It is always possible to build a TD from the clusters induced by Elimination order

Elimination process	Primal graph	Elimination order	Clusters

While the graph is not empty

1) **Choose a variable** v
2) Add edges between unconnected neighbors
3) Create a cluster (v ∪ neighbors)
4) Eliminate v



l_3

Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

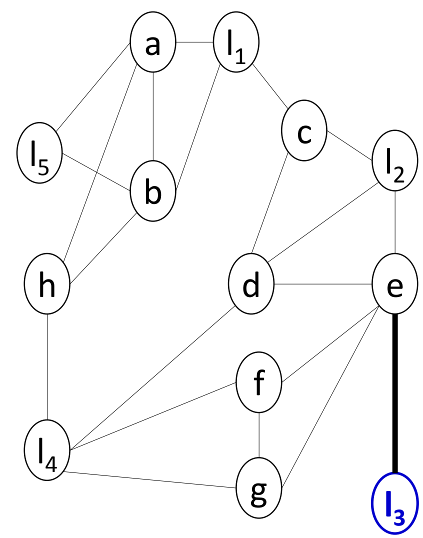
Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

It is always possible to build a TD from the clusters induced by Elimination order

Elimination process	Primal graph	Elimination order	Clusters

While the graph is not empty

1) Choose a variable v
2) **Add edges between unconnected neighbors**
3) Create a cluster (v ∪ neighbors)
4) Eliminate v



l_3

Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

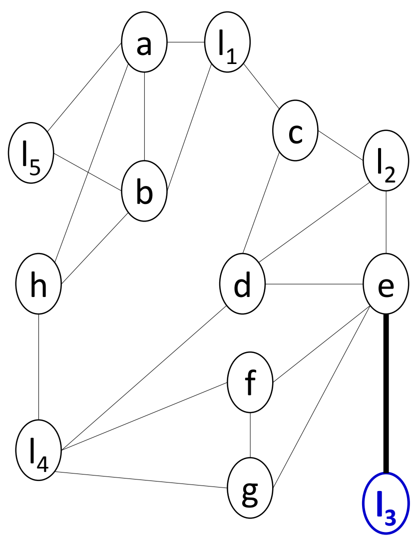
Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

It is always possible to build a TD from the clusters induced by Elimination order

Elimination process	Primal graph	Elimination order	Clusters

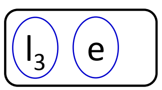
While the graph is not empty

1) Choose a variable v
2) Add edges between unconnected neighbors
4) **Create a cluster (v ∪ neighbors)**
3) Eliminate v



l_3

ct_{l3}



Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

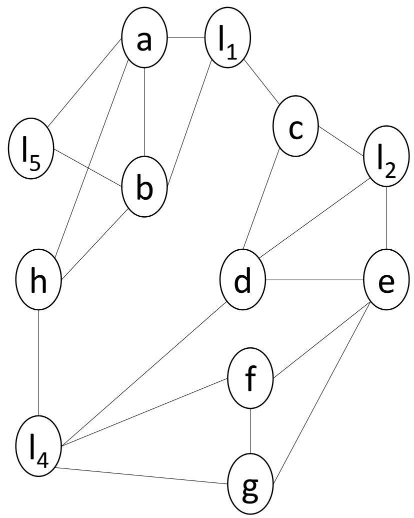
It is always possible to build a TD from the clusters induced by Elimination order

Elimination process

While the graph is not empty

1) Choose a variable v
2) Add edges between unconnected neighbors
4) Create a cluster (v ∪ neighbors)
3) **Eliminate** v

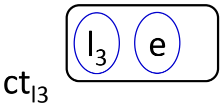
Primal graph



Elimination order

l_3

Clusters



Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

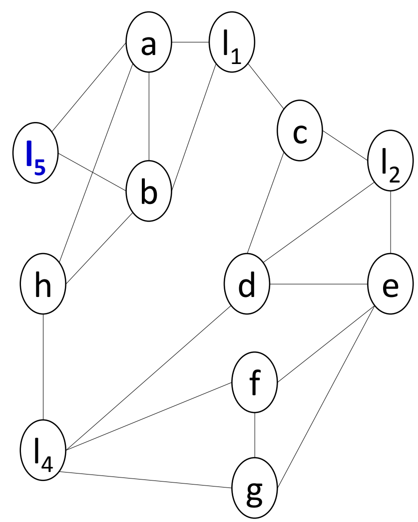
It is always possible to build a TD from the clusters induced by Elimination order

Elimination process

While the graph is not empty

1) **Choose a variable** v
2) Add edges between unconnected neighbors
4) Create a cluster (v ∪ neighbors)
3) Eliminate v

Primal graph



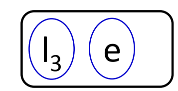
Elimination order

l_3

l_5

Clusters

ct_{l3}



Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

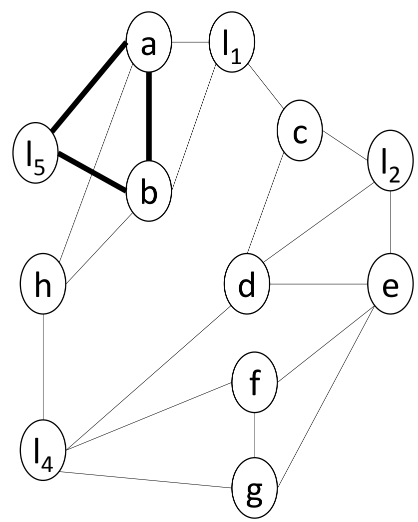
It is always possible to build a TD from the clusters induced by Elimination order

Elimination process

While the graph is not empty

1) Choose a variable v
2) **Add edges between unconnected neighbors**
4) Create a cluster (v ∪ neighbors)
3) eliminate v

Primal graph

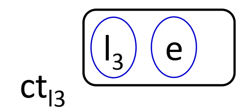


Elimination order

l_3

l_5

Clusters



Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

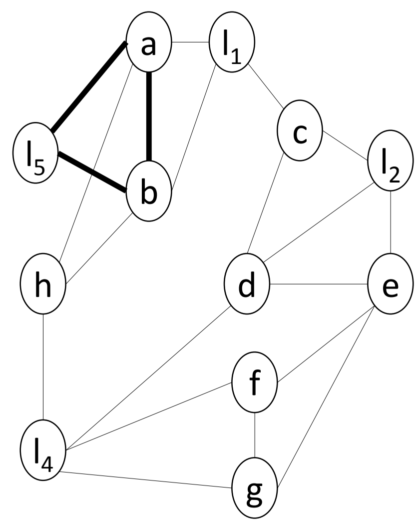
It is always possible to build a TD from the clusters induced by Elimination order

Elimination process

While the graph is not empty

1) Choose a variable v
2) Add edges between unconnected neighbors
4) **Create a cluster (v ∪ neighbors)**
3) eliminate v

Primal graph

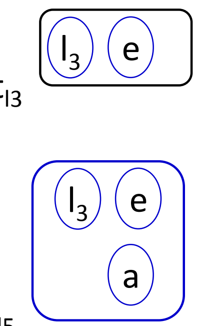


Elimination order

l_3

l_5

Clusters



ct_{l3}

ct_{l5}

Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

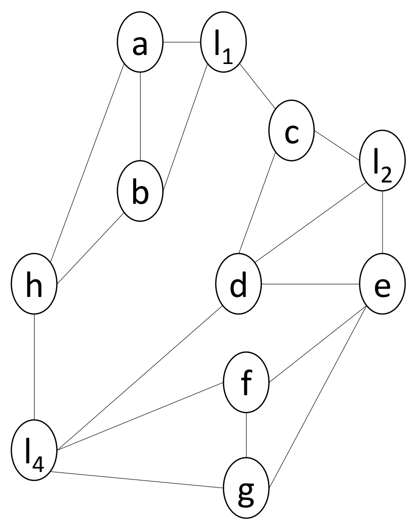
It is always possible to build a TD from the clusters induced by Elimination order

Elimination process

While the graph is not empty

1) Choose a variable v
2) Add edges between unconnected neighbors
4) Create a cluster (v ∪ neighbors)
3) **eliminate** v

Primal graph

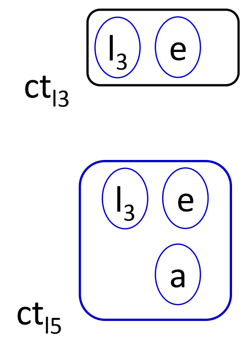


Elimination order

l_3

l_5

Clusters



Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

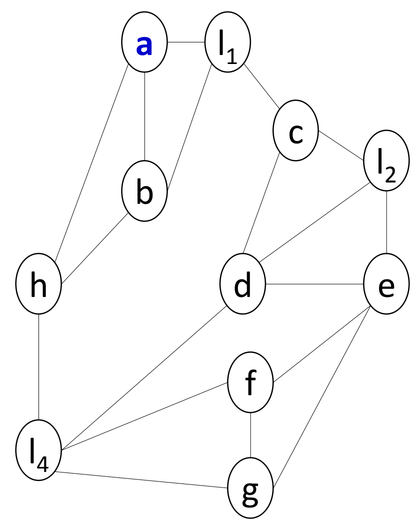
It is always possible to build a TD from the clusters induced by Elimination order

Elimination process

While the graph is not empty

1) **Choose a variable** v
2) Add edges between unconnected neighbors
4) Create a cluster (v ∪ neighbors)
3) Eliminate v

Primal graph



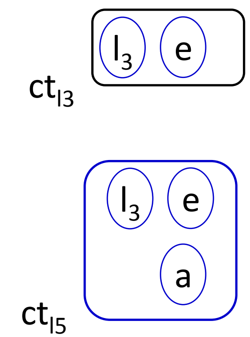
Elimination order

$\mathbf{l_3}$

$\mathbf{l_5}$

a

Clusters



Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

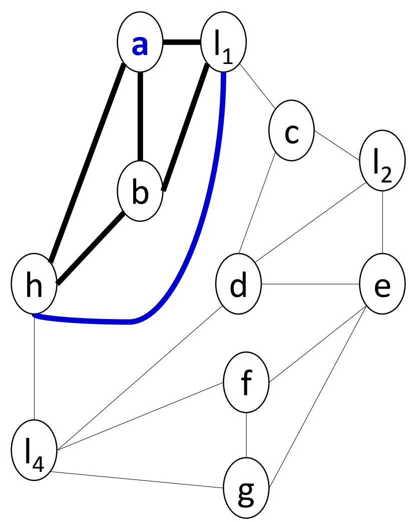
Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

It is always possible to build a TD from the clusters induced by Elimination order

Elimination process	Primal graph	Elimination order	Clusters

While the graph is not empty

1) Choose a variable v
2) **Add edges between unconnected neighbors**
4) Create a cluster (v ∪ neighbors)
3) Eliminate v



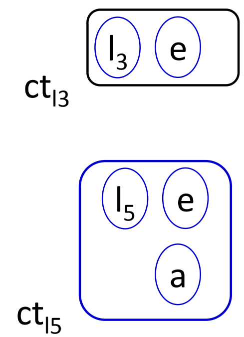
Elimination order:

l_3

l_5

a

…



Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

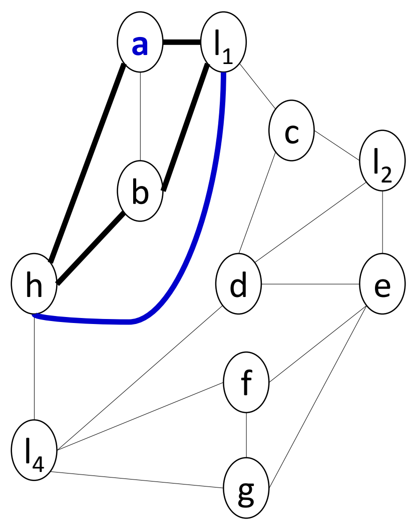
It is always possible to build a TD from the clusters induced by Elimination order

Elimination process

While the graph is not empty

1) Choose a variable v
2) Add edges between unconnected neighbors
4) **Create a cluster (v ∪ neighbors)**
3) Eliminate v

Primal graph



Elimination order

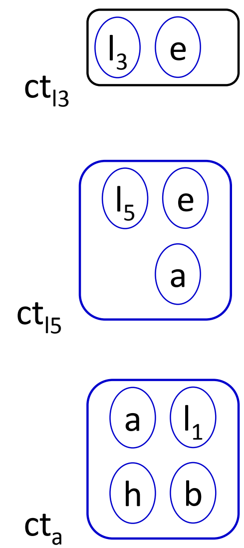
l_3

l_5

a

...

