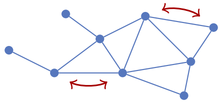
Distributed Maintenance of Anytime Available Spanning Trees in Dynamic Networks

A. Casteigts, S. Chaumette, F. Guinand, Y. Pigné

ADHOC-NOW'13

Wrocław, Poland

Distributed Computing



Collaboration of distinct entities to perform a common task.

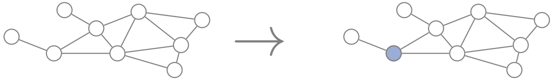
No centralization available. Direct interaction only.

(Think globally, act locally)

Examples of distributed problems

Leader election

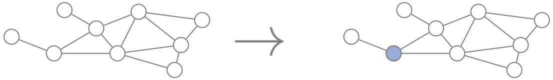
Distinguishing exactly one node among all.



Examples of distributed problems

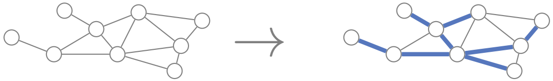
Leader election

Distinguishing exactly one node among all.



Spanning tree

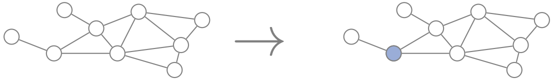
Selecting a cycle-free set of edges that interconnects all nodes.



Examples of distributed problems

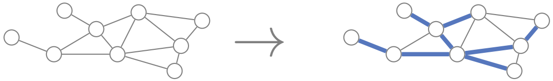
Leader election

Distinguishing exactly one node among all.



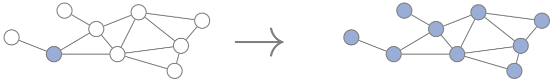
Spanning tree

Selecting a cycle-free set of edges that interconnects all nodes.



Broadcast

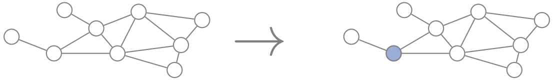
Propagating a piece of information from one node to all others.



Examples of distributed problems

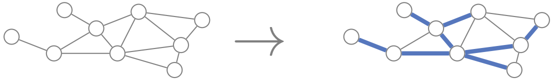
Leader election

Distinguishing exactly one node among all.



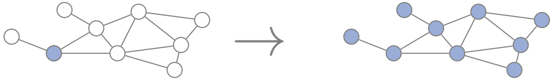
Spanning tree

Selecting a cycle-free set of edges that interconnects all nodes.



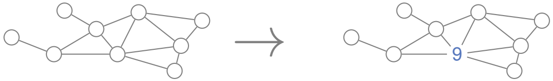
Broadcast

Propagating a piece of information from one node to all others.



Counting

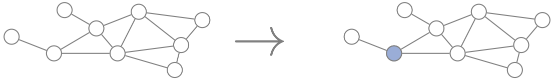
Determining how many participants there are.



Examples of distributed problems

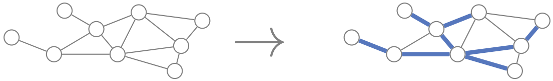
Leader election

Distinguishing exactly one node among all.



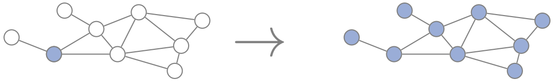
Spanning tree

Selecting a cycle-free set of edges that interconnects all nodes.



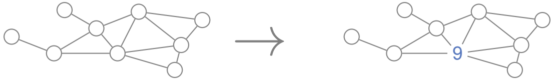
Broadcast

Propagating a piece of information from one node to all others.



Counting

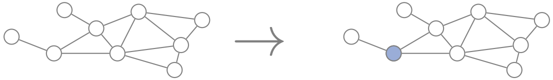
Determining how many participants there are.



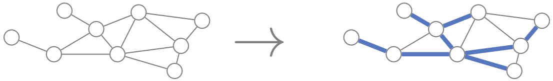
Consensus, naming, routing, exploration, ...

