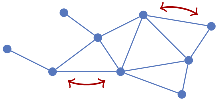
Distributed Computing in Dynamic Networks
–
Impact of the dynamics on definitions and feasibility

Arnaud Casteigts
–
University of Bordeaux

Séminaire Systèmes Complexes

Mai 2014, Paris 6

Distributed Computing



Collaboration of distinct entities to perform a common task.

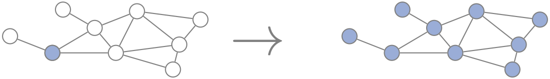
No centralization available. Direct interaction.

(Think globally, act locally)

Examples of problems

Broadcast

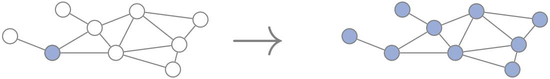
Propagating a piece of information from one node to all others.



Examples of problems

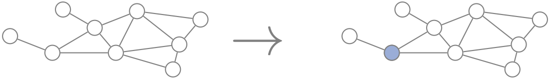
Broadcast

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



Election

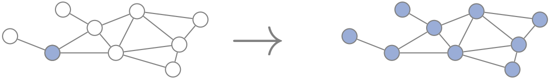
Distinguishing exactly one node among all.



Examples of problems

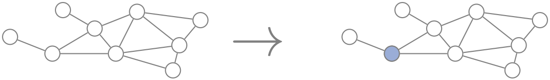
Broadcast

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



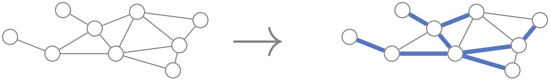
Election

Distinguishing exactly one node among all.



Spanning tree

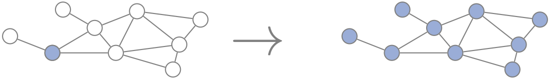
Selecting a cycle-free set of edges that interconnects all nodes.



Examples of problems

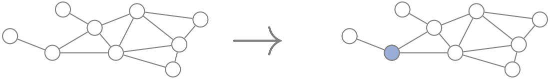
Broadcast

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



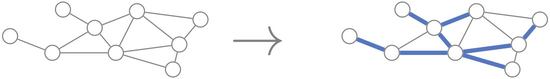
Election

Distinguishing exactly one node among all.



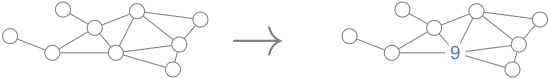
Spanning tree

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



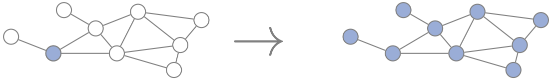
Counting

Determining how many participants there are.

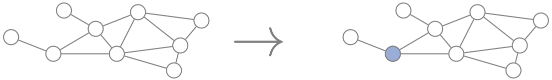


Examples of problems

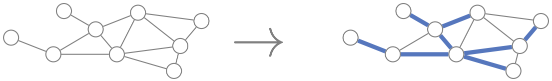
Broadcast Propagating a piece of information from one node to all others.



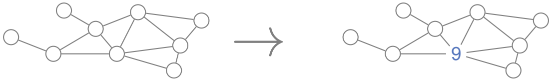
Election Distinguishing exactly one node among all.



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



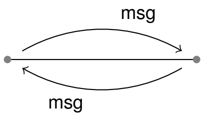
Counting Determining how many participants there are.



Consensus, naming, routing, exploration, ...

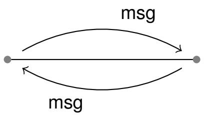
Communication Models

Communication Models

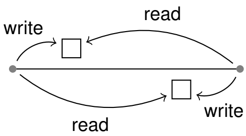


(a) Message passing

Communication Models

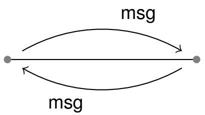


(a) Message passing

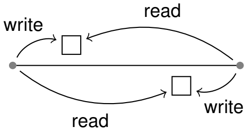


(b) Registers

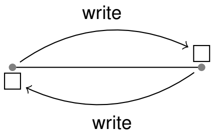
Communication Models



(a) Message passing

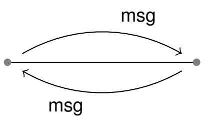


(b) Registers

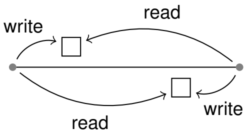


(c) Mailboxes

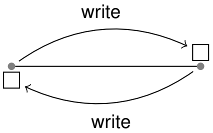
Communication Models



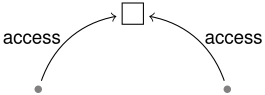
(a) Message passing



(b) Registers

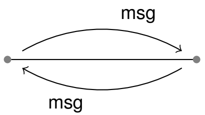


(c) Mailboxes

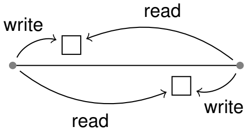


(d) Shared memory

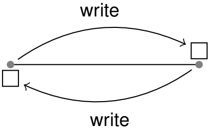
Communication Models



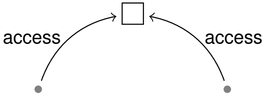
(a) Message passing



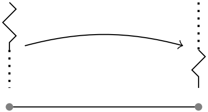
(b) Registers



(c) Mailboxes



(d) Shared memory



(e) Mobile agents

Abstracting Communications

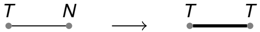
Atomic interaction (Population protocols *(Angluin et al., 2004)*; Graph relabeling systems *(Litovsky et al., 1999)*)

Abstracting Communications

Atomic interaction

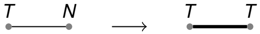
(Population protocols *(Angluin et al., 2004)*; Graph relabeling systems *(Litovsky et al., 1999)*)

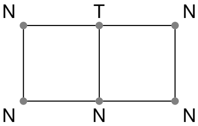
Ex :



Atomic interaction

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

Ex : 

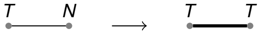


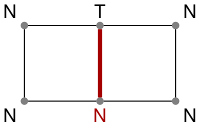
Abstracting Communications

Atomic interaction

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

Ex :

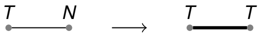


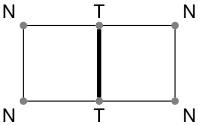


Abstracting Communications

Atomic interaction

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

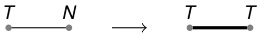
Ex : 

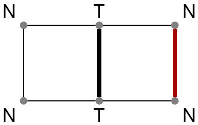


Atomic interaction

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

Ex :

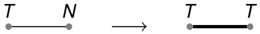


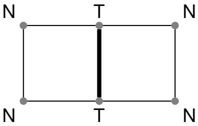


Abstracting Communications

Atomic interaction

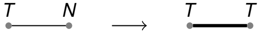
(Population protocols *(Angluin et al., 2004)*; Graph relabeling systems *(Litovsky et al., 1999)*)

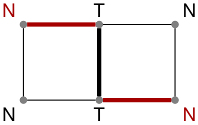
Ex : 



Atomic interaction

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

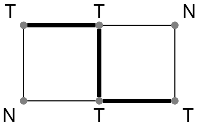
Ex : 



Atomic interaction

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

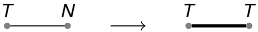
Ex : T —— N $\longrightarrow$ T —— T

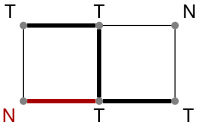


Abstracting Communications

Atomic interaction

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

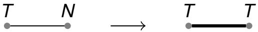
Ex : 

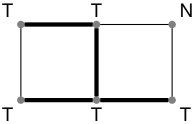


Abstracting Communications

Atomic interaction

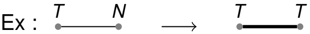
(Population protocols *(Angluin et al., 2004)* ; Graph relabeling systems *(Litovsky et al., 1999)*)

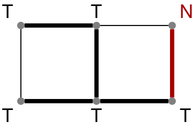
Ex : 



Atomic interaction

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

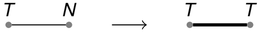


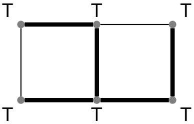


Abstracting Communications

Atomic interaction (Population protocols *(Angluin et al., 2004)*; Graph relabeling systems *(Litovsky et al., 1999)*)

Ex :

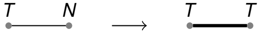


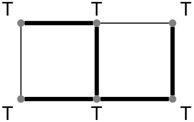


Abstracting Communications

Atomic interaction

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

Ex : 

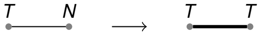


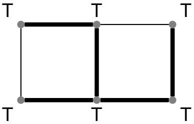
Note : Scheduling is not part of the algorithm !

→ Can be adversarial, randomized, etc.

Abstracting Communications

Atomic interaction (Population protocols *(Angluin et al., 2004)*; Graph relabeling systems *(Litovsky et al., 1999)*)

Ex : 



Note : Scheduling is not part of the algorithm !

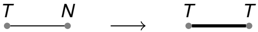
→ Can be adversarial, randomized, etc.

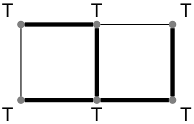
Scope of the models Relations between them *(Chalopin, 2006)*

Abstracting Communications

Atomic interaction (Population protocols *(Angluin et al., 2004)*; Graph relabeling systems *(Litovsky et al., 1999)*)

Ex :

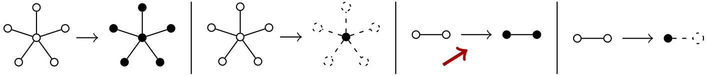




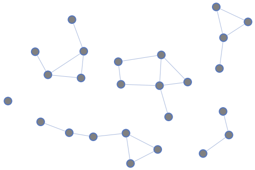
Note : Scheduling is not part of the algorithm !

→ Can be adversarial, randomized, etc.