Clusters



Lesson learned from centralized context

What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

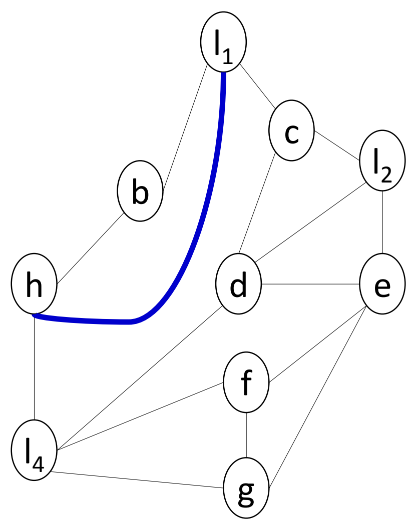
It is always possible to build a TD from the clusters induced by Elimination order

Elimination process

While the graph is not empty

1) Choose a variable v
2) Add edges between unconnected neighbors
4) Create a cluster (v ∪ neighbors)
3) **Eliminate** v

Primal graph



Elimination order

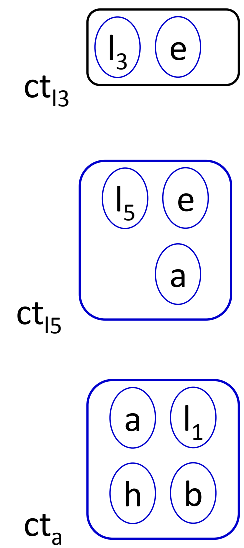
l_3

l_5

a

…

Clusters

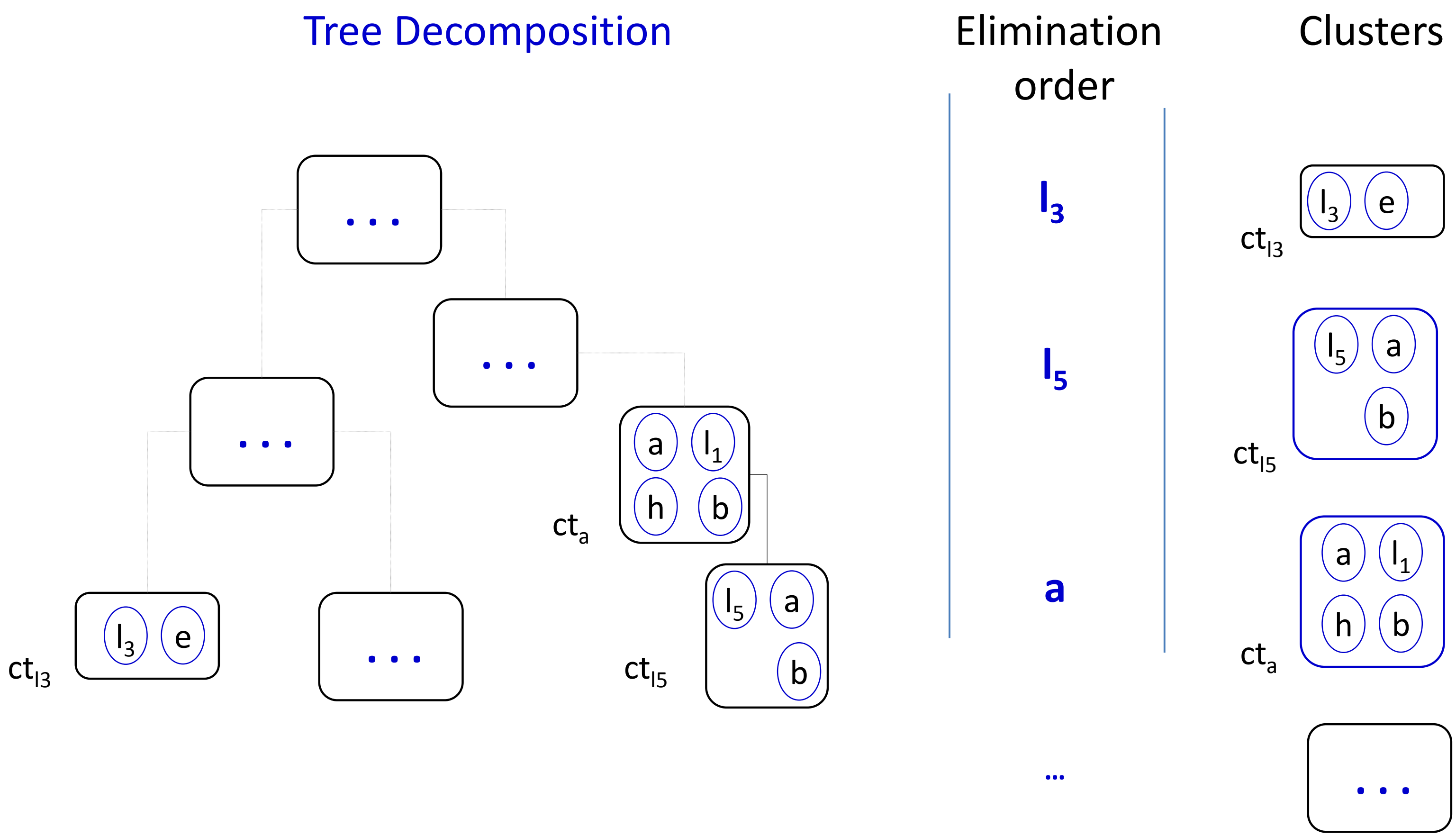


Lesson learned from centralized context

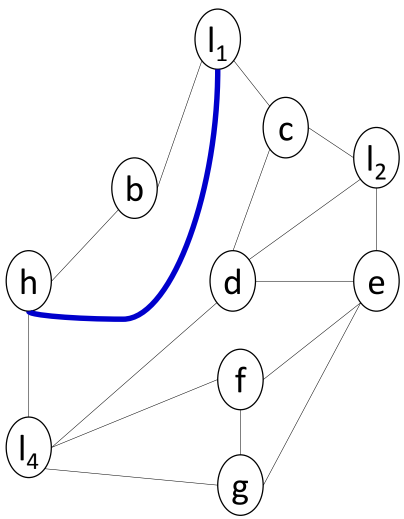
What are the good tree decomposition techniques? Why?

Finding optimal Tree Decomposition ⇔ Finding optimal Elimination Order

It is always possible to build a TD from the clusters induced by Elimination order



Lesson learned from centralized context



Observation: The edge added between l1 and h will increase the size of the cluster induced l1 or h

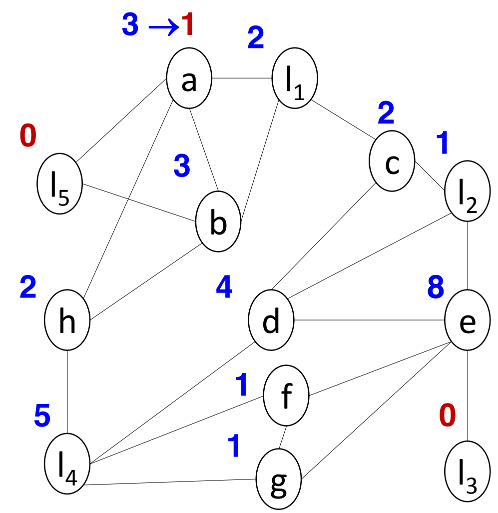
⇩

Remark: If we add no edges → Perfect elimination

⇩

Heuristic: Eliminate first the variable that minimizes the number of additional edges : (**Min Fill**)

⇩



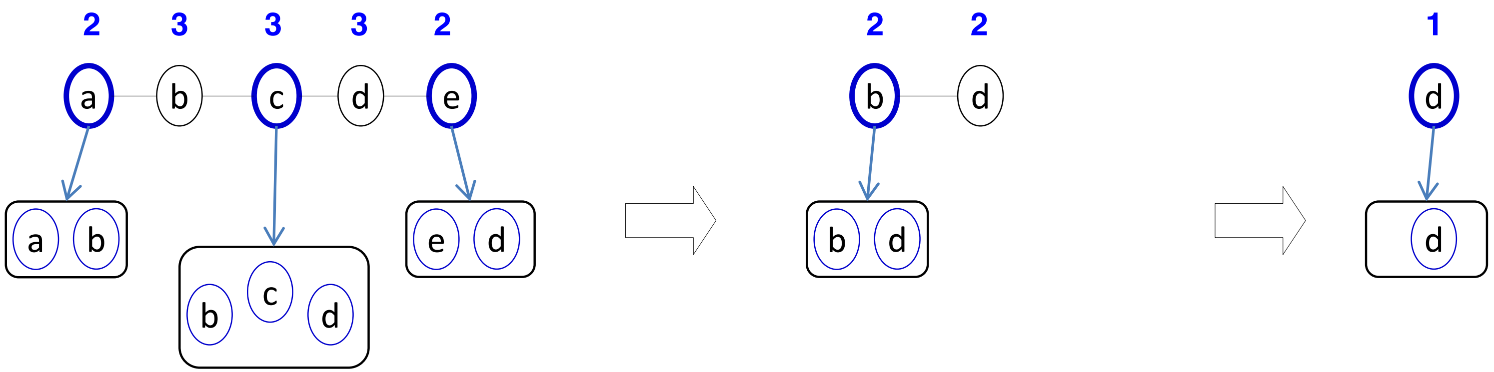
Pb: elimination order cannot be directly applied
No privacy, No notion of acquaintance links

Idea : Weight each node by the quality of the clusters that the node will produce if it is the next to be eliminated

Lesson learn from distributed context

Intuition:

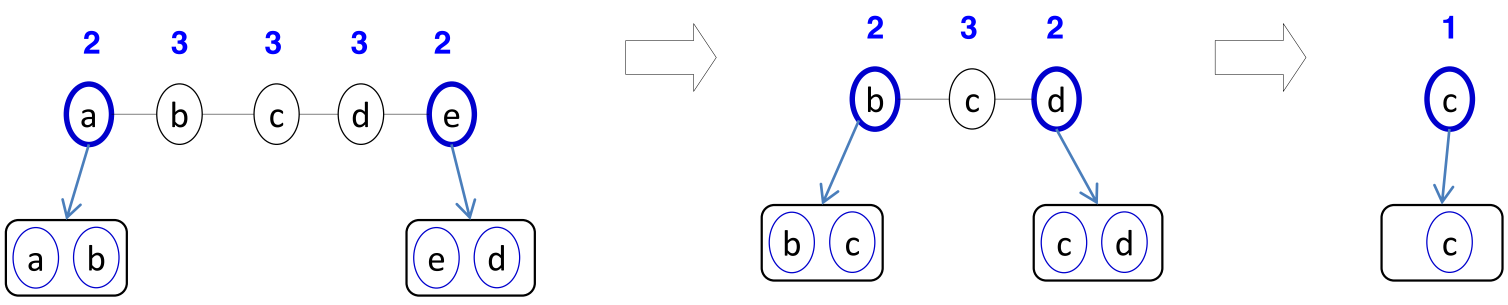
distributed settings can speed up the elimination process by concurrent eliminations



width:3

concurrent eliminations

{a, c ,e }



width:2

concurrent eliminations

{a, e }

concurrent eliminations

{b, d }

Concurrent eliminations can be bad for tree decomposition

Outline

- Introduction
- Distributed tree decomposition
  Preserve network structure
  Keep local information local
- Centralized tree decomp. VS concurrent approaches
- ***Token elimination***
- Experimental results on small-world graphs
- Conclusion / perspectives

Token Elimination: Principle

- Distributed algorithm
  - Phase 1: Implicit building of a DTD
    - **Elimination**
    - **Local elections and votes**
    - **Token passing**
  - Phase 2: clusters reconnection (acquaintance property).