Examples of distributed problems

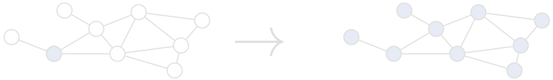
Leader election — Distinguishing exactly one node among all.



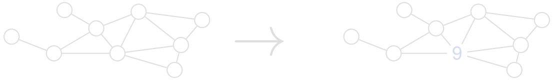
Spanning tree — Selecting a cycle-free set of edges that interconnects all nodes.



Broadcast — Propagating a piece of information from one node to all others.

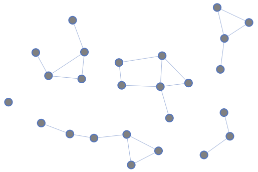


Counting — Determining how many participants there are.



Consensus, naming, routing, exploration, ...

Dynamic Networks



Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

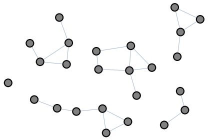
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Network is partitioned most of the time.



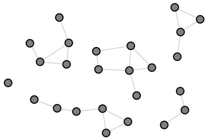
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

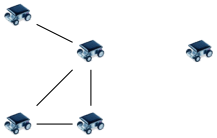
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Network is partitioned most of the time.



Example of scenario

(say, exploration by mobile robots)



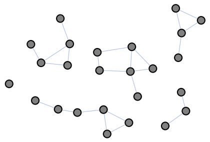
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

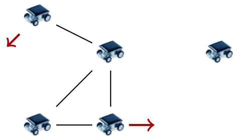
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Network is partitioned most of the time.



Example of scenario

(say, exploration by mobile robots)



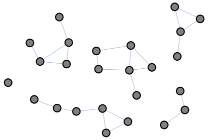
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

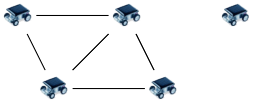
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Network is partitioned most of the time.



Example of scenario

(say, exploration by mobile robots)



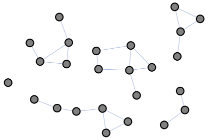
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

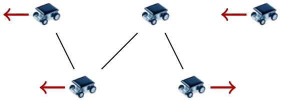
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Network is partitioned most of the time.



Example of scenario

(say, exploration by mobile robots)



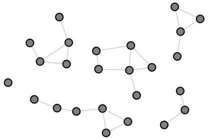
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

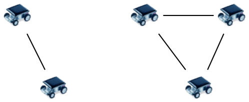
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Network is partitioned most of the time.



Example of scenario

(say, exploration by mobile robots)



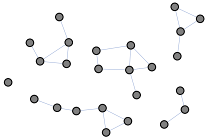
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

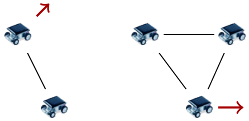
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Network is partitioned most of the time.



Example of scenario

(say, exploration by mobile robots)



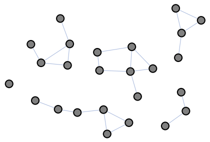
Dynamic networks ?

In fact, *highly* dynamic networks.

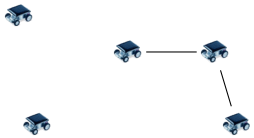
Ex :

How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Network is partitioned most of the time.

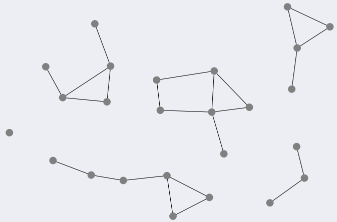


Example of scenario
(say, exploration by mobile robots)



Distributed problems in highly dynamic networks ?

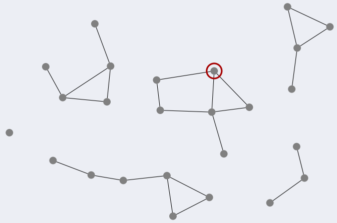
Ex : Election, SpanningTree



How to define them ?

Distributed problems in highly dynamic networks ?

Ex : Election, SpanningTree



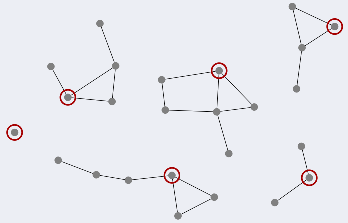
How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)

Distributed problems in highly dynamic networks ?