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



Dynamic Networks



In fact, *highly* dynamic networks.

Ex : 

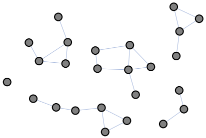
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex :

How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Partitioned network

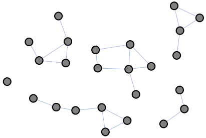


Dynamic networks ?

In fact, *highly* dynamic networks.

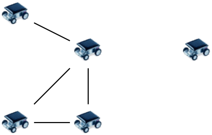
Ex : 

How changes are perceived ?



- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Partitioned network

Example of scenario
(say, exploration by mobile robots)



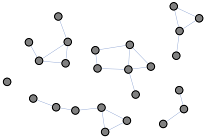
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex : 

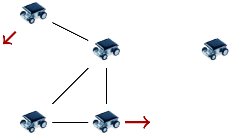
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Partitioned network



Example of scenario

(say, exploration by mobile robots)

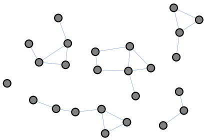


Dynamic networks ?

In fact, *highly* dynamic networks.

Ex : 

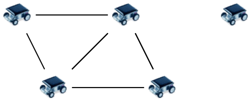
How changes are perceived ?



- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Partitioned network

Example of scenario

(say, exploration by mobile robots)



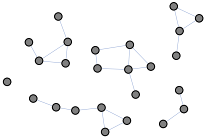
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex : 

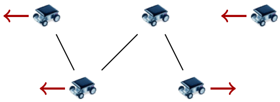
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Partitioned network



Example of scenario

(say, exploration by mobile robots)



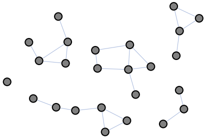
Dynamic networks ?

In fact, *highly* dynamic networks.

Ex : 

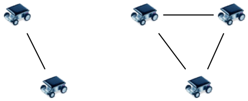
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Partitioned network



Example of scenario

(say, exploration by mobile robots)



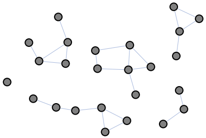
Dynamic networks ?

In fact, *highly* dynamic networks.

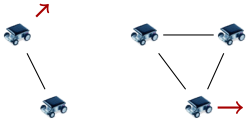
Ex : 

How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Partitioned network



Example of scenario
(say, exploration by mobile robots)

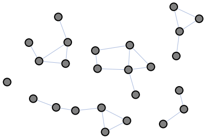


In fact, *highly* dynamic networks.

Ex : 

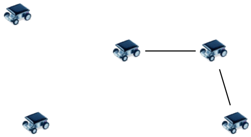
How changes are perceived ?

- ~~Faults and Failures ?~~
- Nature of the system. Change is normal.
- Partitioned network



Example of scenario

(say, exploration by mobile robots)



Dynamic Graphs

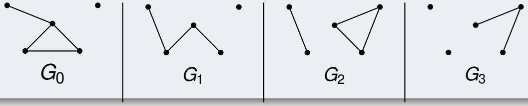
Also called *evolving graphs* or *time-varying graphs*.

Dynamic Graphs

Also called *evolving graphs* or *time-varying graphs*.

Global point of view

Sequence of static graphs $\mathcal{G} = G_0, G_1, ...$ [+table of dates in $\mathbb{T}$]

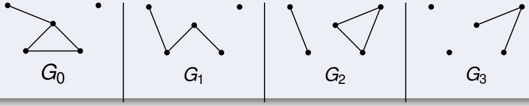


Dynamic Graphs

Also called *evolving graphs* or *time-varying graphs*.

Global point of view

Sequence of static graphs $\mathcal{G} = G_0, G_1, \ldots$ [+table of dates in $\mathbb{T}$]

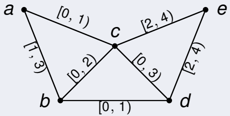


Local point of view

$\mathcal{G} = (V, E, \mathcal{T}, \rho)$,

with ρ being a *presence function*

$\rho : E \times \mathcal{T} \to \{0, 1\}$

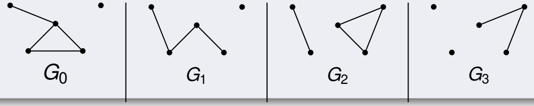


Dynamic Graphs

Also called *evolving graphs* or *time-varying graphs*.

Global point of view

Sequence of static graphs $\mathcal{G} = G_0, G_1, \ldots$ [+table of dates in $\mathbb{T}$]

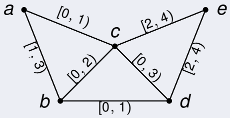


Local point of view

$\mathcal{G} = (V, E, \mathcal{T}, \rho)$,

with ρ being a *presence function*

$\rho : E \times \mathcal{T} \to \{0, 1\}$



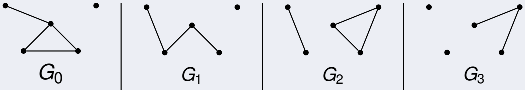
$\rightarrow$ Both are theoretically equivalent if ρ is countable (e.g. not like this )

Dynamic Graphs

Also called *evolving graphs* or *time-varying graphs*.

Global point of view

Sequence of static graphs $\mathcal{G} = G_0, G_1, ...$ [+table of dates in $\mathbb{T}$]

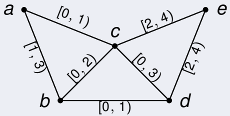


Local point of view

$\mathcal{G} = (V, E, \mathcal{T}, \rho)$,

with ρ being a *presence function*

$\rho : E \times \mathcal{T} \rightarrow \{0, 1\}$



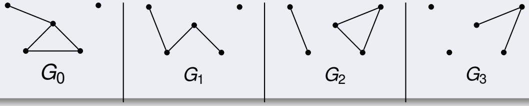
$\rightarrow$ Both are theoretically equivalent if ρ is countable (e.g. not like this )

$\rightarrow$ Further extensions possible (latency function, node-presence function, ...)

Also called *evolving graphs* or *time-varying graphs*.

Global point of view

Sequence of static graphs $\mathcal{G} = G_0, G_1, ...$ [+table of dates in $\mathbb{T}$]

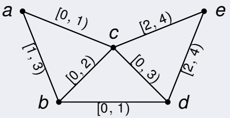


Local point of view

$\mathcal{G} = (V, E, \mathcal{T}, \rho)$,

with ρ being a *presence function*

$\rho : E \times \mathcal{T} \to \{0, 1\}$

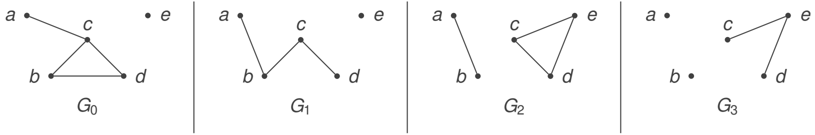


$\to$ Both are theoretically equivalent if ρ is countable (e.g. not like this )

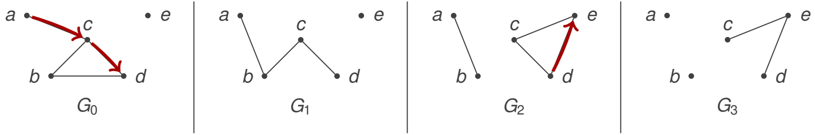
$\to$ Further extensions possible (latency function, node-presence function, ...)

For references, see *(Ferreira, 2004)* and *(C., Flocchini, Quattrociocchi, Santoro, 2012)*

Basic graph concepts



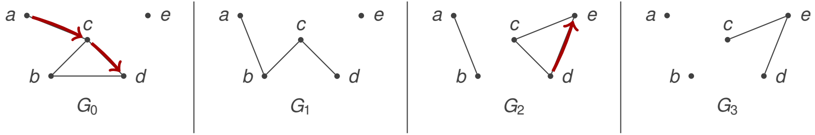
Basic graph concepts



$\Longrightarrow$ Paths become temporal (*journey*)

Ex : $((ac, t_1), (cd, t_2), (de, t_3))$ with $t_{i+1} \geq t_i$ and $\rho(e_i, t_i) = 1$

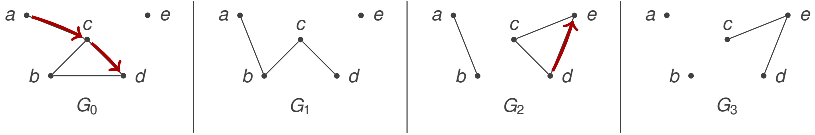
Basic graph concepts



$\Longrightarrow$ Paths become temporal (*journey*)

Ex : $((ac, t_1), (cd, t_2), (de, t_3))$ with $t_{i+1} \geq t_i$ and $\rho(e_i, t_i) = 1$

$\Longrightarrow$ *Temporal connectivity*. Not symmetrical ! (e.g. $a \rightsquigarrow e$, but $e \not\rightsquigarrow a$)



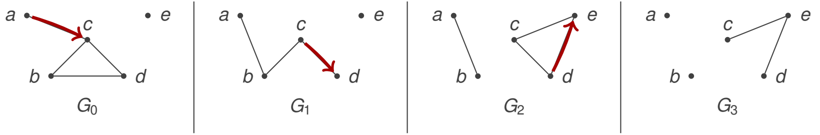
$\Longrightarrow$ Paths become temporal (*journey*)

Ex : $((ac, t_1), (cd, t_2), (de, t_3))$ with $t_{i+1} \geq t_i$ and $\rho(e_i, t_i) = 1$

$\Longrightarrow$ *Temporal connectivity*. Not symmetrical ! (e.g. $a \rightsquigarrow e$, but $e \not\rightsquigarrow a$)

$\Longrightarrow$ *Strict* journeys *vs.* *non-strict* journeys. (Important for analysis.)