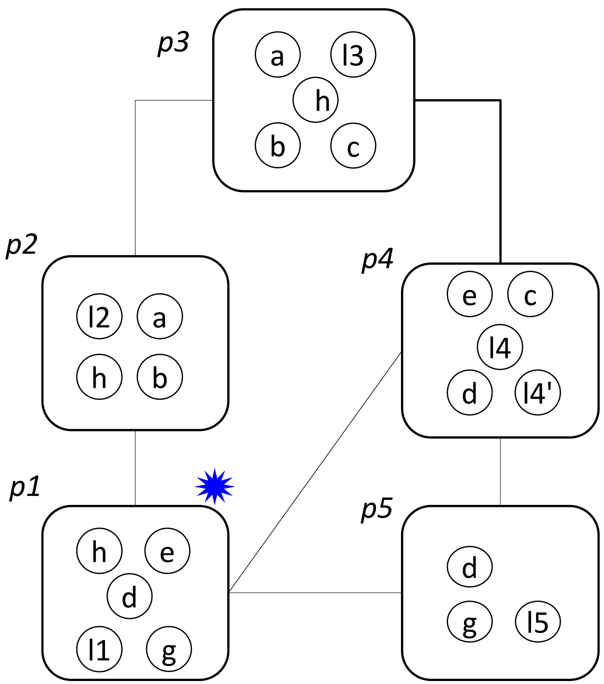
- Heuristics:
  - Min-Cluster: Each peer estimates the size of the cluster it will produce if it is the next to be eliminated.
  - Min-Proj : Each peer estimates the size of additional variables it will add to the token if it is the next to be eliminated.

Token Elimination: Min Cluster

Distributed algorithm

p receives the token

- organizes a local election
- peers vote , p is a local minimal ?
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster,
    adds shared variables to the token,
    reorganizes local election
    sends the token

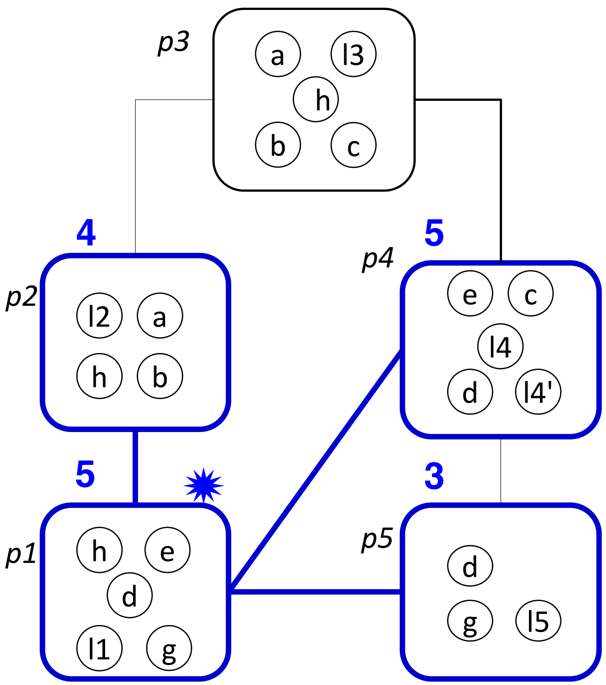


On going Distributed Tree Decomposition

Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organizes a local election
- peers vote , p is a local minimal ?
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster,
    adds shared variables to the token,
    reorganizes local election
    sends the token

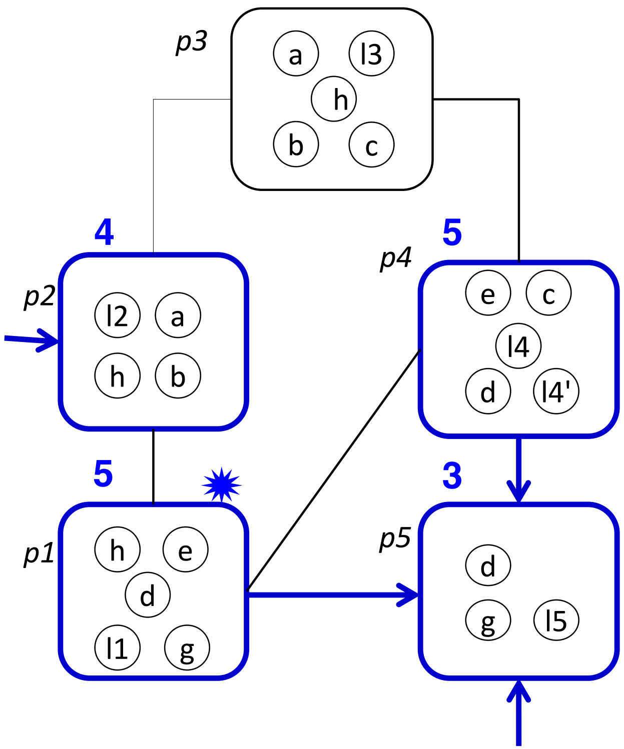


On going Distributed Tree Decomposition

Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster,
    adds shared variables to the token,
    reorganizes local election
    sends the token

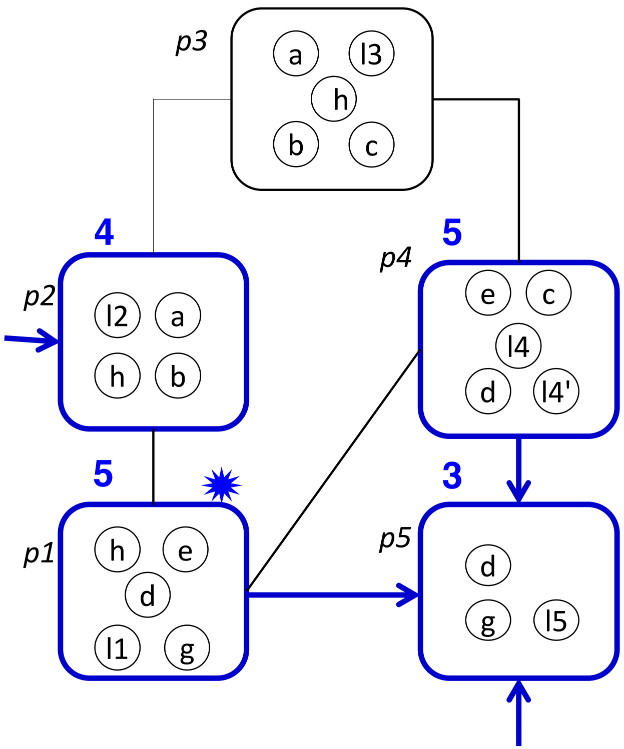


On going Distributed Tree Decomposition

Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . **No: sends the token**
  . Yes: eliminates itself, creates a new cluster,
    adds shared variables to the token,
    reorganizes local election
    sends the token



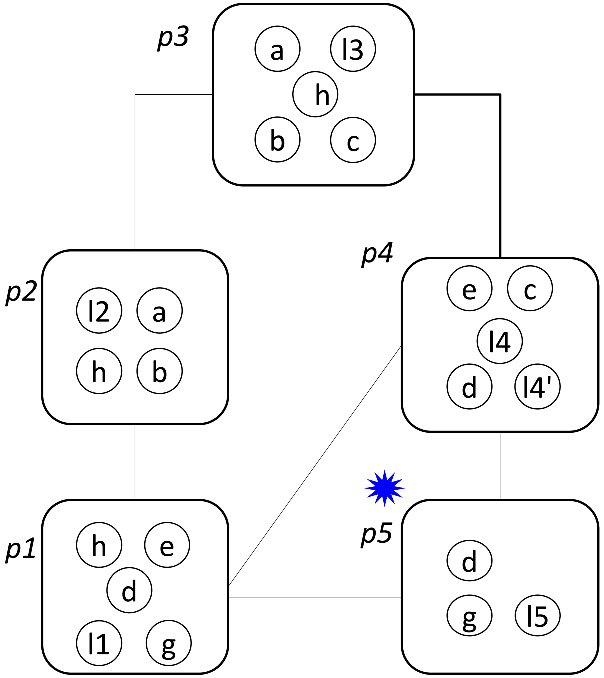
On going Distributed Tree Decomposition

Token Elimination: Min Cluster

Distributed algorithm

p receives the token

- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster,
    adds shared variables to the token,
    reorganizes local election
    sends the token

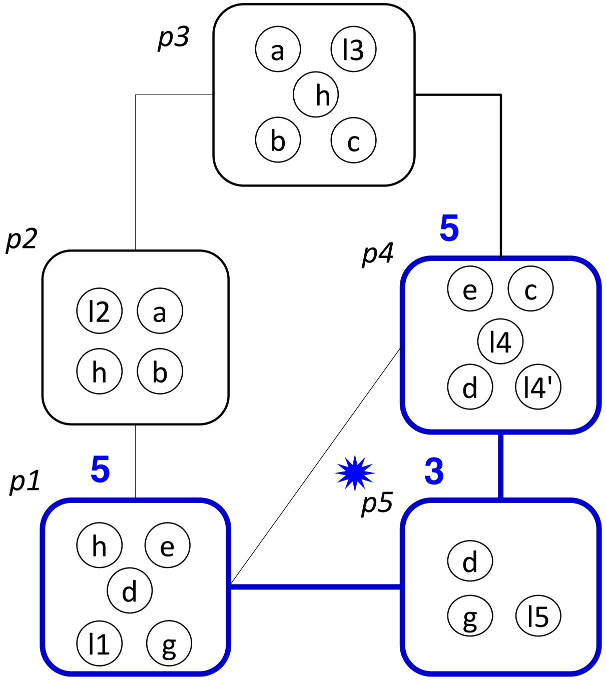


On going Distributed Tree Decomposition

Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organizes a local election
- peers vote , p is a local minimal ?
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster,
    adds shared variables to the token,
    reorganizes local election
    sends the token

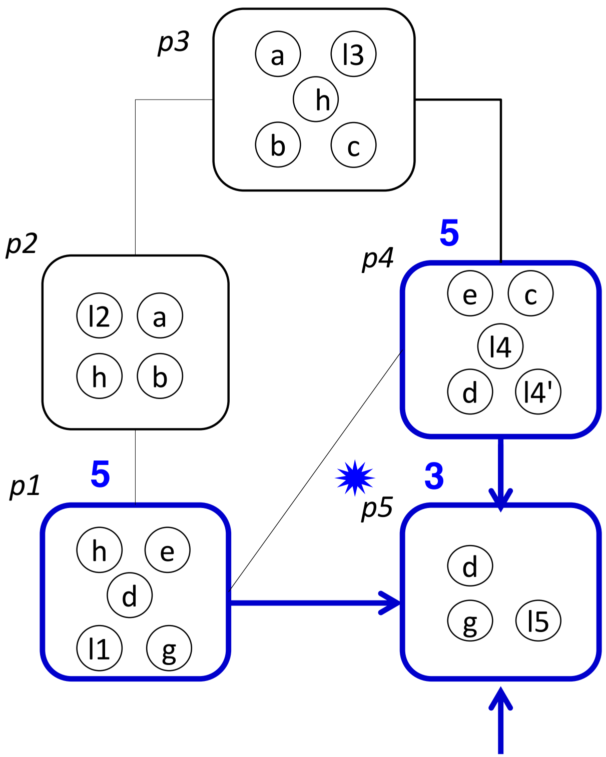


On going Distributed Tree Decomposition

Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster, adds shared variables to the token, reorganizes local election sends the token

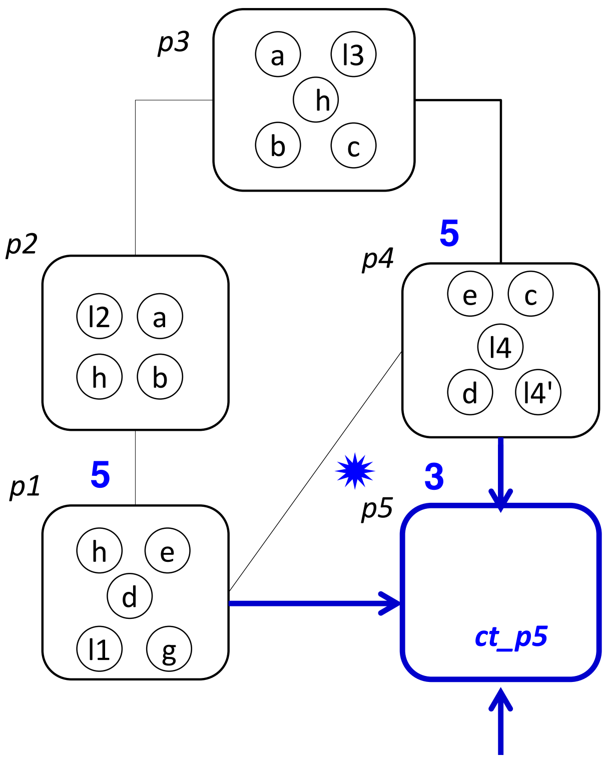


On going Distributed Tree Decomposition

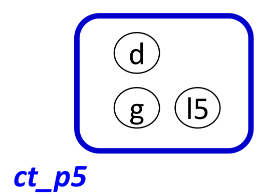
Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . **Yes: eliminates itself, creates a new cluster**, adds shared variables to the token, reorganizes local election
  peers vote and sends the token