Ex : Election, SpanningTree



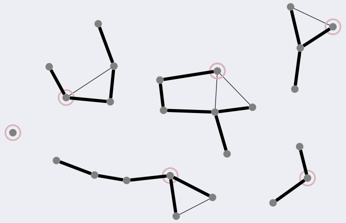
How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

Distributed problems in highly dynamic networks ?

Ex : Election, SpanningTree



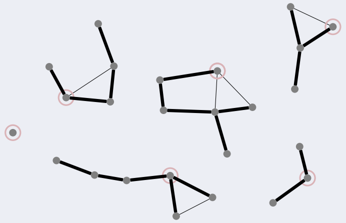
How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

Distributed problems in highly dynamic networks ?

Ex : Election, SpanningTree



How to define them ?

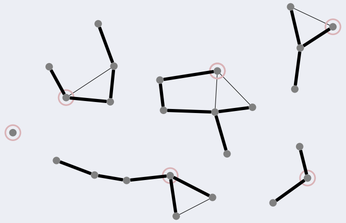
→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

→ Maintenance problem.

Distributed problems in highly dynamic networks ?

Ex : ELECTION, SPANNINGTREE



How to define them ?

→ Two options :

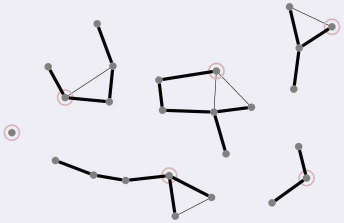
1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

→ Maintenance problem.

What assumptions ?

Distributed problems in highly dynamic networks ?

Ex : ELECTION, SPANNINGTREE



How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

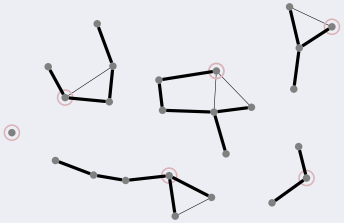
→ Maintenance problem.

What assumptions ?

- No stability period

Distributed problems in highly dynamic networks ?

Ex : ELECTION, SPANNINGTREE



How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

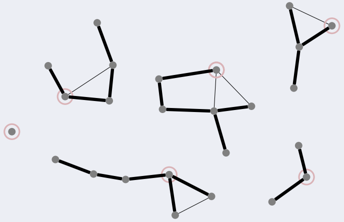
→ Maintenance problem.

What assumptions ?

- No stability period
- No restriction on the rate of events

Distributed problems in highly dynamic networks ?

Ex : Election, SpanningTree



How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

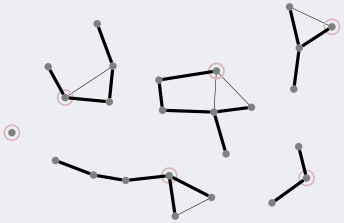
→ Maintenance problem.

What assumptions ?

- No stability period
- No restriction on the rate of events
- No unique identifiers (only local orientation)

Distributed problems in highly dynamic networks ?

Ex : Election, SpanningTree



How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

→ Maintenance problem.

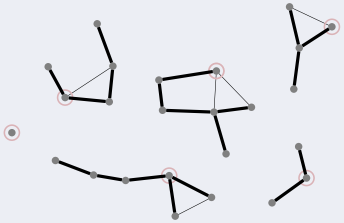
What assumptions ?

- No stability period
- No restriction on the rate of events
- No unique identifiers (only local orientation)

This work : what can we still expect in such a setting ?

Distributed problems in highly dynamic networks ?

Ex : Election, SpanningTree



How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

→ Maintenance problem.

What assumptions ?

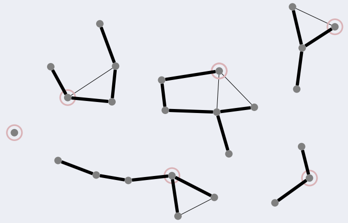
- No stability period
- No restriction on the rate of events
- No unique identifiers (only local orientation)

$\}$ No recomputation from scratch

This work : what can we still expect in such a setting ?