Basic graph concepts



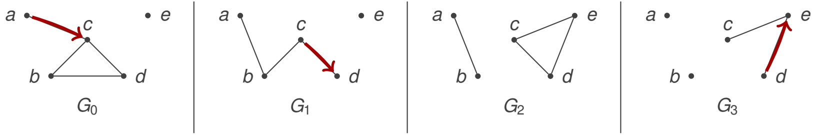
$\Longrightarrow$ Paths become temporal (*journey*)

Ex : $((ac, t_1), (cd, t_2), (de, t_3))$ with $t_{i+1} \geq t_i$ and $\rho(e_i, t_i) = 1$

$\Longrightarrow$ *Temporal connectivity*. Not symmetrical ! (e.g. $a \rightsquigarrow e$, but $e \not\rightsquigarrow a$)

$\Longrightarrow$ *Strict* journeys *vs.* *non-strict* journeys. (Important for analysis.)

Basic graph concepts



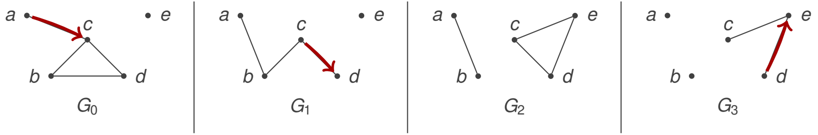
$\Longrightarrow$ Paths become temporal (*journey*)

Ex : $((ac, t_1), (cd, t_2), (de, t_3))$ with $t_{i+1} \geq t_i$ and $\rho(e_i, t_i) = 1$

$\Longrightarrow$ *Temporal connectivity*. Not symmetrical ! (e.g. $a \rightsquigarrow e$, but $e \not\rightsquigarrow a$)

$\Longrightarrow$ *Strict* journeys *vs.* *non-strict* journeys. (Important for analysis.)

Basic graph concepts



$\Longrightarrow$ Paths become temporal (*journey*)

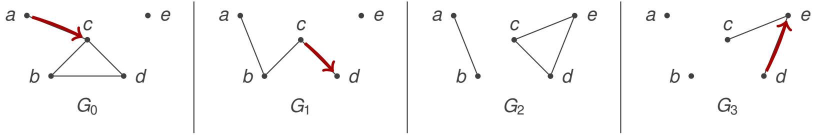
Ex : $((ac, t_1), (cd, t_2), (de, t_3))$ with $t_{i+1} \geq t_i$ and $\rho(e_i, t_i) = 1$

$\Longrightarrow$ *Temporal connectivity*. Not symmetrical ! (e.g. $a \rightsquigarrow e$, but $e \not\rightsquigarrow a$)

$\Longrightarrow$ *Strict* journeys *vs.* *non-strict* journeys. (Important for analysis.)

In the literature : *Schedule-conforming path (Berman, 1996)* ; *Time-respecting path (Kempe et al., 2008 ; Holme, 2005)* ; *Temporal path (Chaintreau et al., 2008)* ; *Journey (Bui-Xuan et al., 2003)*.

Basic graph concepts



$\Longrightarrow$ Paths become temporal (*journey*)

Ex : $((ac, t_1), (cd, t_2), (de, t_3))$ with $t_{i+1} \geq t_i$ and $\rho(e_i, t_i) = 1$

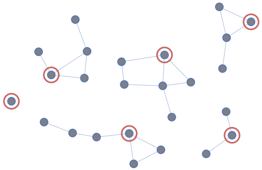
$\Longrightarrow$ *Temporal connectivity*. Not symmetrical ! (e.g. $a \leadsto e$, but $e \not\leadsto a$)

$\Longrightarrow$ *Strict* journeys *vs.* *non-strict* journeys. (Important for analysis.)

In the literature : *Schedule-conforming path (Berman, 1996)*; *Time-respecting path (Kempe et al., 2008 ; Holme, 2005)*; *Temporal path (Chaintreau et al., 2008)*; *Journey (Bui-Xuan et al., 2003)*.

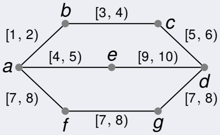
Many other concepts... (ask me !).

Re-definition of problems



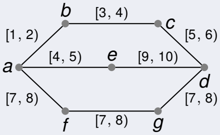
Re-definition of problems

Broadcast ?



Re-definition of problems

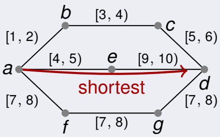
Broadcast ?



Which way is optimal from a to d ?

Re-definition of problems

Broadcast ?

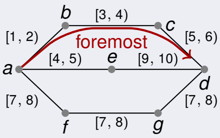


Which way is optimal from *a* to *d* ?

-min hop ?

Re-definition of problems

Broadcast ?



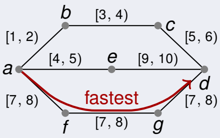
Which way is optimal from *a* to *d* ?

-min hop ?

-earliest arrival ?

Re-definition of problems

Broadcast ?



Which way is optimal from *a* to *d* ?

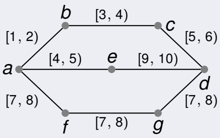
-min hop ?

-earliest arrival ?

-fastest traversal ?

Re-definition of problems

Broadcast ?



Which way is optimal from *a* to *d* ?

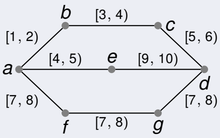
-min hop ?

-earliest arrival ?

-fastest traversal ?

Re-definition of problems

Broadcast ?



Which way is optimal from *a* to *d* ?

-min hop ?

-earliest arrival ?

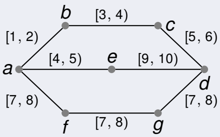
-fastest traversal ?

Associated problems

- Computing shortest, foremost and fastest journeys in dynamic networks (*centralized, with prior knowledge of the evolution, (Bui-Xuan, Jarry, Ferreira, 2003)*)

Re-definition of problems

Broadcast ?



Which way is optimal from *a* to *d* ?

-min hop ?

-earliest arrival ?

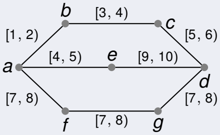
-fastest traversal ?

Associated problems

- Computing shortest, foremost and fastest journeys in dynamic networks (*centralized, with prior knowledge of the evolution, (Bui-Xuan, Jarry, Ferreira, 2003)*)
- Shortest, foremost and fastest broadcast in dynamic networks (*distributed, with partial knowledge, (C., Flocchini, Mans, Santoro, 2012)*)

Re-definition of problems

Broadcast ?



Which way is optimal from a to d ?

-min hop ?

-earliest arrival ?

-fastest traversal ?

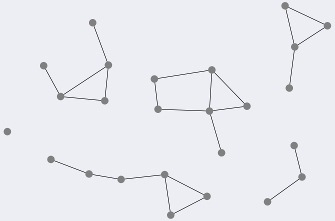
Associated problems

- Computing shortest, foremost and fastest journeys in dynamic networks (*centralized, with prior knowledge of the evolution, (Bui-Xuan, Jarry, Ferreira, 2003)*)
- Shortest, foremost and fastest broadcast in dynamic networks (*distributed, with partial knowledge, (C., Flocchini, Mans, Santoro, 2012)*)

Difficulty : Depends on the starting date → set of time-dependent solutions.

Re-definition of problems (2)

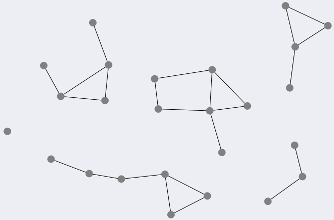
Ex : ELECTION, SPANNINGTREE, ...



How are they defined ?

Re-definition of problems (2)

Ex : ELECTION, SPANNINGTREE, ...

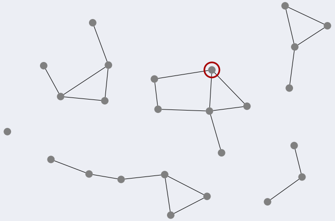


How are they defined ?

$\rightarrow$ Several options

Re-definition of problems (2)

Ex : ELECTION, SPANNINGTREE, ...



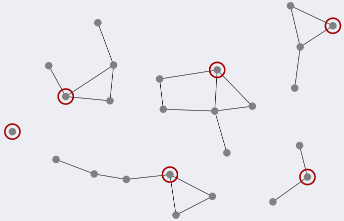
How are they defined ?

→ Several options

- One global leader, elected once and forever

Re-definition of problems (2)

Ex : Election, SpanningTree, ...



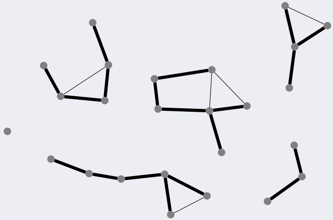
How are they defined ?

→ Several options

- One global leader, elected once and forever
- One leader per component, changing as the graph changes

Re-definition of problems (2)

Ex : ELECTION, SPANNINGTREE, ...



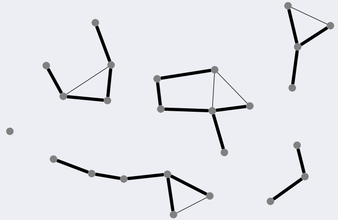
How are they defined ?

→ Several options

- One global leader, elected once and forever
- One leader per component, changing as the graph changes

Re-definition of problems (2)

Ex : Election, SpanningTree, ...



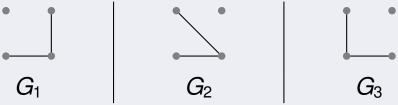
How are they defined ?

→ Several options

- One global leader, elected once and forever
- One leader per component, changing as the graph changes

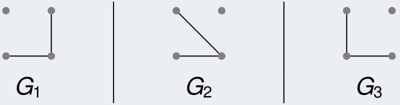
Both are very different in essence !