On going Distributed Tree Decomposition

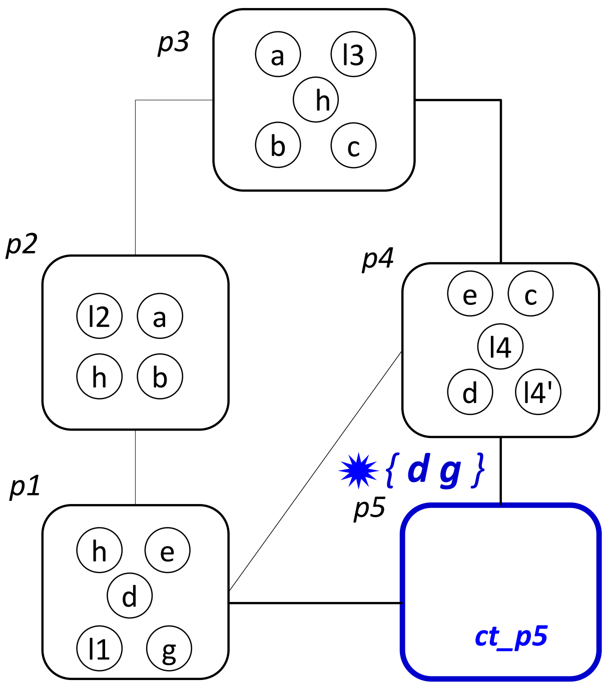


p5 creates the cluster for l5 (privacy)

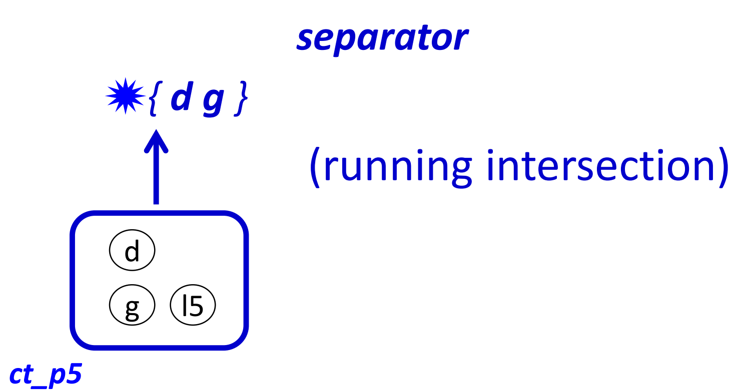
Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster,
  adds shared variables to the token,
  reorganizes local election
  peers vote and sends the token



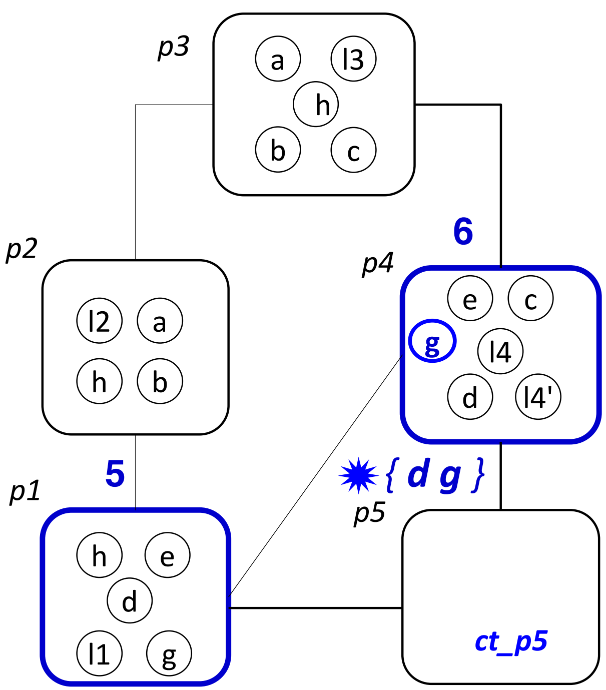
On going Distributed Tree Decomposition



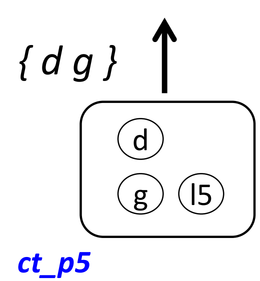
Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster, adds shared variables to the token, **reorganizes local election** peers vote and sends the token



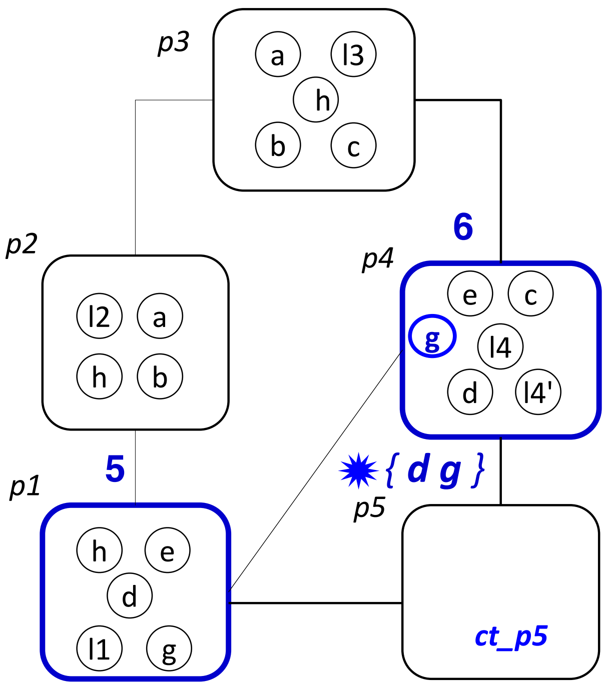
On going Distributed Tree Decomposition



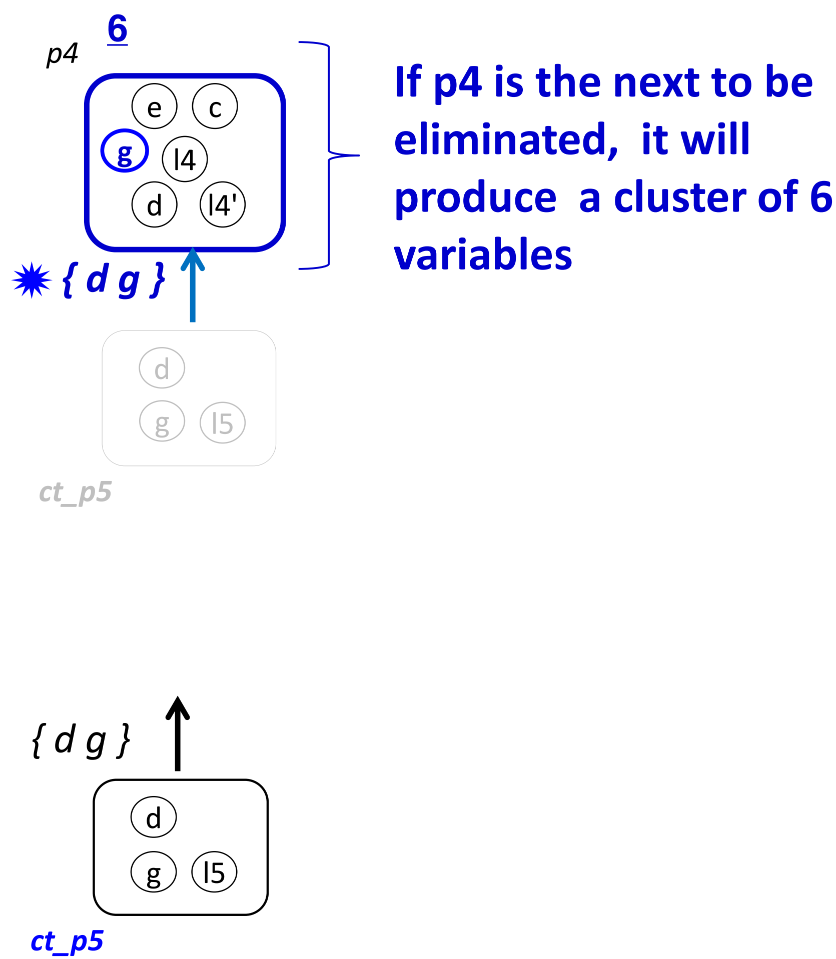
Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster, adds shared variables to the token, **reorganizes local election**
  peers vote and sends the token



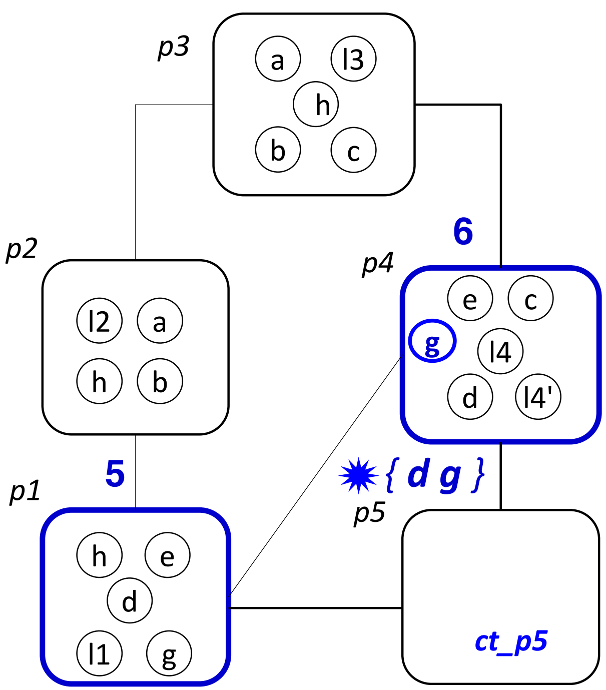
On going Distributed Tree Decomposition



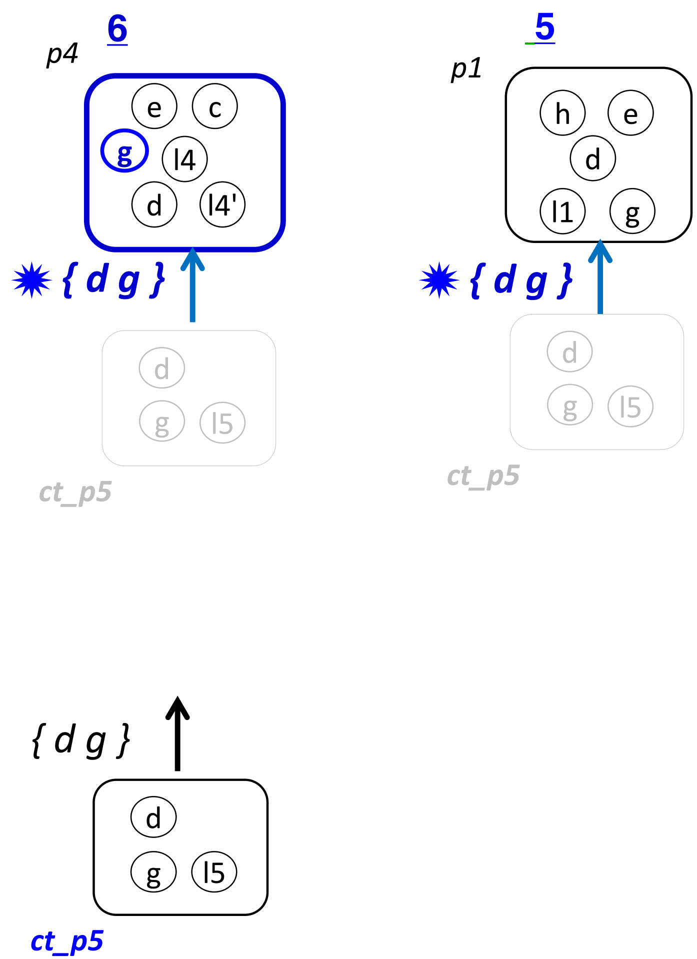
Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster, adds shared variables to the token, **reorganizes local election** peers vote and sends the token