Distributed problems in highly dynamic networks ?

Ex : ELECTION, SPANNINGTREE



How to define them ?

→ Two options :

1. Global solution, which holds over time (computed once)
2. Local solutions, which hold anytime (constantly maintained)

→ Maintenance problem.

What assumptions ?

- No stability period
- No restriction on the rate of events
- No unique identifiers (only local orientation)

} No recomputation from scratch

} Decision should be purely local!

This work : what can we still expect in such a setting ?

Computational model

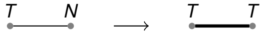
Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ;
$\rightarrow$ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ;

→ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*

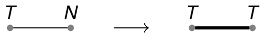


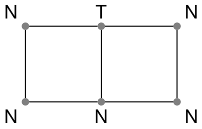
Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)*; Population protocols *(Angluin et al., 2004)*)

$\rightarrow$ Pairwise atomic interaction

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*



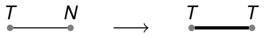


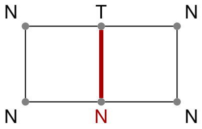
Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)*;

→ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*

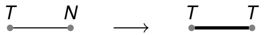


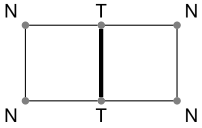


Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ;
→ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*



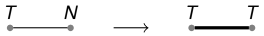


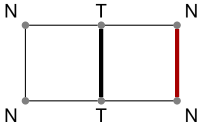
Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ;

→ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*

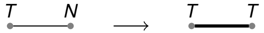


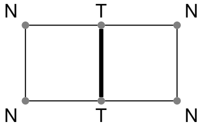


Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ;
→ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*



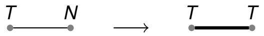


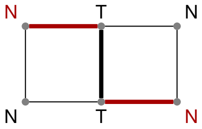
Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ; Population protocols *(Angluin et al., 2004)*)

→ Pairwise atomic interaction

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*



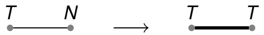


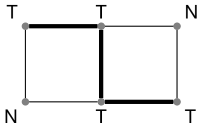
Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)*;

→ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*





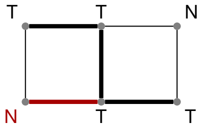
Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ; Population protocols *(Angluin et al., 2004)*)

$\rightarrow$ Pairwise atomic interaction

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*

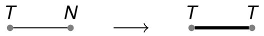
$$T \text{---} N \longrightarrow T \text{===} T$$

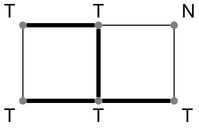


Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ;
$\rightarrow$ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*



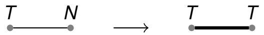


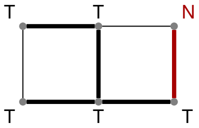
Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ;

→ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*



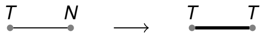


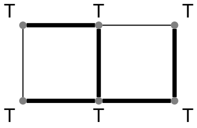
Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)* ;

$\rightarrow$ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*





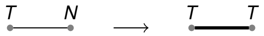
Note : Scheduling is not part of the algorithm !

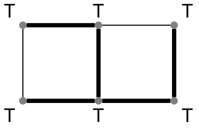
$\rightarrow$ It is, e.g., probabilistic, adversarial, or even abstracted.

Computational model

Coarse-grain model (Graph relabeling systems *(Litovsky et al., 1999)*;
→ Pairwise atomic interaction Population protocols *(Angluin et al., 2004)*)

Ex : *Spanning tree algorithm in a static network, with a leader initially labeled T.*

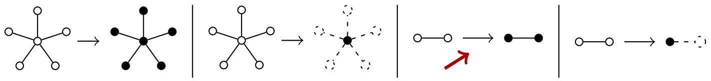




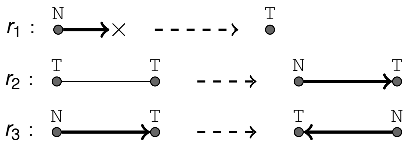
Note : Scheduling is not part of the algorithm !