Ex : DominatingSet



Re-definition of problems (3) *(C., Mans, Mathieson, 2011)*

Ex : DominatingSet

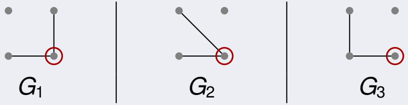


→ *Temporal* variant

Re-definition of problems (3)

(C., Mans, Mathieson, 2011)

Ex : DominatingSet

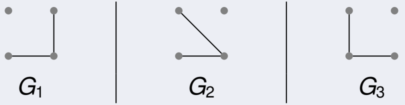


→ *Temporal* variant

Re-definition of problems (3)

(C., Mans, Mathieson, 2011)

Ex : DOMINATINGSET

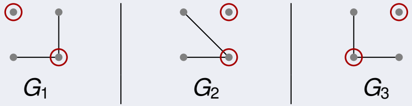


- → *Temporal* variant
- → *Evolving* variant

Re-definition of problems (3)

(C., Mans, Mathieson, 2011)

Ex : DOMINATINGSET



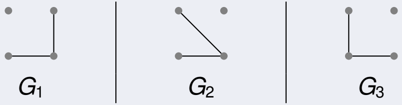
→ *Temporal* variant

→ *Evolving* variant

Re-definition of problems (3)

(C., Mans, Mathieson, 2011)

Ex : DominatingSet

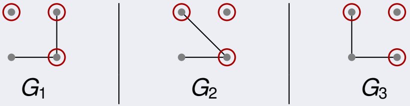


→ *Temporal* variant

→ *Evolving* variant

→ *Permanent* variant

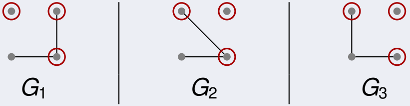
Ex : DominatingSet



- → *Temporal* variant
- → *Evolving* variant
- → *Permanent* variant

Re-definition of problems (3) *(C., Mans, Mathieson, 2011)*

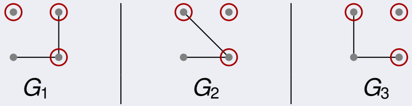
Ex : DominatingSet



- → *Temporal* variant
- → *Evolving* variant
- → *Permanent* variant

Note : $PermanentDS \supsetneq EvolvingDS_i \supsetneq TemporalDS$.

Ex : DominatingSet



- → *Temporal* variant
- → *Evolving* variant
- → *Permanent* variant

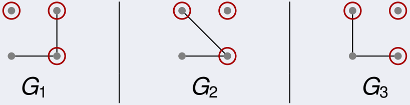
Note : $PermanentDS \supsetneq EvolvingDS_i \supsetneq TemporalDS$.

Works with other problems... but not all problems in general.

Re-definition of problems (3)

(C., Mans, Mathieson, 2011)

Ex : DominatingSet

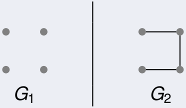


- → *Temporal* variant
- → *Evolving* variant
- → *Permanent* variant

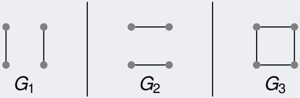
Note : *PermanentDS* $\supsetneq$ *EvolvingDS*$_i$ $\supsetneq$ *TemporalDS*.

Works with other problems... but not all problems in general.

Permanent version for Election or SpanningTree ?



(a) Election ?

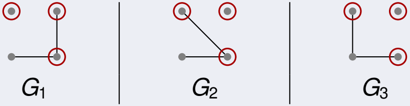


(b) Spanning tree

Re-definition of problems (3)

(C., Mans, Mathieson, 2011)

Ex : DOMINATINGSET

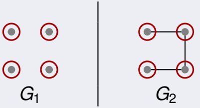


- → *Temporal* variant
- → *Evolving* variant
- → *Permanent* variant

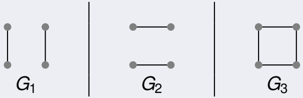
Note : $PermanentDS \supsetneq EvolvingDS_i \supsetneq TemporalDS$.

Works with other problems... but not all problems in general.

Permanent version for ELECTION or SPANNINGTREE ?



(a) Election ?

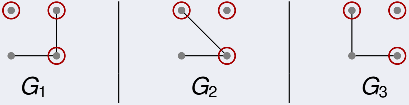


(b) Spanning tree

Re-definition of problems (3)

(C., Mans, Mathieson, 2011)

Ex : DominatingSet

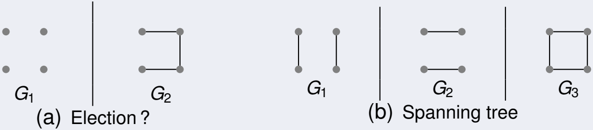


- → *Temporal* variant
- → *Evolving* variant
- → *Permanent* variant

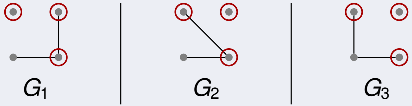
Note : $PermanentDS \supsetneq EvolvingDS_i \supsetneq TemporalDS$.

Works with other problems... but not all problems in general.

Permanent version for Election or SpanningTree ? Makes no sense !



Ex : DominatingSet



→ *Temporal* variant

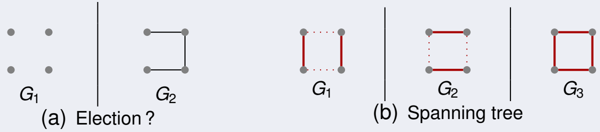
→ *Evolving* variant

→ *Permanent* variant

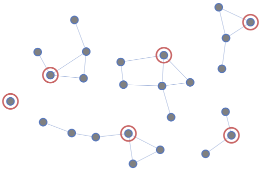
Note : $PermanentDS \supsetneq EvolvingDS_i \supsetneq TemporalDS$.

Works with other problems... but not all problems in general.

Permanent version for Election or SpanningTree ? Makes no sense !



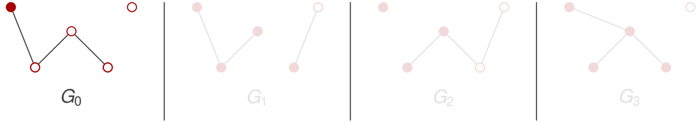
Topological assumptions for distributed algorithms



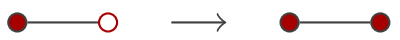
Feasibility, Necessary and sufficient conditions, ...

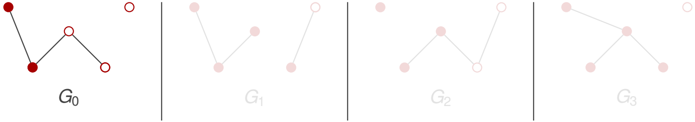
Informal example

Ex : Broadcast algorithm 



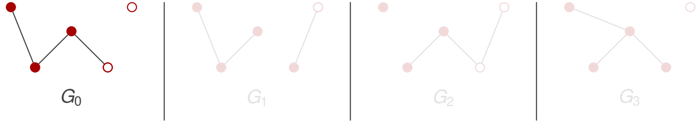
Informal example

Ex : Broadcast algorithm 



Informal example

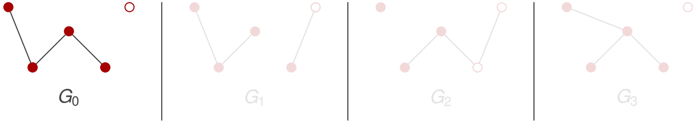
Ex : Broadcast algorithm 



Lucky version.

Informal example

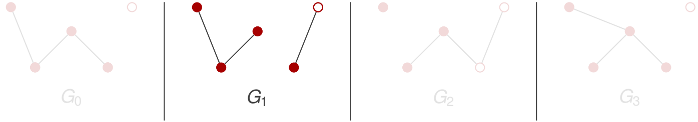
Ex : Broadcast algorithm 



Lucky version.

Informal example

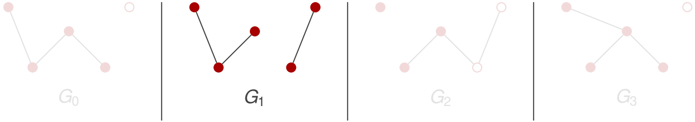
Ex : Broadcast algorithm 



Lucky version.

Informal example

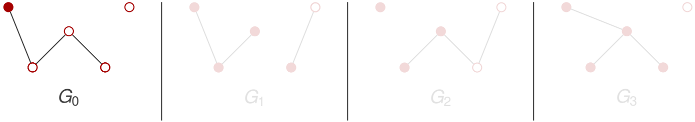
Ex : Broadcast algorithm 



Lucky version. Yeah ! !

Informal example

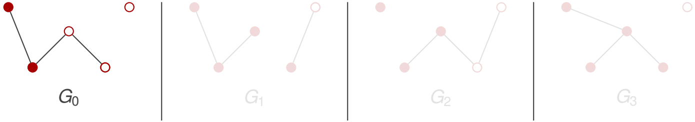
Ex : Broadcast algorithm



But things could have gone differently.

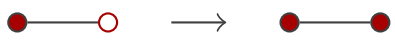
Informal example

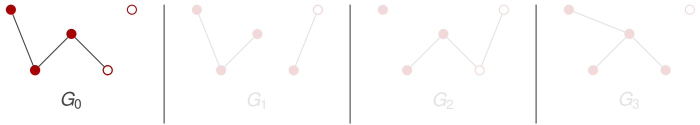
Ex : Broadcast algorithm 



But things could have gone differently.

Informal example

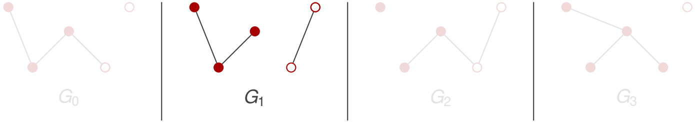
Ex : Broadcast algorithm 



But things could have gone differently.