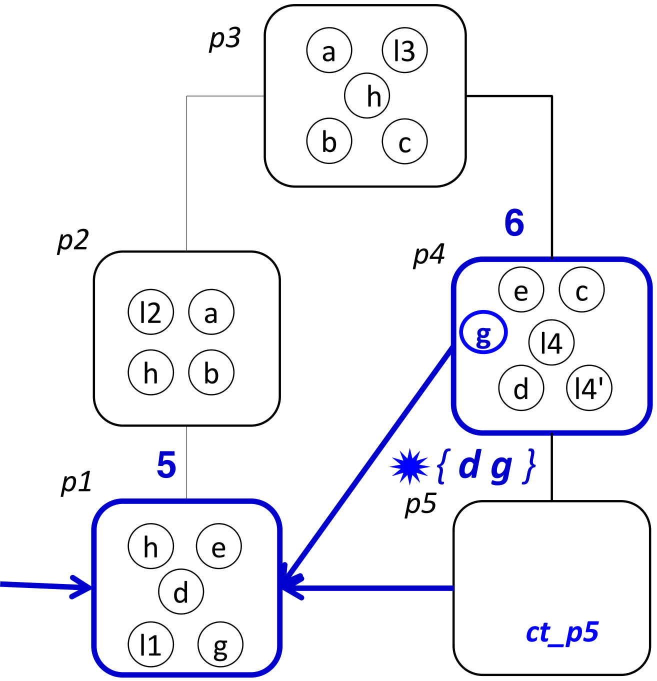
On going Distributed Tree Decomposition



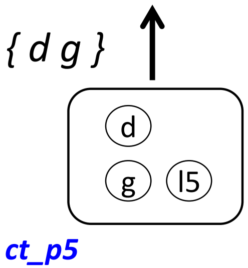
Token Elimination: Min Cluster

Distributed algorithm

p receives the token
- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster, adds shared variables to the token, **reorganizes local election**
  peers vote and p sends the token



On going Distributed Tree Decomposition

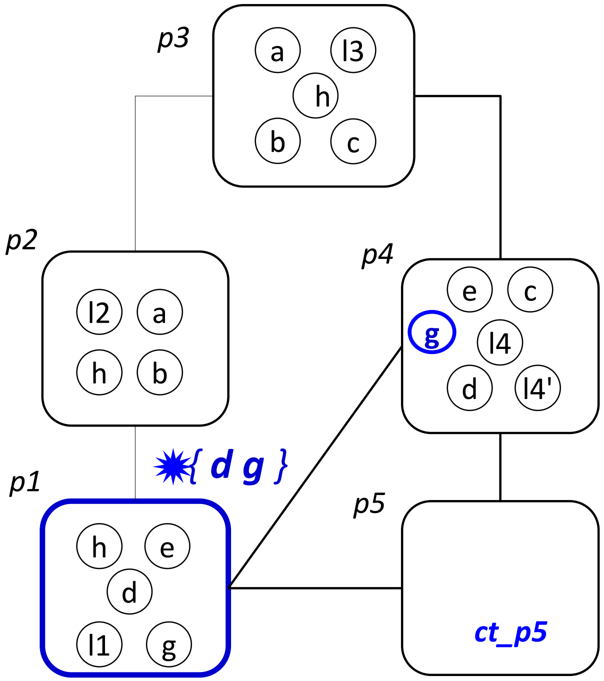


Token Elimination: Min Cluster

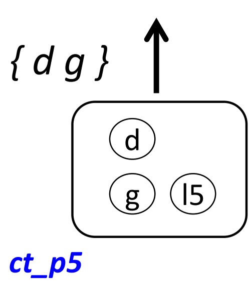
Distributed algorithm

p receives the token

- organize a local election
- **peers vote , p is a local minimal ?**
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster, adds shared variables to the token, **reorganizes local election** **peers vote and p sends the token**



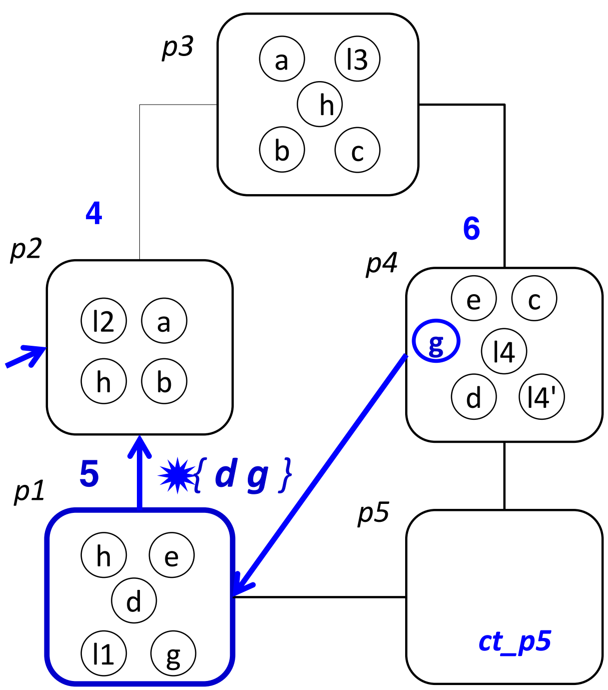
On going Distributed Tree Decomposition



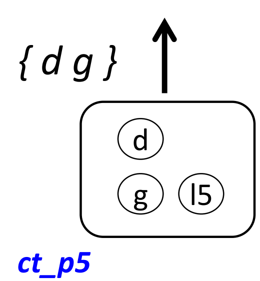
Token Elimination: Min Cluster

Distributed algorithm

p1 receives the token
- organize a local election
- peers vote , p1 is a local minimal ?
. **No: sends the token**
. Yes: eliminates itself, creates a new cluster, adds shared variables to the token, reorganizes local election peers vote and p sends the token



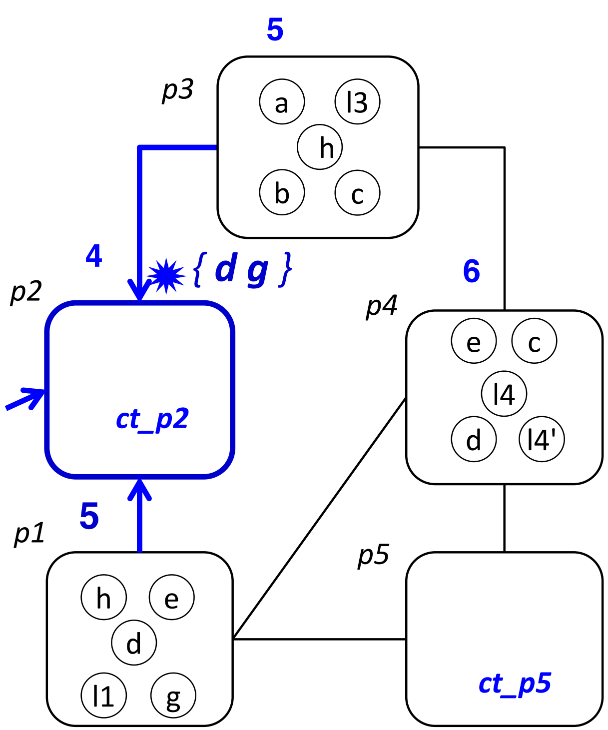
On going Distributed Tree Decomposition



Token Elimination: Min Cluster

Distributed algorithm

p2 receives the token
- organize a local election
- peers vote , p2 is a local minimal ?
. No: sends the token
. **Yes: eliminates itself, creates a new cluster**,
adds shared variables to the token,
reorganizes local election
peers vote and p sends the token

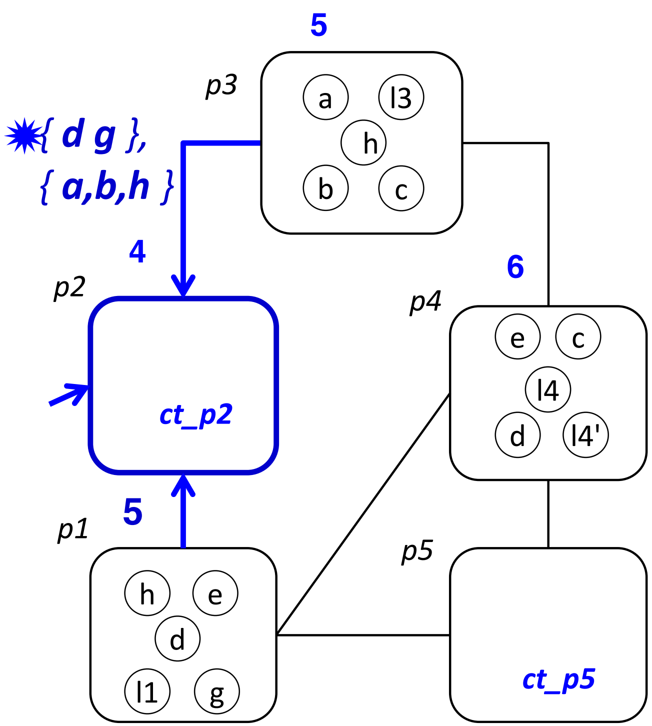


On going Distributed Tree Decomposition

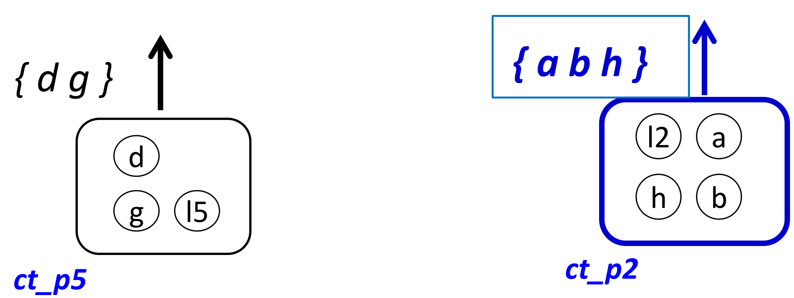
Token Elimination: Min Cluster

Distributed algorithm

p2 receives the token
- organize a local election
- peers vote , p2 is a local minimal ?
. No: sends the token
. **Yes: eliminates itself, creates a new cluster**,
adds shared variables to the token,
reorganizes local election
peers vote and p sends the token



On going Distributed Tree Decomposition



Token Elimination: Min Cluster

Distributed algorithm

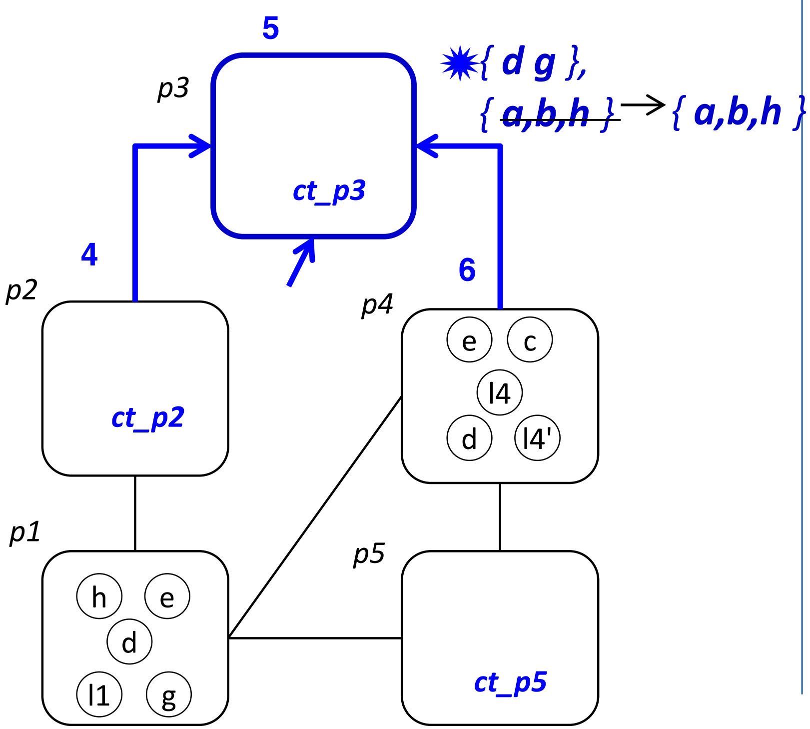
p3 receives the token

- organize a local election

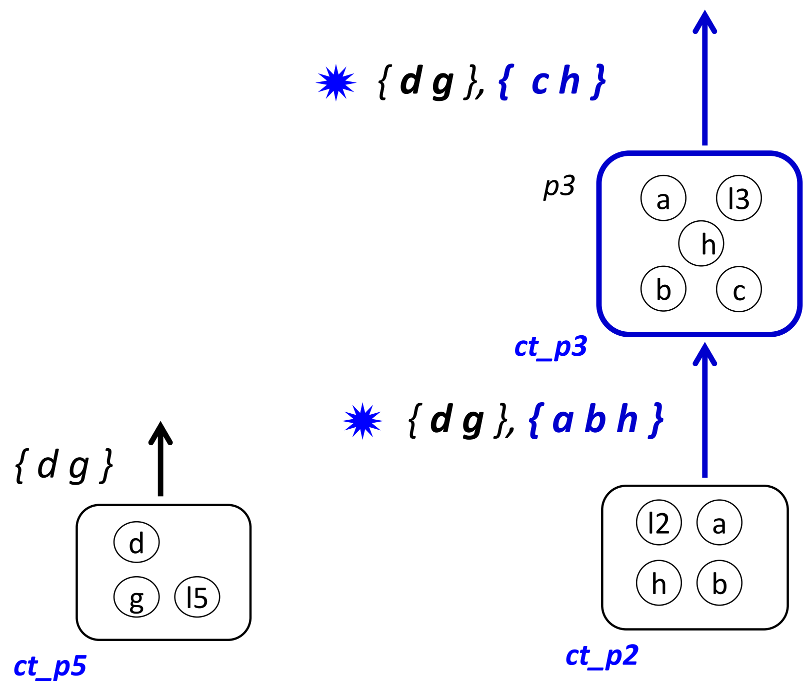
- peers vote , p3 is a local minimal ?