→ It is, e.g., probabilistic, adversarial, or even abstracted.

Scope of the models Relations between them *(Chalopin, 2006)*



The spanning forest algorithm



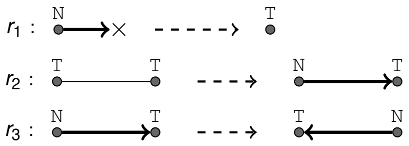
initial states :

- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent

The spanning forest algorithm

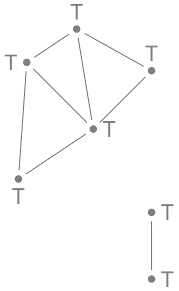


initial states :

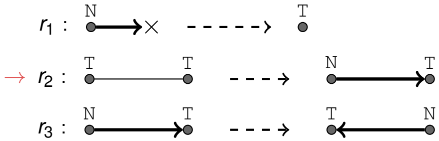
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

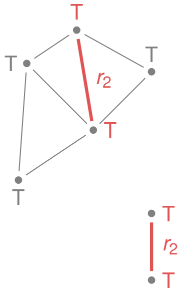


initial states :

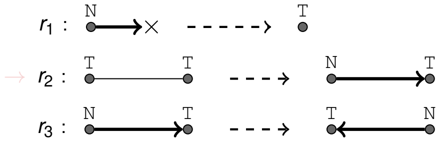
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

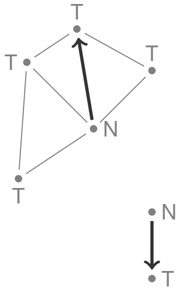


initial states :

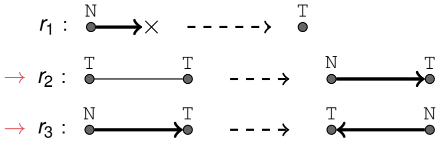
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

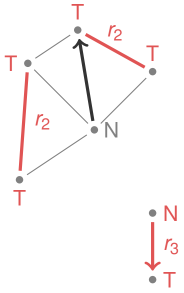


initial states :

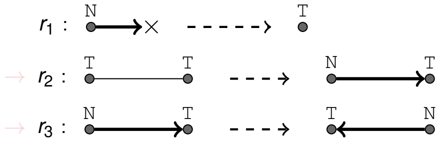
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

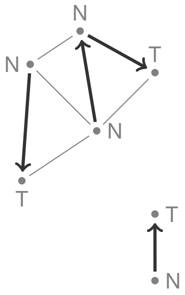


initial states :

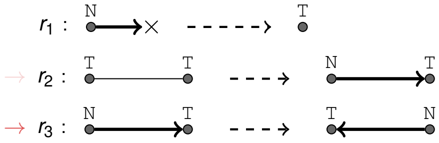
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

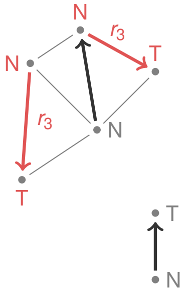


initial states :

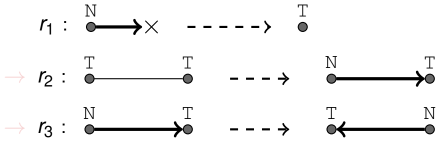
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

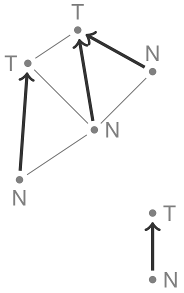


initial states :

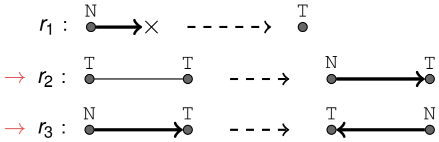
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

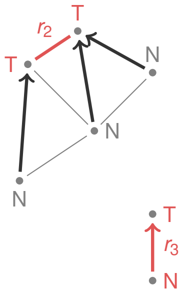


initial states :

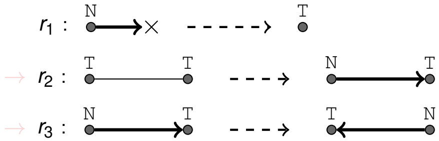
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- → : relation from child to parent