Informal example

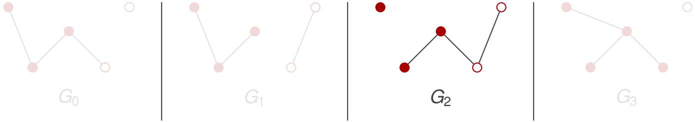
Ex : Broadcast algorithm 



But things could have gone differently. Too late !

Informal example

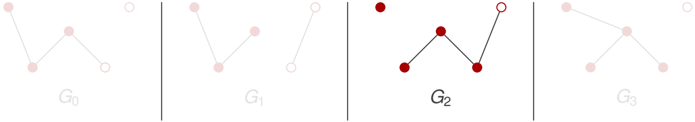
Ex : Broadcast algorithm 



But things could have gone differently. Too late !

Informal example

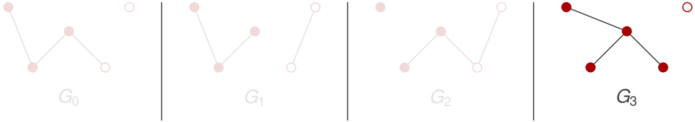
Ex : Broadcast algorithm 



But things could have gone differently. Too late !

Informal example

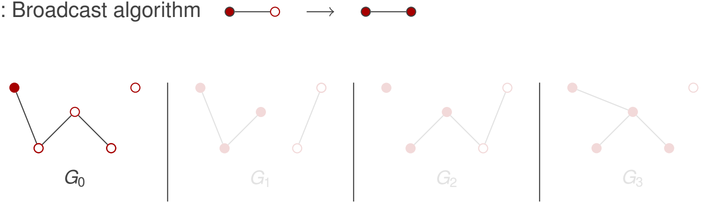
Ex : Broadcast algorithm 



But things could have gone differently. Too late ! Failure !

Informal example

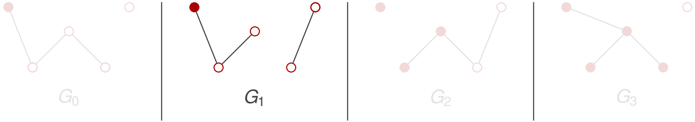
Ex : Broadcast algorithm



Or even worse..

Informal example

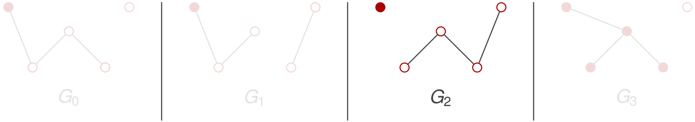
Ex : Broadcast algorithm 



Or even worse.. Too fast !

Informal example

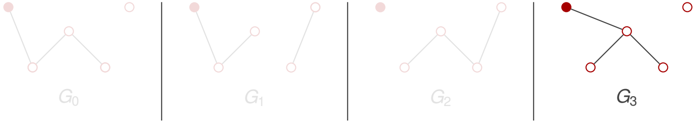
Ex : Broadcast algorithm 



Or even worse.. Too fast ! Too fast !

Informal example

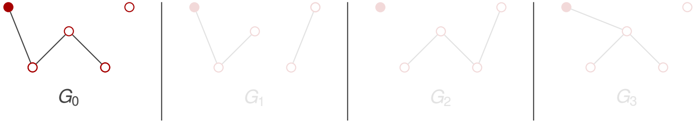
Ex : Broadcast algorithm



Or even worse.. Too fast ! Too fast ! Failure !

Informal example

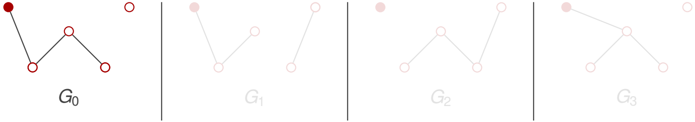
Ex : Broadcast algorithm 



$\Longrightarrow$ Additional assumptions needed to guarantee something.

Informal example

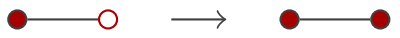
Ex : Broadcast algorithm 

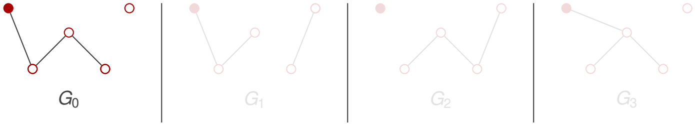


$\Longrightarrow$ Additional assumptions needed to guarantee something.

Assumption : Every present edge is “selected” at least once (but we don’t know in what order...)

Informal example

Ex : Broadcast algorithm 



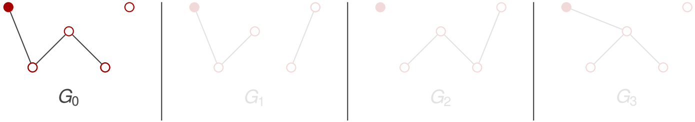
$\Longrightarrow$ Additional assumptions needed to guarantee something.

Assumption : Every present edge is “selected” at least once (but we don’t know in what order...)

$\rightarrow$ Now, is the success guaranteed ?

Informal example

Ex : Broadcast algorithm 



$\Longrightarrow$ Additional assumptions needed to guarantee something.

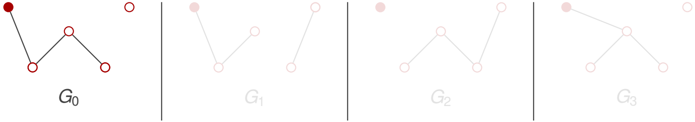
Assumption : Every present edge is “selected” at least once (but we don’t know in what order...)

$\rightarrow$ Now, is the success guaranteed ?

$\rightarrow$ Is the success possible ?

Informal example

Ex : Broadcast algorithm



$\Longrightarrow$ Additional assumptions needed to guarantee something.

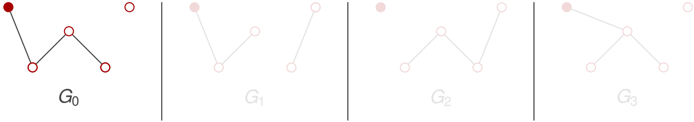
Assumption : Every present edge is “selected” at least once (but we don’t know in what order...)

→ Now, is the success guaranteed ?

→ Is the success possible ? Of course, but why ?

Informal example

Ex : Broadcast algorithm 



$\Longrightarrow$ Additional assumptions needed to guarantee something.

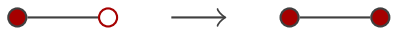
Assumption : Every present edge is “selected” <u>at least once</u> (but we don’t know in what order...)

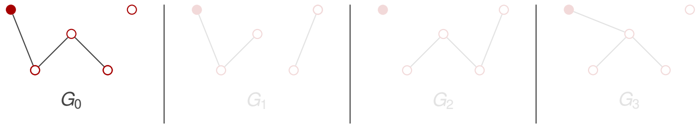
$\rightarrow$ Now, is the success <u>guaranteed</u> ?

$\rightarrow$ Is the success <u>possible</u> ? Of course, but why ? Because ($src \rightsquigarrow *$)

Informal example

Ex : Broadcast algorithm





$\Longrightarrow$ Additional assumptions needed to guarantee something.

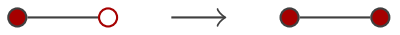
Assumption : Every present edge is "selected" at least once (but we don't know in what order...)

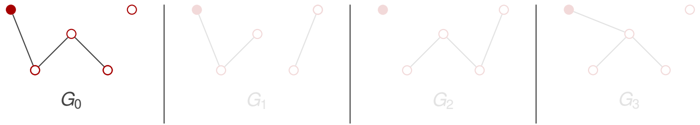
$\rightarrow$ Now, is the success guaranteed ? Why not ?

$\rightarrow$ Is the success possible ? Of course, but why ? Because ($src \rightsquigarrow *$)

Informal example

Ex : Broadcast algorithm





$\Longrightarrow$ Additional assumptions needed to guarantee something.

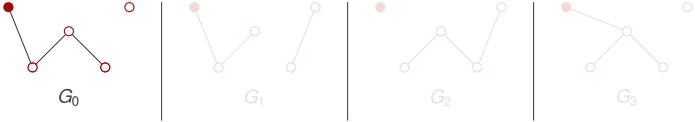
Assumption : Every present edge is "selected" at least once (but we don't know in what order...)

$\rightarrow$ Now, is the success guaranteed ? Why not ? Because $\neg(src \overset{st}{\rightsquigarrow} *)$

$\rightarrow$ Is the success possible ? Of course, but why ? Because $(src \rightsquigarrow *)$

Informal example

Ex : Broadcast algorithm



$\Longrightarrow$ Additional assumptions needed to guarantee something.

Assumption : Every present edge is "selected" at least once (but we don't know in what order...)

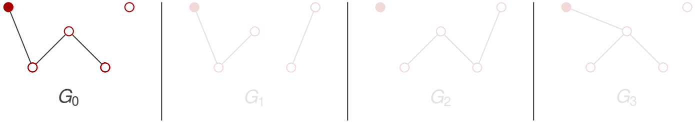
$\rightarrow$ Now, is the success guaranteed ? Why not ? Because $\neg(src \overset{st}{\leadsto} *)$

$\rightarrow$ Is the success possible ? Of course, but why ? Because $(src \leadsto *)$

Notions of *necessary condition* (e.g. $src \leadsto *$) or *sufficient condition* (e.g. $src \overset{st}{\leadsto} *$) for a given algorithm. These conditions relate only to the topology.

Informal example

Ex : Broadcast algorithm 



$\Longrightarrow$ Additional assumptions needed to guarantee something.

Assumption : Every present edge is “selected” at least once (but we don’t know in what order...)