. No: sends the token

. **Yes: eliminates itself, creates a new cluster**, adds shared variables to the token, **reorganizes local election** **peers vote and p sends the token**



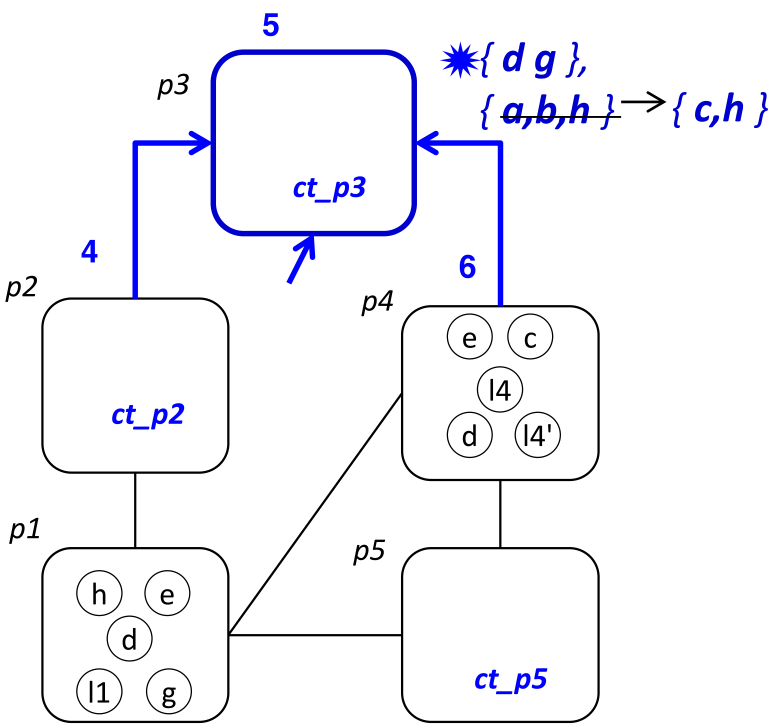
On going Distributed Tree Decomposition



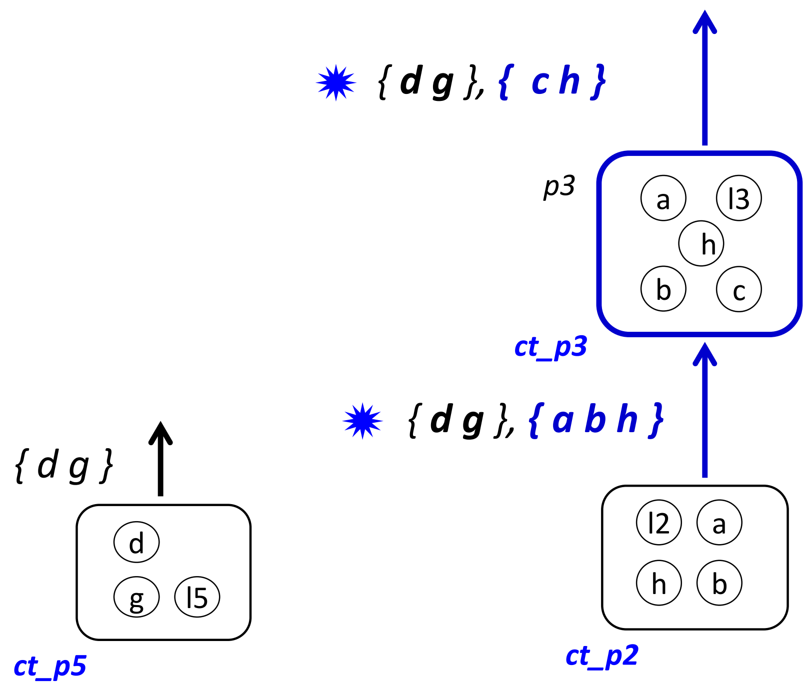
Token Elimination: Min Cluster

Distributed algorithm

p3 receives the token
- **organize a local election**
- **peers vote , p3 is a local minimal ?**
  - No: sends the token
  - **Yes: eliminates itself, creates a new cluster**, adds shared variables to the token, **reorganizes local election** **peers vote and p sends the token**



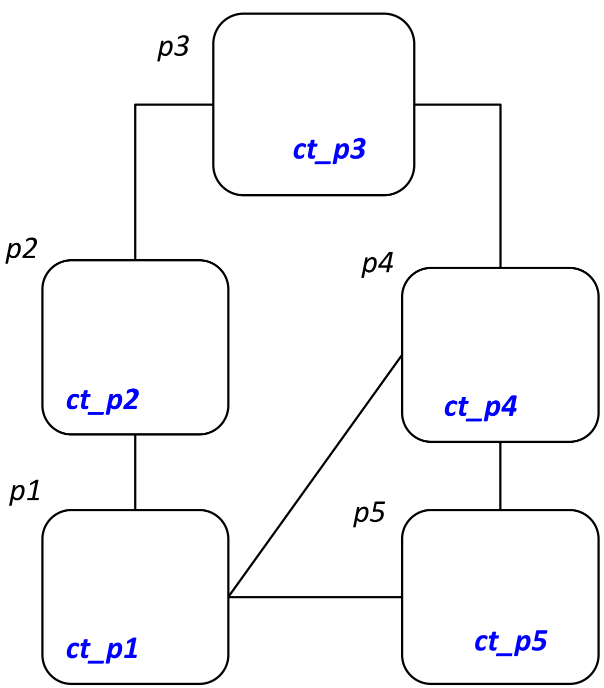
On going Distributed Tree Decomposition



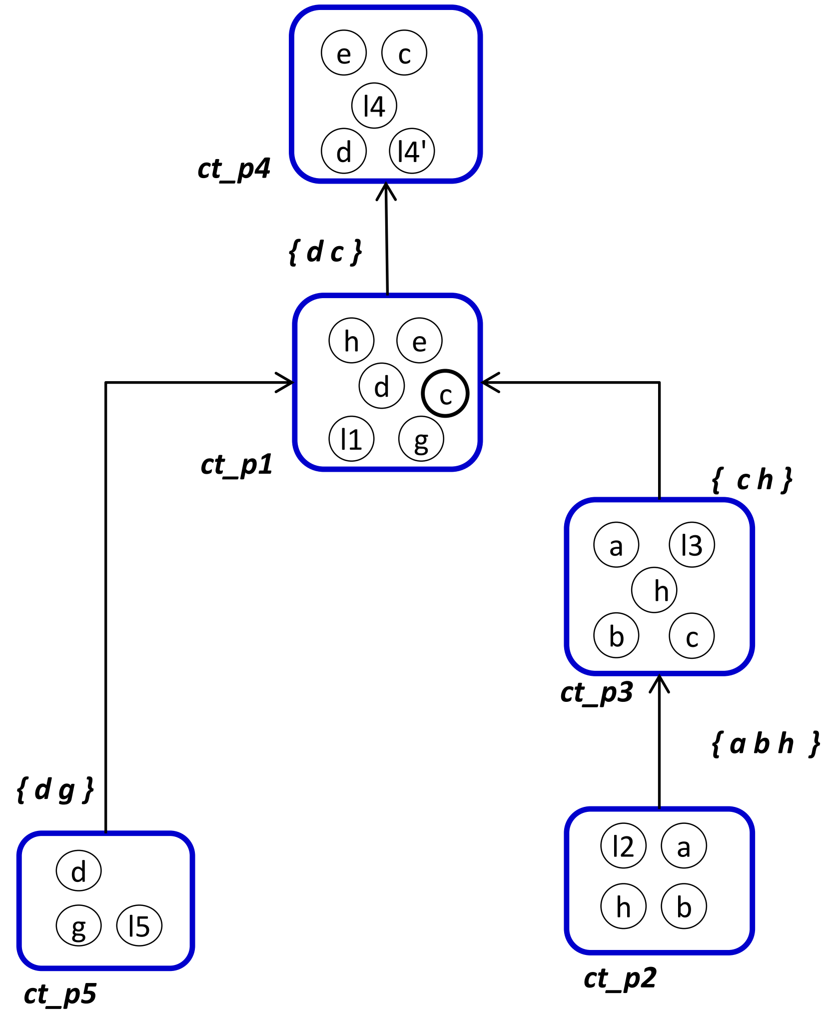
Token Elimination: Min Cluster

Distributed algorithm

p3 receives the token
- organize a local election
- peers vote , p3 is a local minimal ?
  . No: sends the token
  . Yes: eliminates itself, creates a new cluster, adds shared variables to the token, reorganizes local election peers vote and p sends the token



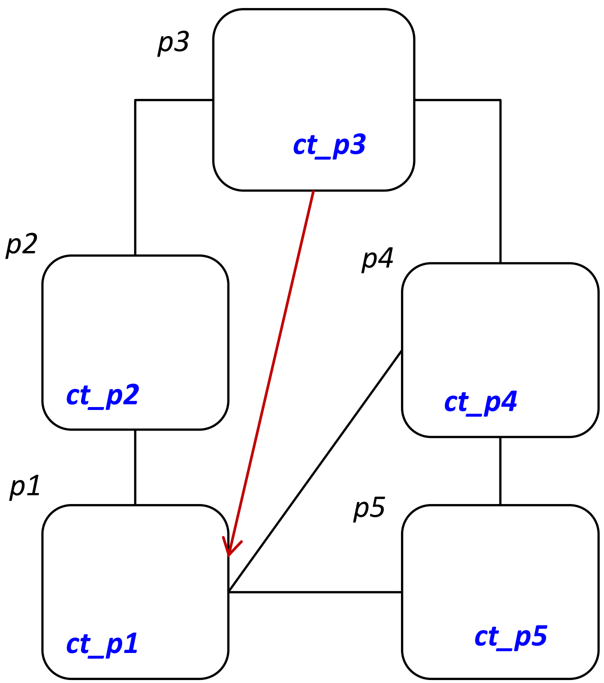
On going Distributed Tree Decomposition



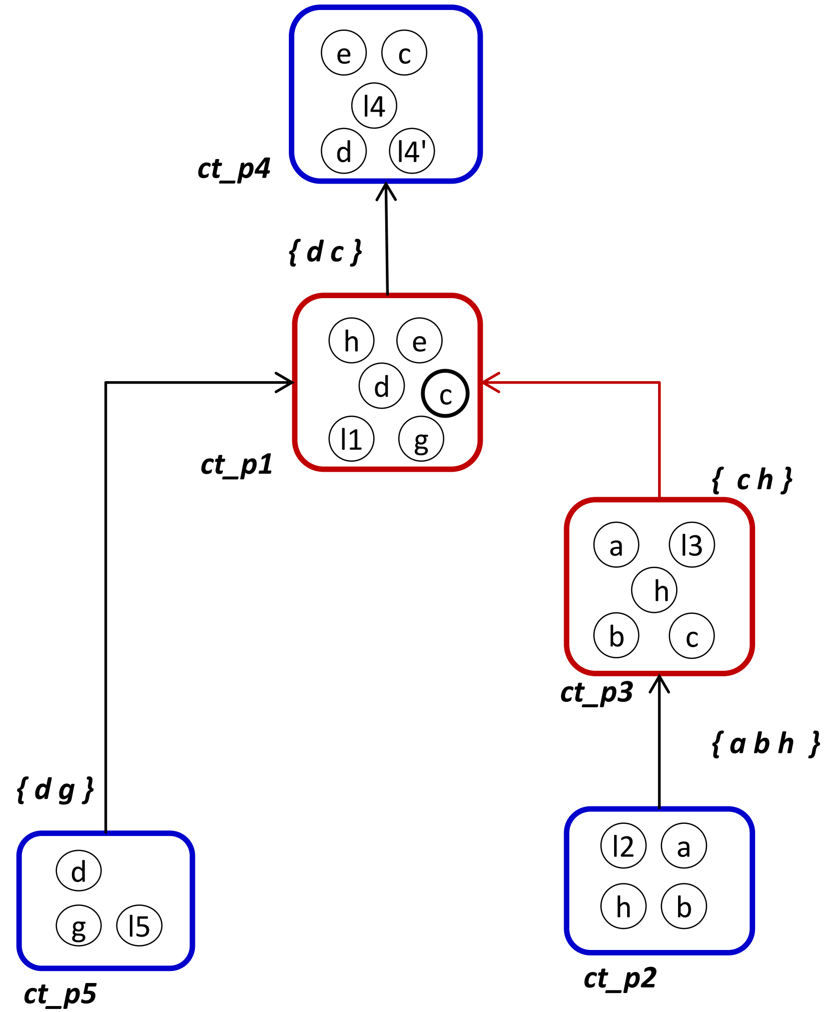
Token Elimination: Min Cluster

Distributed algorithm

p3 receives the token
- organize a local election
- peers vote , p3 is a local minimal ?
. No: sends the token
. **Yes: eliminates itself, creates a new cluster,**
adds shared variables to the token,
reorganizes local election
peers vote and p sends the token