The spanning forest algorithm

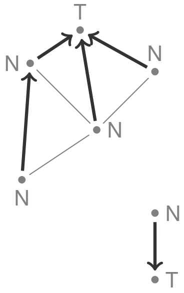


initial states :

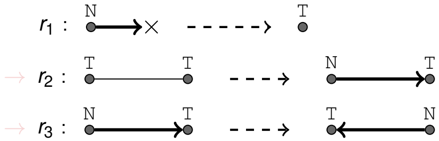
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

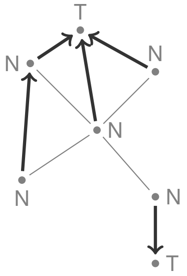


initial states :

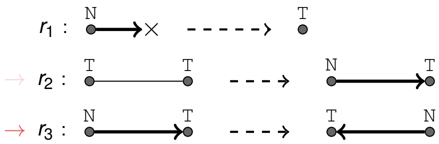
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

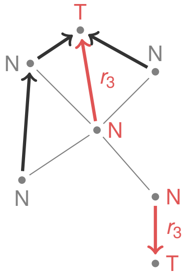


initial states :

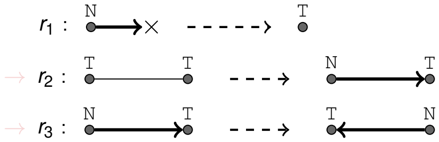
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

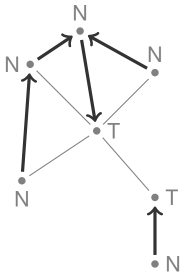


initial states :

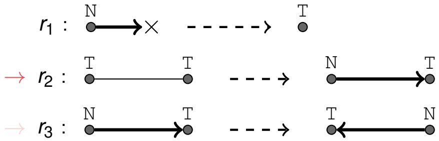
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

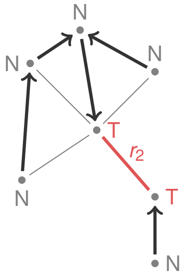


initial states :

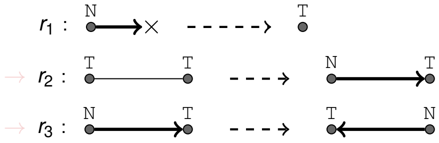
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

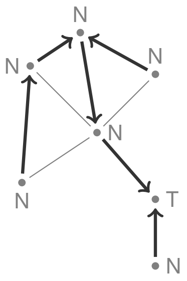


initial states :

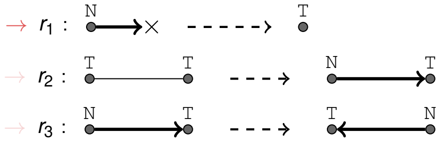
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

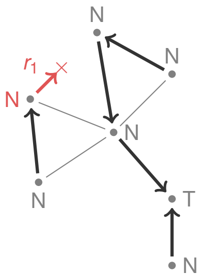


initial states :

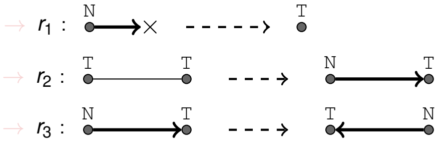
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

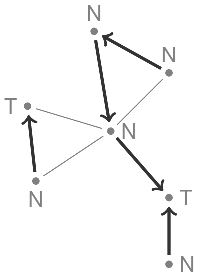


initial states :

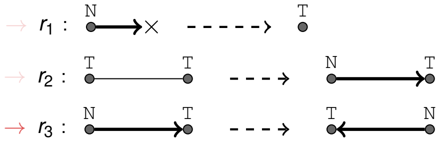
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

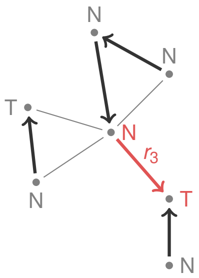


initial states :