$\rightarrow$ Now, is the success guaranteed ? Why not ? Because $\neg(src \overset{st}{\leadsto} *)$

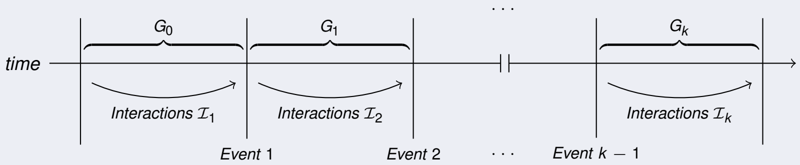
$\rightarrow$ Is the success possible ? Of course, but why ? Because $(src \leadsto *)$

Notions of *necessary condition* (e.g. $src \leadsto *$) or *sufficient condition* (e.g. $src \overset{st}{\leadsto} *$) for a given algorithm. These conditions relate only to the topology.

More formally...

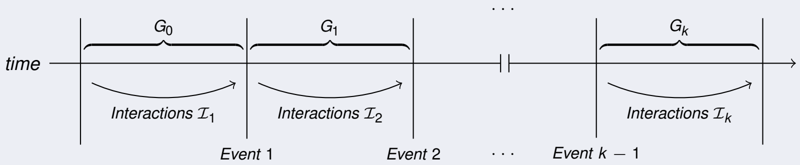
Interaction over dynamic graphs

Interactions over a Dynamic Graph $\mathcal{G} = \{G_0, G_1, ..., G_k\}$



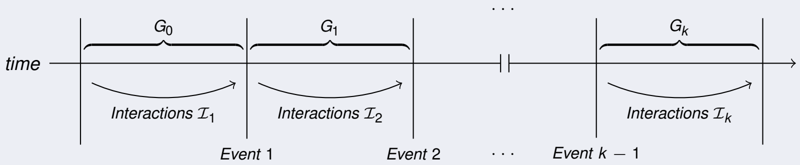
Interaction over dynamic graphs

Interactions over a Dynamic Graph $\mathcal{G} = \{G_0, G_1, ..., G_k\}$



Interaction over dynamic graphs

Interactions over a Dynamic Graph $\mathcal{G} = \{G_0, G_1, ..., G_k\}$

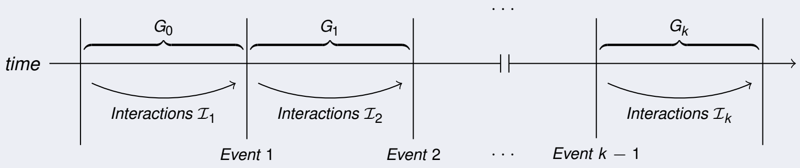


An execution is an alternated sequence of interactions and topological events :

$X = \mathcal{I}_k \circ Event_{k-1} \circ .. \circ Event_2 \circ \mathcal{I}_2 \circ Event_1 \circ \mathcal{I}_1(G_0)$

Interaction over dynamic graphs

Interactions over a Dynamic Graph $\mathcal{G} = \{G_0, G_1, ..., G_k\}$

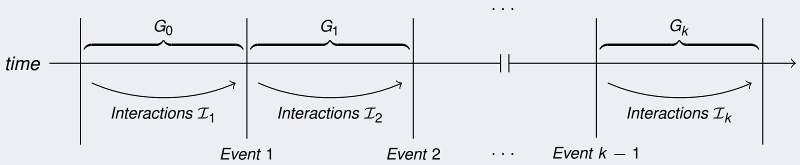


An execution is an alternated sequence of interactions and topological events :

$X = \mathcal{I}_k \circ Event_{k-1} \circ .. \circ Event_2 \circ \mathcal{I}_2 \circ Event_1 \circ \mathcal{I}_1(G_0)$ *Non deterministic !*

Interaction over dynamic graphs

Interactions over a Dynamic Graph $\mathcal{G} = \{G_0, G_1, ..., G_k\}$



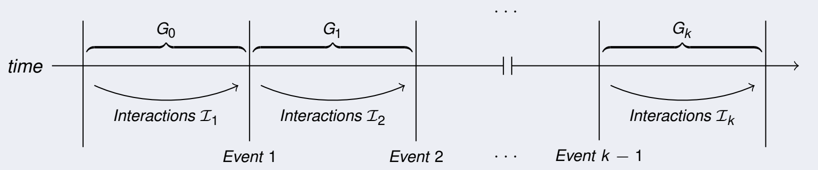
An execution is an alternated sequence of interactions and topological events :

$X = \mathcal{I}_k \circ Event_{k-1} \circ .. \circ Event_2 \circ \mathcal{I}_2 \circ Event_1 \circ \mathcal{I}_1(G_0)$ *Non deterministic !*

$\rightarrow \mathcal{X}$: set of all possible executions (for a given algorithm and graph $\mathcal{G}$).

Interaction over dynamic graphs

Interactions over a Dynamic Graph $\mathcal{G} = \{G_0, G_1, ..., G_k\}$



An execution is an alternated sequence of interactions and topological events :

$X = \mathcal{I}_k \circ Event_{k-1} \circ .. \circ Event_2 \circ \mathcal{I}_2 \circ Event_1 \circ \mathcal{I}_1(G_0)$ *Non deterministic !*

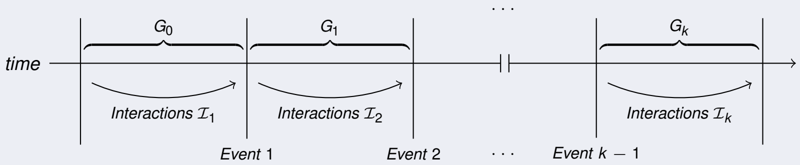
$\rightarrow$ $\mathcal{X}$: set of all possible executions (for a given algorithm and graph $\mathcal{G}$).

What makes a graph property $\mathcal{P}$ a necessary or sufficient condition for success on $\mathcal{G}$?

$\rightarrow$ Necessary condition : $\neg\mathcal{P}(\mathcal{G}) \implies \forall X \in \mathcal{X}$, failure(X).

Interaction over dynamic graphs

Interactions over a Dynamic Graph $\mathcal{G} = \{G_0, G_1, ..., G_k\}$



An execution is an alternated sequence of interactions and topological events :

$X = \mathcal{I}_k \circ Event_{k-1} \circ .. \circ Event_2 \circ \mathcal{I}_2 \circ Event_1 \circ \mathcal{I}_1(G_0)$ *Non deterministic !*

→ $\mathcal{X}$: set of all possible executions (for a given algorithm and graph $\mathcal{G}$).

What makes a graph property $\mathcal{P}$ a necessary or sufficient condition for success on $\mathcal{G}$?

→ Necessary condition : $\neg\mathcal{P}(\mathcal{G}) \implies \forall X \in \mathcal{X}$, failure(X).

→ Sufficient condition : $\mathcal{P}(\mathcal{G}) \implies \forall X \in \mathcal{X}$, success(X).

back to the broadcast example

back to the broadcast example

Necessary condition

→ $\mathcal{P}_{\mathcal{N}}$: there exists a journey from the source to all other nodes (noted $src \rightsquigarrow *$).

back to the broadcast example

Necessary condition

$\rightarrow \mathcal{P}_{\mathcal{N}}$: there exists a journey from the source to all other nodes (noted *src* $\rightsquigarrow *$).

Sufficient condition

$\rightarrow \mathcal{P}_{\mathcal{S}}$: there exists a strict journey from the source to all other nodes (noted *src* $\overset{st}{\rightsquigarrow} *$).

back to the broadcast example

Necessary condition

→ $\mathcal{P}_{\mathcal{N}}$: there exists a journey from the source to all other nodes (noted $src \rightsquigarrow *$).

Sufficient condition

→ $\mathcal{P}_{\mathcal{S}}$: there exists a strict journey from the source to all other nodes (noted $src \overset{st}{\rightsquigarrow} *$).

Classes of dynamic graphs

→ $\mathcal{C}_1$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by at least one node (noted $1 \rightsquigarrow *$).

back to the broadcast example

Necessary condition

$\rightarrow \mathcal{P}_{\mathcal{N}}$: there exists a journey from the source to all other nodes (noted $src \rightsquigarrow *$).

Sufficient condition

$\rightarrow \mathcal{P}_{\mathcal{S}}$: there exists a strict journey from the source to all other nodes (noted $src \overset{st}{\rightsquigarrow} *$).

Classes of dynamic graphs

$\rightarrow \mathcal{C}_1$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by at least one node (noted $1 \rightsquigarrow *$).

$\rightarrow \mathcal{C}_2$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by all nodes ($* \rightsquigarrow *$).

back to the broadcast example

Necessary condition

$\rightarrow \mathcal{P}_{\mathcal{N}}$: there exists a journey from the source to all other nodes (noted $src \rightsquigarrow *$).

Sufficient condition

$\rightarrow \mathcal{P}_{\mathcal{S}}$: there exists a strict journey from the source to all other nodes (noted $src \overset{st}{\rightsquigarrow} *$).

Classes of dynamic graphs

- $\rightarrow \mathcal{C}_1$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by at least one node (noted $1 \rightsquigarrow *$).
- $\rightarrow \mathcal{C}_2$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by all nodes ($* \rightsquigarrow *$).
- $\rightarrow \mathcal{C}_3$: $\mathcal{P}_{\mathcal{S}}$ is satisfied by at least one node ($1 \overset{st}{\rightsquigarrow} *$).

back to the broadcast example

Necessary condition

$\rightarrow \mathcal{P}_{\mathcal{N}}$: there exists a journey from the source to all other nodes (noted $src \rightsquigarrow *$).

Sufficient condition

$\rightarrow \mathcal{P}_{\mathcal{S}}$: there exists a strict journey from the source to all other nodes (noted $src \overset{st}{\rightsquigarrow} *$).

Classes of dynamic graphs

- $\rightarrow \mathcal{C}_1$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by at least one node (noted $1 \rightsquigarrow *$).
- $\rightarrow \mathcal{C}_2$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by all nodes ($* \rightsquigarrow *$).
- $\rightarrow \mathcal{C}_3$: $\mathcal{P}_{\mathcal{S}}$ is satisfied by at least one node ($1 \overset{st}{\rightsquigarrow} *$).
- $\rightarrow \mathcal{C}_4$: $\mathcal{P}_{\mathcal{S}}$ is satistied by all nodes ($* \overset{st}{\rightsquigarrow} *$).

back to the broadcast example

Necessary condition

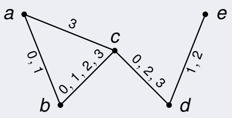
$\rightarrow \mathcal{P}_{\mathcal{N}}$: there exists a journey from the source to all other nodes (noted *src* $\rightsquigarrow *$).