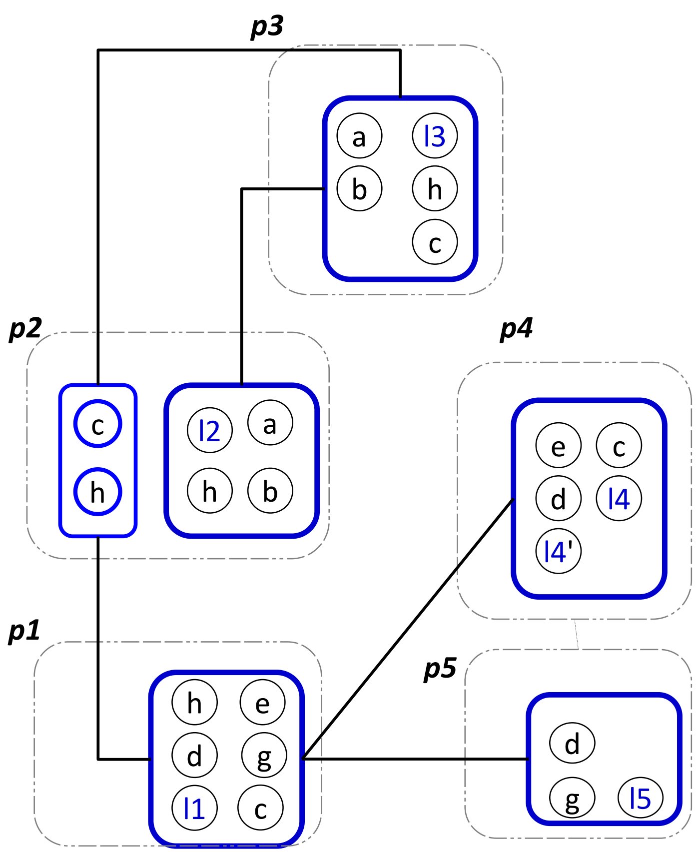
On going Distributed Tree Decomposition



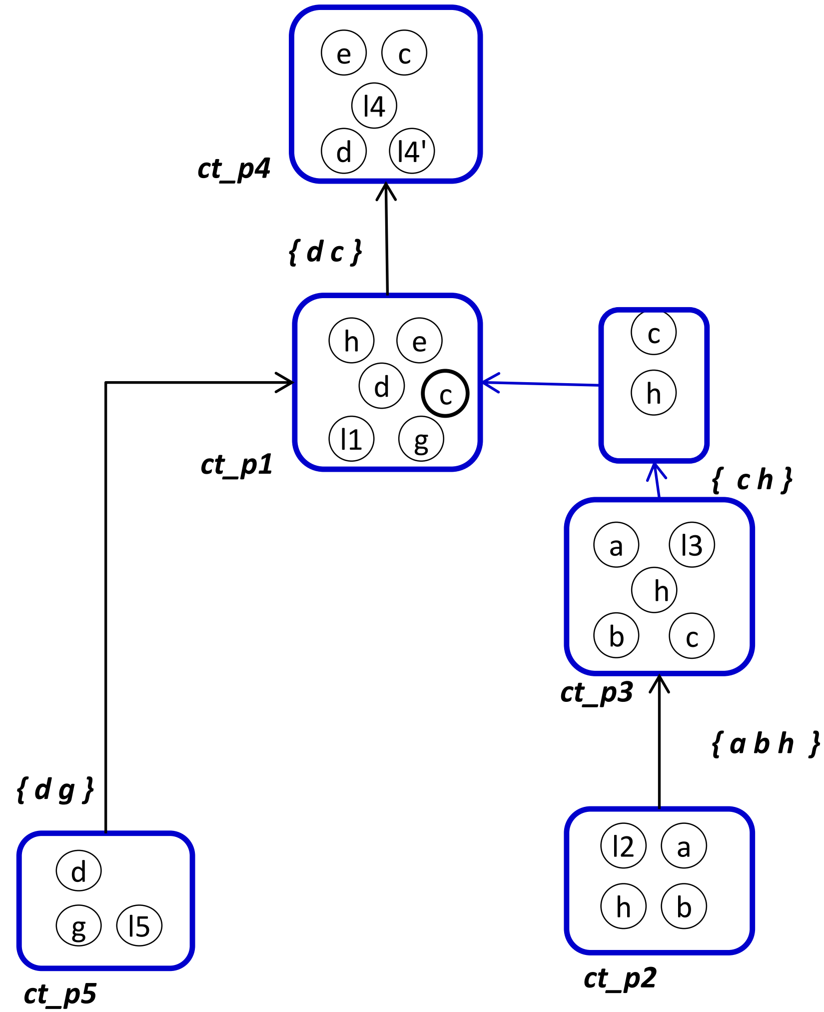
Pb : link between p3 and p1 does not follow the accointances

Token Elimination: Min Cluster

Distributed structured network



Final Distributed Tree Decomposition

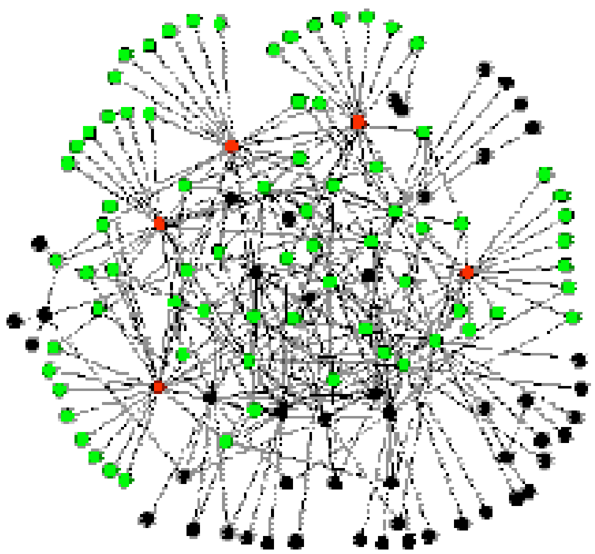


Outline

- Preliminary: Tree Decomposition
- Problematic: How to decompose a distributed system respecting privacy and acquaintances
- Distributed Tree Decomposition
- Token Elimination
- Experimental results on small world graph
- Conclusion et perspectives

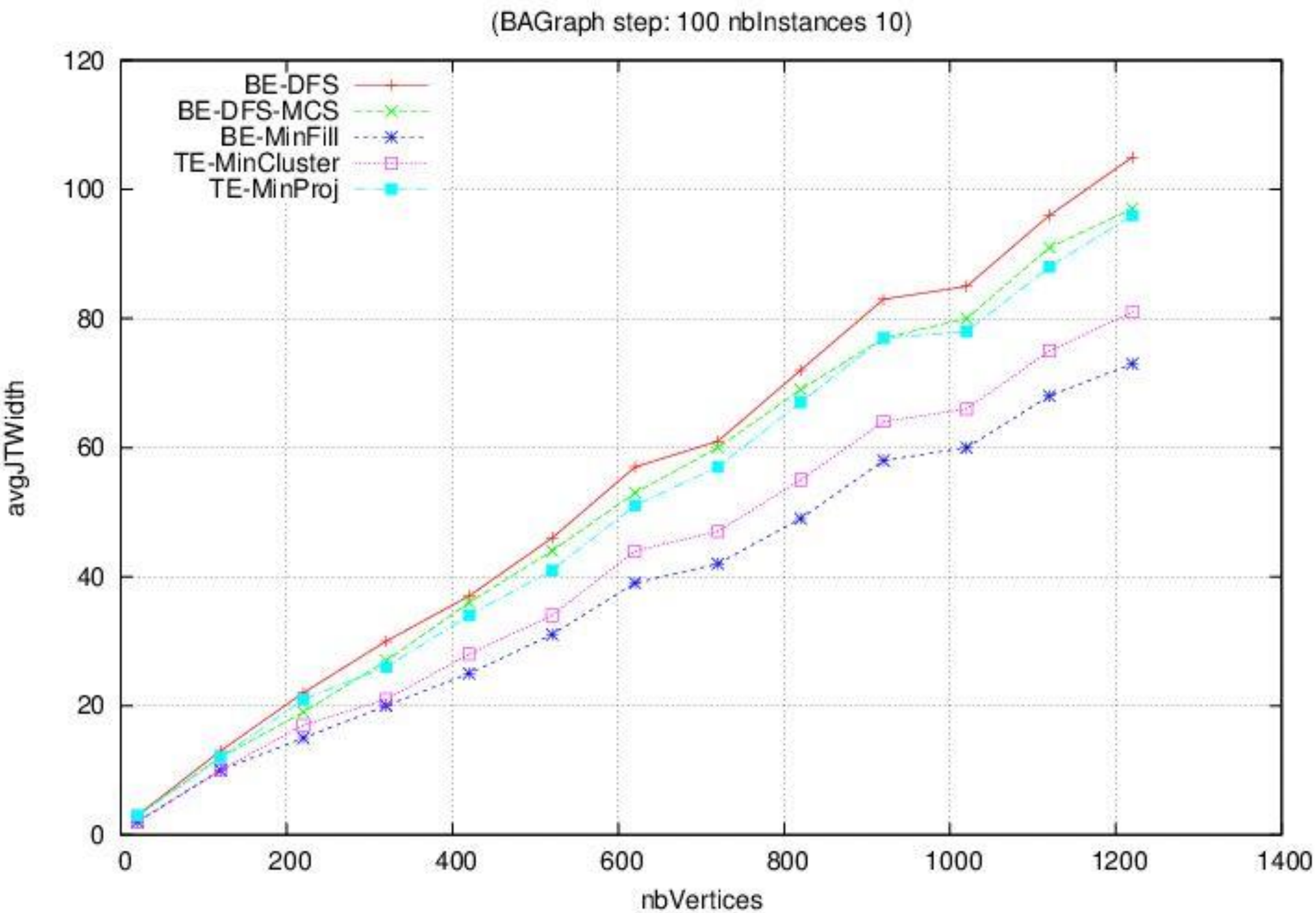
Tree Decomposition of small world graphs