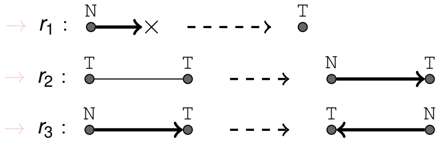
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

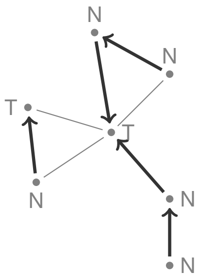


initial states :

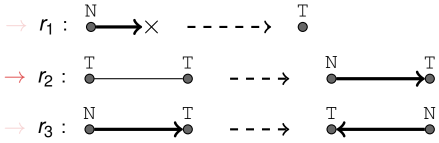
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

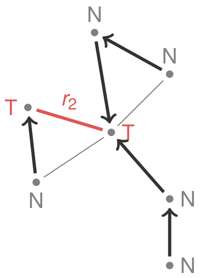


initial states :

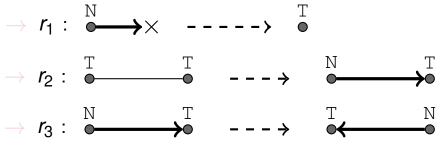
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

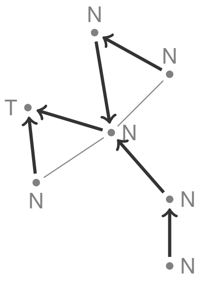


initial states :

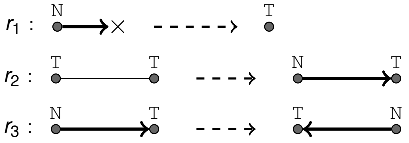
- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



The spanning forest algorithm

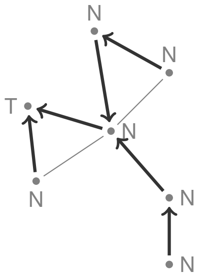


initial states :

- T for every node,

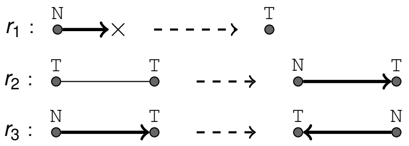
meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



Properties that hold permanently :

The spanning forest algorithm

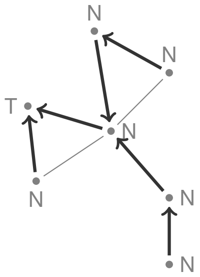


initial states :

- T for every node,

meaning of the states :

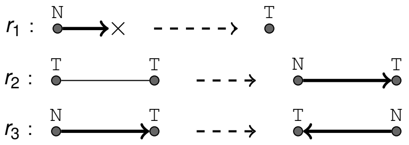
- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



Properties that hold permanently :

- Each node belongs to exactly one tree

The spanning forest algorithm

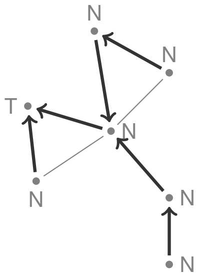


initial states :

- T for every node,

meaning of the states :

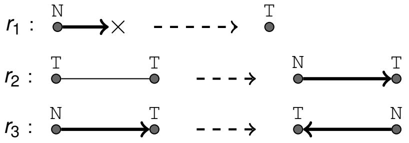
- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



Properties that hold permanently :

- Each node belongs to exactly one tree
- There is exactly one token per tree

The spanning forest algorithm

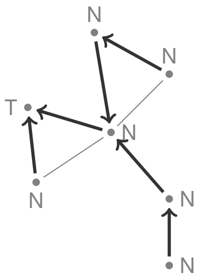


initial states :

- T for every node,

meaning of the states :

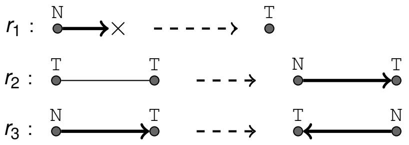
- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



Properties that hold permanently :

- Each node belongs to exactly one tree
- There is exactly one token per tree
- There are no cycles

The spanning forest algorithm

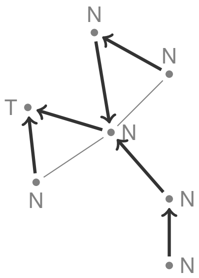


initial states :

- T for every node,

meaning of the states :

- T : a token is on this node
- N : no token is on this node
- $\rightarrow$: relation from child to parent



Properties that hold permanently :

- Each node belongs to exactly one tree
- There is exactly one token per tree
- There are no cycles

How about performance ?