Sufficient condition

$\rightarrow \mathcal{P}_{\mathcal{S}}$: there exists a strict journey from the source to all other nodes (noted *src* $\overset{st}{\rightsquigarrow} *$).

Classes of dynamic graphs

- $\rightarrow \mathcal{C}_1$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by at least one node (noted $1 \rightsquigarrow *$).
- $\rightarrow \mathcal{C}_2$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by all nodes ($* \rightsquigarrow *$).
- $\rightarrow \mathcal{C}_3$: $\mathcal{P}_{\mathcal{S}}$ is satisfied by at least one node ($1 \overset{st}{\rightsquigarrow} *$).
- $\rightarrow \mathcal{C}_4$: $\mathcal{P}_{\mathcal{S}}$ is satistied by all nodes ($* \overset{st}{\rightsquigarrow} *$).



back to the broadcast example

Necessary condition

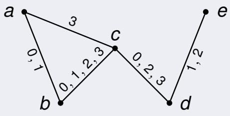
$\rightarrow \mathcal{P}_{\mathcal{N}}$: there exists a journey from the source to all other nodes (noted $src \rightsquigarrow *$).

Sufficient condition

$\rightarrow \mathcal{P}_{\mathcal{S}}$: there exists a <u>strict</u> journey from the source to all other nodes (noted $src \overset{st}{\rightsquigarrow} *$).

Classes of dynamic graphs

- $\rightarrow \mathcal{C}_1$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by at least one node (noted $1 \rightsquigarrow *$).
- $\rightarrow \mathcal{C}_2$: $\mathcal{P}_{\mathcal{N}}$ is satisfied by all nodes ($* \rightsquigarrow *$).
- $\rightarrow \mathcal{C}_3$: $\mathcal{P}_{\mathcal{S}}$ is satisfied by at least one node ($1 \overset{st}{\rightsquigarrow} *$).
- $\rightarrow \mathcal{C}_4$: $\mathcal{P}_{\mathcal{S}}$ is satistied by all nodes ($* \overset{st}{\rightsquigarrow} *$).



$\in \mathcal{C}_2 \cap \mathcal{C}_3$

Counting algorithm (non-uniform)

Counting with a distinguished counter

- Initial states : 1 for the counter, N for all other nodes.
- Algorithm : 

→ Hopefully, after some time, the counter is labelled n.

Counting algorithm (non-uniform)

Counting with a distinguished counter

- Initial states : 1 for the counter, N for all other nodes.
- Algorithm : 

$\rightarrow$ Hopefully, after some time, the counter is labelled n. But when ?

Counting algorithm (non-uniform)

Counting with a distinguished counter

- Initial states : 1 for the counter, N for all other nodes.
- Algorithm : 

$\rightarrow$ Hopefully, after some time, the counter is labelled n. But when ?

Necessary or sufficient conditions

- $\mathcal{P}_{\mathcal{N}}$: there exists an edge, at some time, between the counter and every other node.

Counting algorithm (non-uniform)

Counting with a distinguished counter

- Initial states : 1 for the counter, N for all other nodes.
- Algorithm : 

$\rightarrow$ Hopefully, after some time, the counter is labelled n. But when ?

Necessary or sufficient conditions

- $\mathcal{P}_{\mathcal{N}}$: there exists an edge, at some time, between the counter and every other node.
- $\mathcal{P}_{\mathcal{S}} = \mathcal{P}_{\mathcal{N}}$.

Counting algorithm (non-uniform)

Counting with a distinguished counter

- Initial states : 1 for the counter, N for all other nodes.
- Algorithm : 

$\rightarrow$ Hopefully, after some time, the counter is labelled n. But when ?

Necessary or sufficient conditions

- $\mathcal{P}_{\mathcal{N}}$: there exists an edge, at some time, between the counter and every other node.
- $\mathcal{P}_{\mathcal{S}} = \mathcal{P}_{\mathcal{N}}$.

Classes of dynamic graphs

$\rightarrow$ $\mathcal{C}_5$: at least one node verifies $\mathcal{P}$, (noted 1–$*$).

Counting algorithm (non-uniform)

Counting with a distinguished counter

- Initial states : 1 for the counter, N for all other nodes.
- Algorithm : 

$\rightarrow$ Hopefully, after some time, the counter is labelled n. But when ?

Necessary or sufficient conditions

- $\mathcal{P}_{\mathcal{N}}$: there exists an edge, at some time, between the counter and every other node.
- $\mathcal{P}_{\mathcal{S}} = \mathcal{P}_{\mathcal{N}}$.

Classes of dynamic graphs

$\rightarrow$ $\mathcal{C}_5$: at least one node verifies $\mathcal{P}$, (noted 1–$*$).

$\rightarrow$ $\mathcal{C}_6$: all the nodes verify $\mathcal{P}$, (noted $*$–$*$).

Counting algorithm (uniform)

Uniform counting (every body is initially a counter)

- Initial states : 1 (all nodes).

Counting algorithm (uniform)

Uniform counting (every body is initially a counter)

- Initial states : 1 (all nodes).
- Algorithm :

$\rightarrow$ Hopefully, after some time, one node is labelled *n*.

Counting algorithm (uniform)

Uniform counting (every body is initially a counter)

- Initial states : 1 (all nodes).
- Algorithm : 

$\rightarrow$ Hopefully, after some time, one node is labelled n. But when ?

Counting algorithm (uniform)

Uniform counting (every body is initially a counter)

- Initial states : 1 (all nodes).
- Algorithm : 

$\rightarrow$ Hopefully, after some time, one node is labelled n. But when ?

Conditions and classes of graphs

- Necessary condition $\mathcal{C}_{\mathcal{N}}$: at least one node can be reached by all ($* \rightsquigarrow 1$).

Counting algorithm (uniform)

Uniform counting (every body is initially a counter)

- Initial states : 1 (all nodes).
- Algorithm :

$\rightarrow$ Hopefully, after some time, one node is labelled n. But when ?

Conditions and classes of graphs

- Necessary condition $\mathcal{C}_{\mathcal{N}}$: at least one node can be reached by all ($* \rightsquigarrow 1$).
- Sufficient condition $\mathcal{C}_{\mathcal{S}}$: all pairs of nodes must share an edge at least once over time ($*$–$*$). *(Marchand de Kerchove, Guinand, 2012)*

Counting algorithm (uniform)

Uniform counting (every body is initially a counter)

- Initial states : 1 (all nodes).
- Algorithm : 

$\rightarrow$ Hopefully, after some time, one node is labelled n. But when ?

Conditions and classes of graphs

- Necessary condition $\mathcal{C}_{\mathcal{N}}$: at least one node can be reached by all ($* \rightsquigarrow 1$).
  - $\rightarrow$ $\mathcal{C}_7$: graphs having this property.
- Sufficient condition $\mathcal{C}_{\mathcal{S}}$: all pairs of nodes must share an edge at least once over time ($*$–$*$). *(Marchand de Kerchove, Guinand, 2012)*
  - $\rightarrow$ $\mathcal{C}_6$ (already seen before).

Tightness of a condition ? *(Marchand de Kerchove, Guinand, 2012)*

Necessary condition

Sufficient condition

Tightness of a condition ? *(Marchand de Kerchove, Guinand, 2012)*

Necessary condition

- Not satisfied $\implies$ failure is guaranteed $(\nexists X, \mathit{success}(X))$

Sufficient condition

- Satisfied $\implies$ success is guaranteed $(\nexists X, \mathit{failure}(X))$

Tightness of a condition ? *(Marchand de Kerchove, Guinand, 2012)*

Tight Necessary condition

- Not satisfied $\implies$ failure is guaranteed $(\nexists X, success(X))$
- *Satisfied* $\implies$ *success is possible* $(\exists X, success(X))$.

Sufficient condition

- Satisfied $\implies$ success is guaranteed $(\nexists X, failure(X))$

Tightness of a condition ? *(Marchand de Kerchove, Guinand, 2012)*

Tight Necessary condition

- Not satisfied $\implies$ failure is guaranteed $(\nexists X, success(X))$
- *Satisfied* $\implies$ *success is possible* $(\exists X, success(X))$.

Tight Sufficient condition

- Satisfied $\implies$ success is guaranteed $(\nexists X, failure(X))$
- *Not satisfied* $\implies$ *failure is possible* $(\exists X, failure(X))$

Tightness of a condition ? *(Marchand de Kerchove, Guinand, 2012)*

Tight Necessary condition

- Not satisfied $\Longrightarrow$ failure is guaranteed $(\nexists X, success(X))$
- *Satisfied* $\Longrightarrow$ *success is possible* $(\exists X, success(X))$.

Tight Sufficient condition

- Satisfied $\Longrightarrow$ success is guaranteed $(\nexists X, failure(X))$
- *Not satisfied* $\Longrightarrow$ *failure is possible* $(\exists X, failure(X))$

Remark : Topological conditions which are both necessary and sufficient may not exist !

Tightness of a condition ? *(Marchand de Kerchove, Guinand, 2012)*

Tight Necessary condition

- Not satisfied $\Longrightarrow$ failure is guaranteed $(\nexists X, success(X))$
- *Satisfied* $\Longrightarrow$ *success is possible* $(\exists X, success(X))$.

Tight Sufficient condition

- Satisfied $\Longrightarrow$ success is guaranteed $(\nexists X, failure(X))$
- *Not satisfied* $\Longrightarrow$ *failure is possible* $(\exists X, failure(X))$

Remark : Topological conditions which are both necessary and sufficient may not exist !