Barabasi et Albert (B.A.) graphs



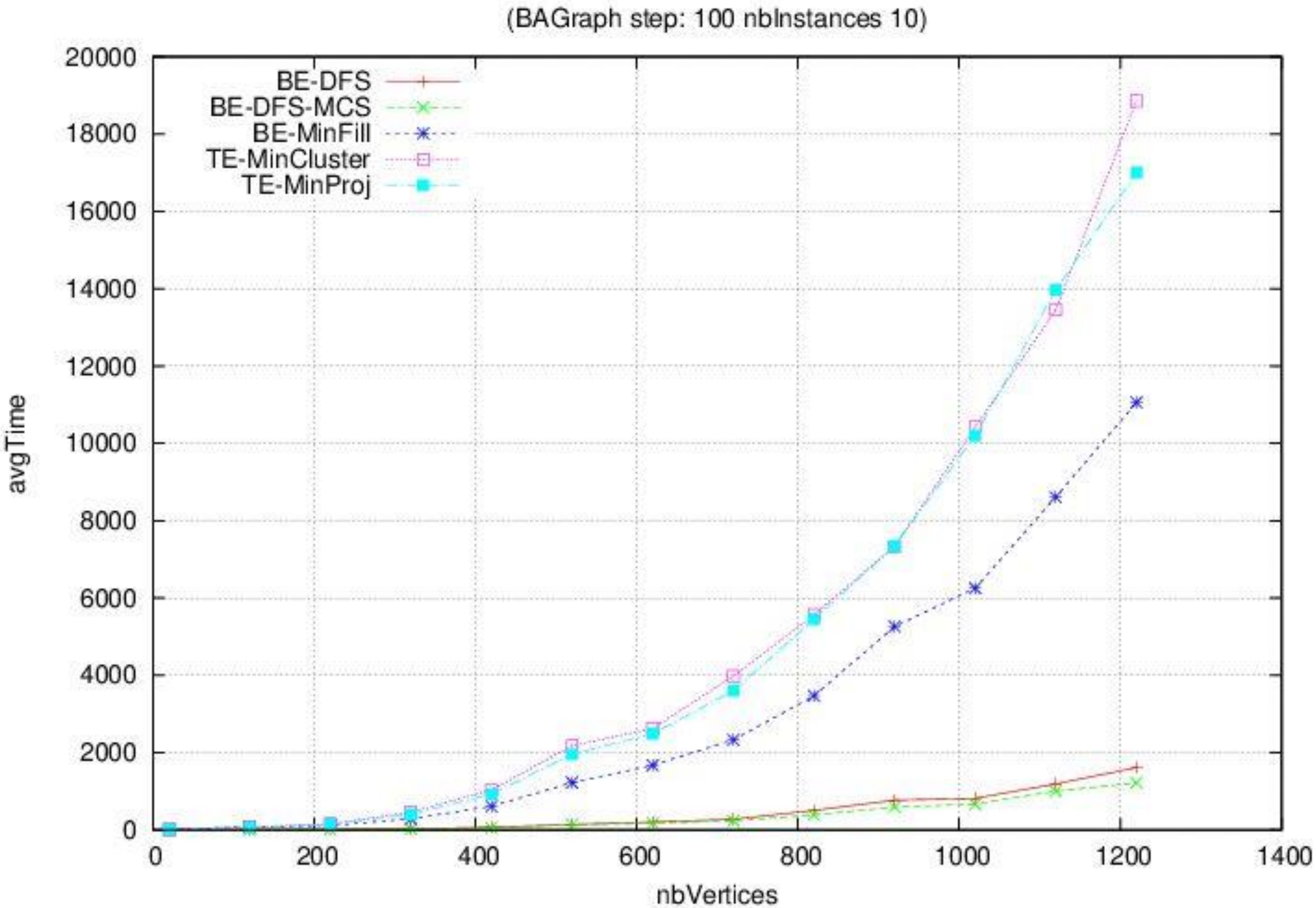
- Properties
  - low average distance between 2 nodes
  - heterogeneity (degree distribution follows a power law)
  - represents interaction graph of a lot of real world applications

width of tree decomposed BA Graphs

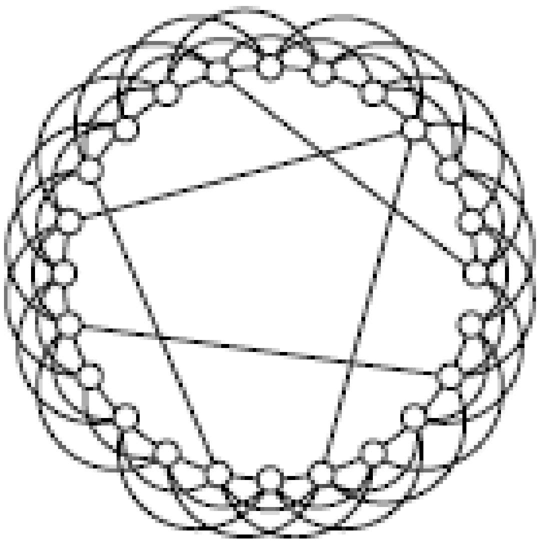


Tree Decomposition of small world graphs

CPU-Time of the tree decomposed BA Graphs



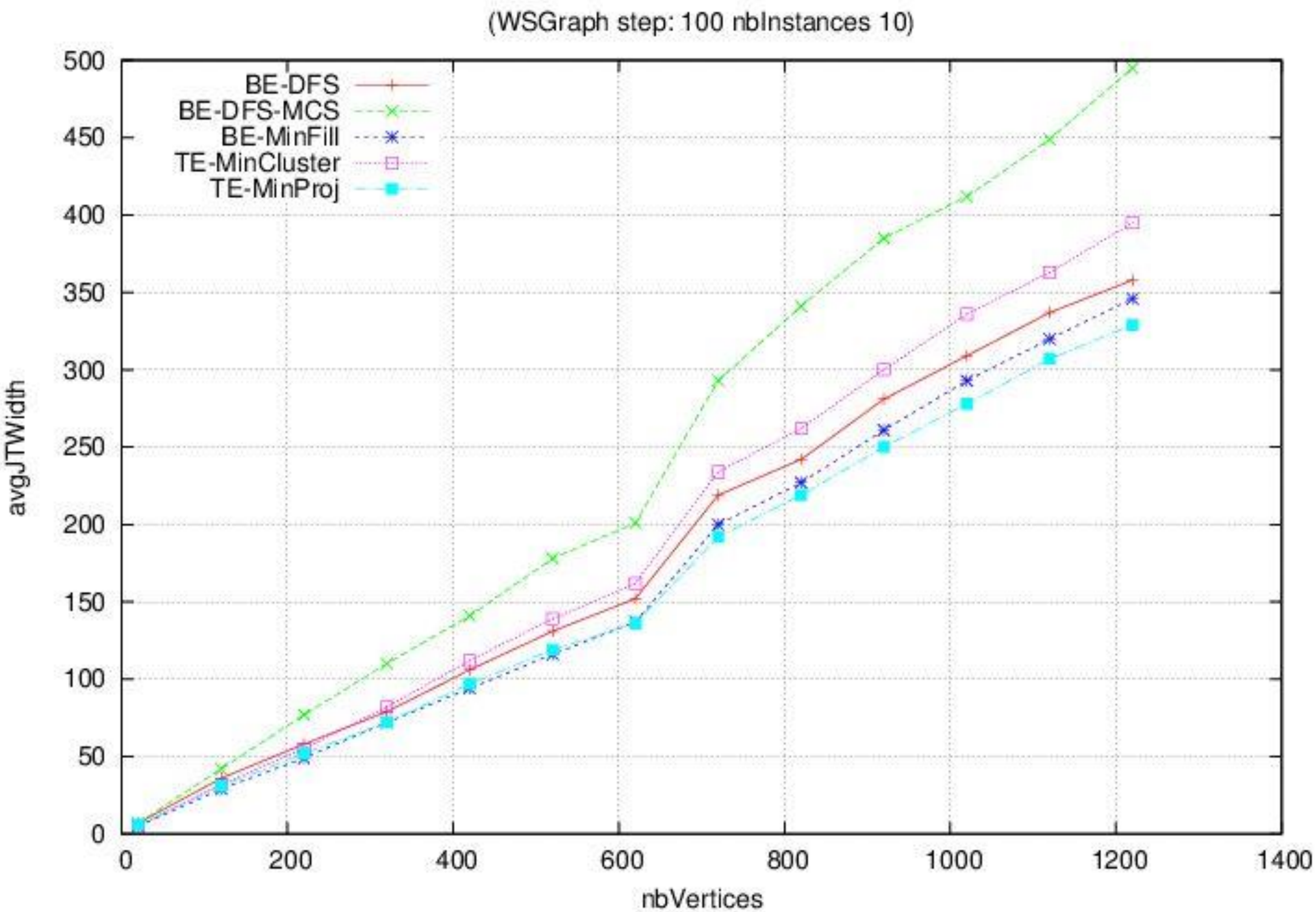
Watts et Strogatz (W.S.) graphs



- Properties
  - Short average distance between nodes
  - Homogenous (degree distribution follows Poisson law)
  - Represents some applications s.t. ISCAS circuits…

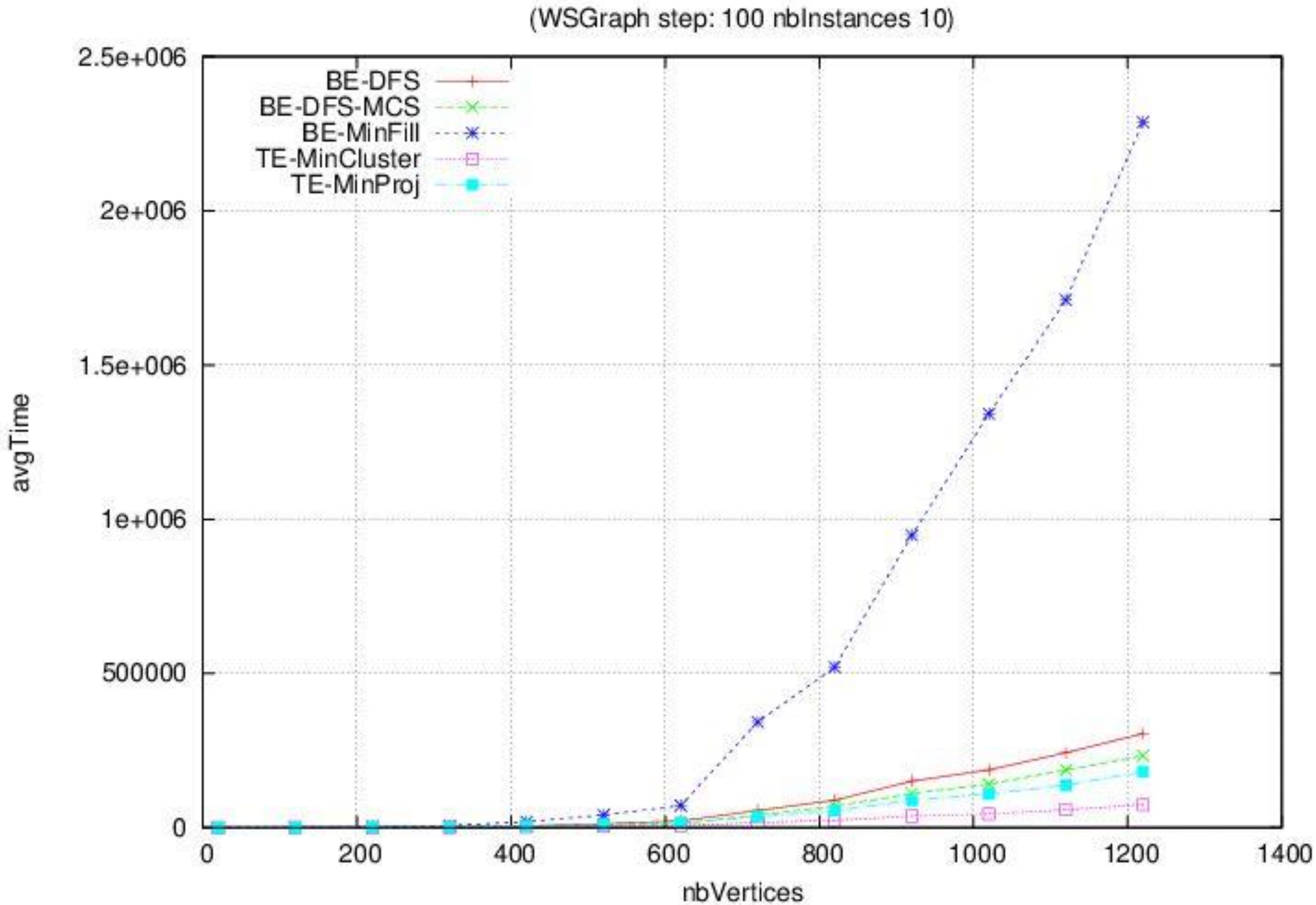
Tree Decomposition of small world graphs

width of the tree decomposed WS Graphs



Tree Decomposition of small world graphs

CPU time of the tree decomposed de WS graph



Conclusions

- Distributed Tree Decomposition respecting
  - privacy (main reason for distributed systems)
  - preserving network acquaintance
- Token Elimination relying
  - On elimination order
  - on votes, token passing
- Results: Token Elimination
  - outperforms classical distributed decomposition methods
  - is competitive with centralized methods

Thanks for your Attention ☺

– Questions?

– varmant@4c.ucc.ie