Performance analysis (mostly open)

Metric of interest ?

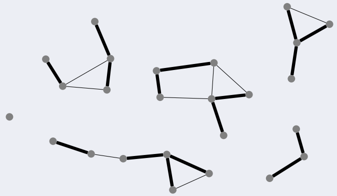
1. Convergence rate
   (though not expected to converge)

Performance analysis (mostly open)

Metric of interest ?

1. Convergence rate
   (though not expected to converge)
2. Average quality
   e.g. $9/6 = 1.5$ there $\rightarrow$

   ($NbTrees/NbConnectedComponents$)

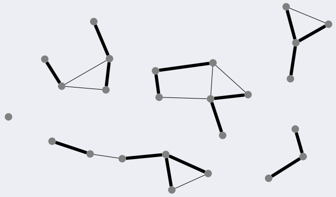


Performance analysis (mostly open)

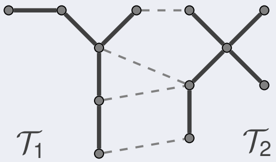
Metric of interest ?

1. Convergence rate
   (though not expected to converge)
2. Average quality
   e.g. $9/6 = 1.5$ there $\rightarrow$

   $(NbTrees/NbConnectedComponents)$



Preliminary elements (merging time between two static trees)



$$\left(\sum_{(u,v)\in Bridges(\mathcal{T}_1,\mathcal{T}_2)} \mathbb{P}(\lambda(u)=\mathrm{T} \wedge \lambda(v)=\mathrm{T})\right)^{-1}$$

Perspectives on ...

... performance evaluation

Perspectives on ...

... performance evaluation

- Analysis
  (coalescing particles in evolving graphs)

Perspectives on ...

... performance evaluation

- Analysis
  (coalescing particles in evolving graphs)
- Simulations
  (some are in the 2009 technical report, much more needed)

Perspectives on ...

... performance evaluation

- Analysis
  (coalescing particles in evolving graphs)
- Simulations
  (some are in the 2009 technical report, much more needed)

... algorithmic aspects

Perspectives on ...

... performance evaluation

- Analysis
  (coalescing particles in evolving graphs)
- Simulations
  (some are in the 2009 technical report, much more needed)

... algorithmic aspects

- Finer-grain adaptation of the principle
  - Synchronous message passing (OK)
  - Semi-synchronous message passing (under study)

Perspectives on ...

... performance evaluation

- Analysis
  (coalescing particles in evolving graphs)
- Simulations
  (some are in the 2009 technical report, much more needed)

... algorithmic aspects

- Finer-grain adaptation of the principle
  - Synchronous message passing (OK)
  - Semi-synchronous message passing (under study)
- Optimization strategies
  (*e.g.* from simple *Tabou* search to full *Propp* machines)

Thank you !

References :

I. Litovsky, Y. Métivier, and E. Sopena., Graph relabelling systems and distributed algorithms., *Handbook of graph grammars and computing by graph transformation, 1999.*

D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, R. Peralta, Computation in networks of passively mobile finite-state sensors, PODC, 2004.

J. Chalopin, Algorithmique Distribuée, Calculs Locaux et Homomorphismes de Graphes, PhD Thesis, University of Bordeaux, 2006.

A. Casteigts, S. Chaumette, F. Guinand, P. Pigné., Distributed Maintenance of Anytime Available Spanning Trees in Dynamic Networks, CoRR,abs/0904.3087v1, 2009.

M.Y. Neggaz., Message-based Adaptation of a Spanning Forest Algorithm in Highly Dynamic Networks, Master's Thesis, 2013.