Ex. uniform counting (last algorithm) :

$\rightarrow$ $(* \rightsquigarrow 1)$ is a tight necessary condition

$\rightarrow$ $(* - *)$ is a tight sufficient condition

In between : outcome is uncertain... might succeed or fail.

Classifying dynamic networks

$1 - *$ $\mathcal{C}_5$

$1 \overset{st}{\rightsquigarrow} *$ $\mathcal{C}_3$

$1 \rightsquigarrow *$ $\mathcal{C}_1$

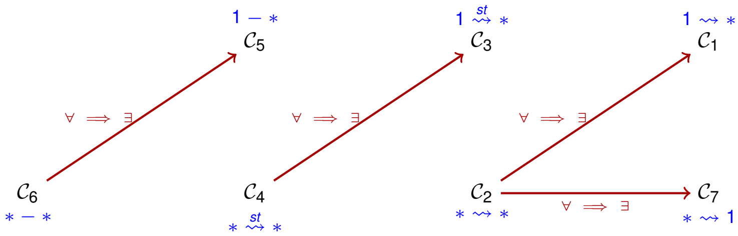
$\mathcal{C}_6$ $* - *$

$\mathcal{C}_4$ $* \overset{st}{\rightsquigarrow} *$

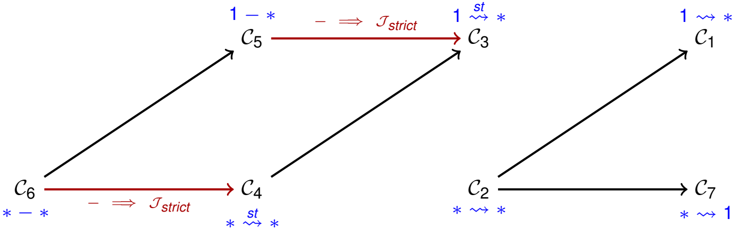
$\mathcal{C}_2$ $* \rightsquigarrow *$

$\mathcal{C}_7$ $* \rightsquigarrow 1$

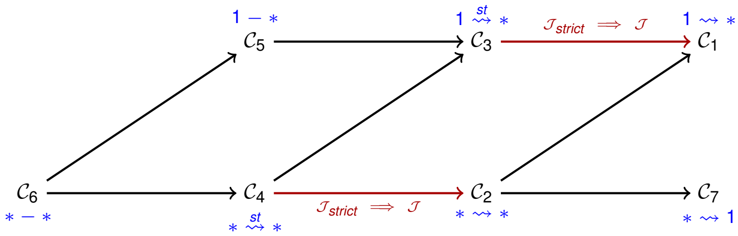
Classifying dynamic networks



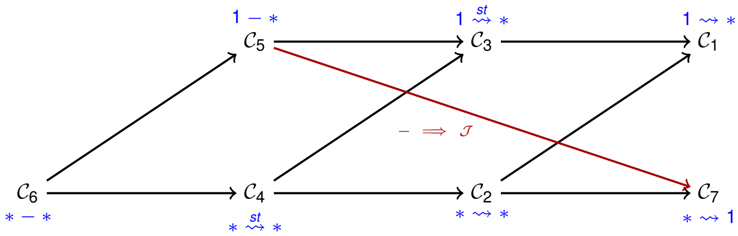
Classifying dynamic networks



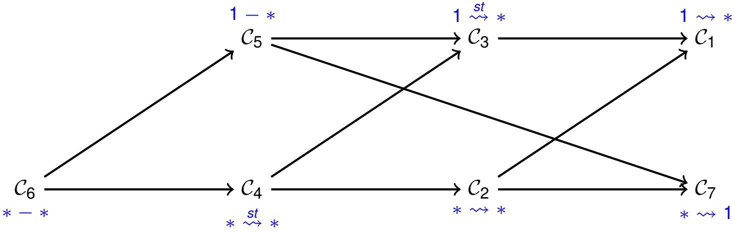
Classifying dynamic networks



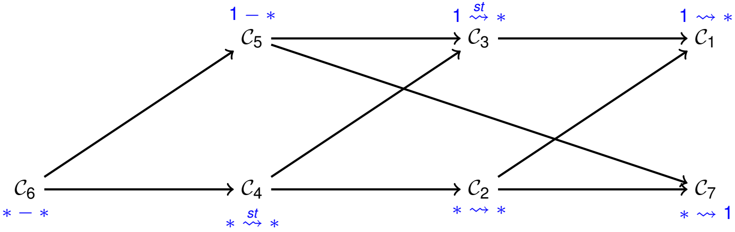
Classifying dynamic networks



Classifying dynamic networks

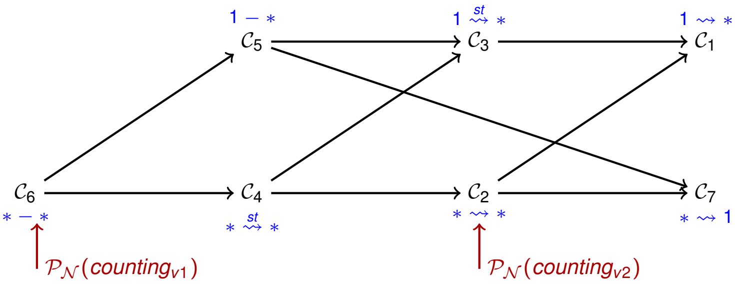


Classifying dynamic networks



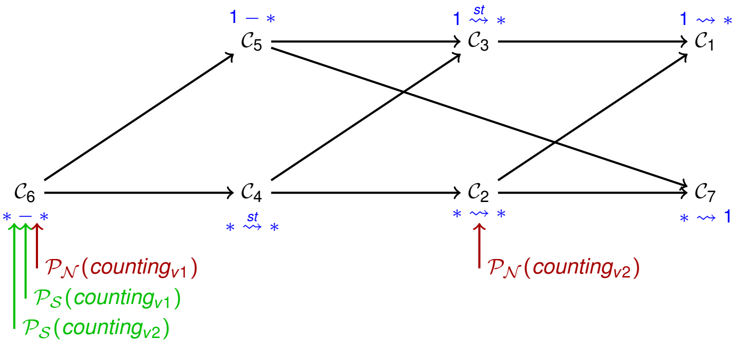
→ Comparison of algorithms on a formal basis

Classifying dynamic networks

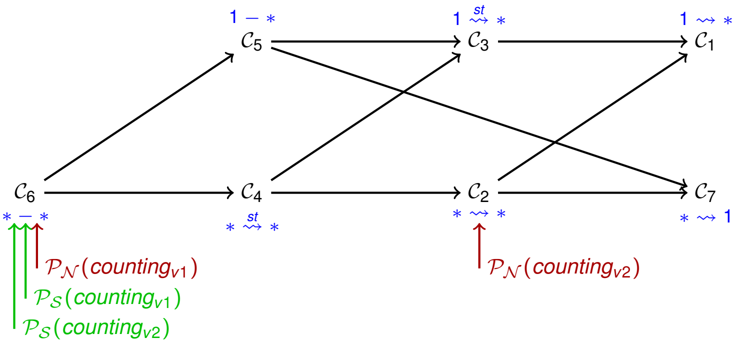


→ Comparison of algorithms on a formal basis

Classifying dynamic networks



→ Comparison of algorithms on a formal basis

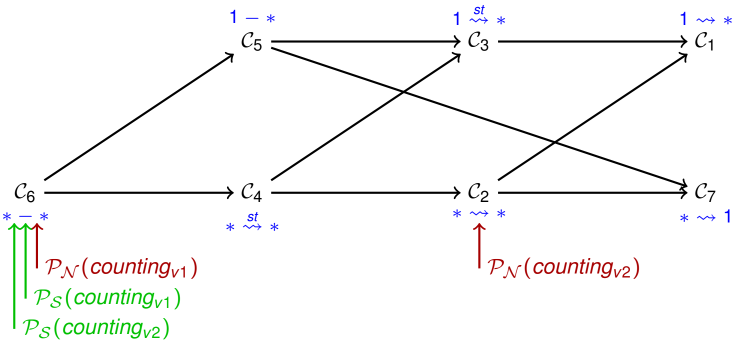


→ Comparison of algorithms on a formal basis

→ Decision making (what algorithm to use ?)

→ e.g. using automated property checking on network traces).

Classifying dynamic networks



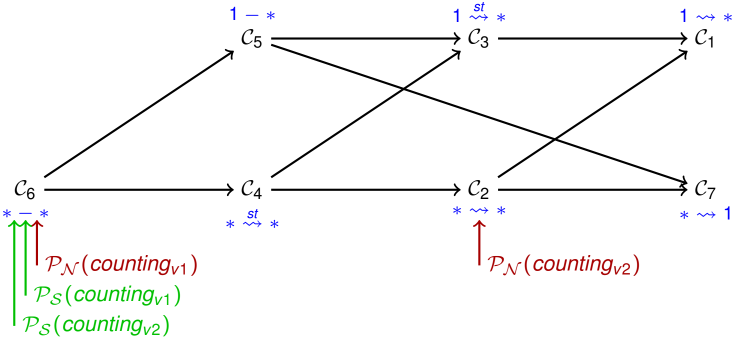
→ Comparison of algorithms on a formal basis

→ Decision making (what algorithm to use ?)

→ e.g. using automated property checking on network traces).

→ Formal proofs ? (Coq)

Classifying dynamic networks



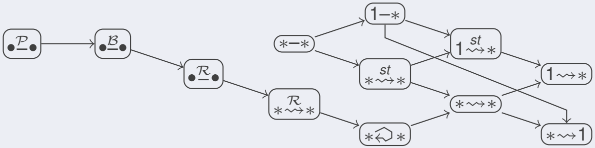
- → Comparison of algorithms on a formal basis
- → Decision making (what algorithm to use ?)
  - → e.g. using automated property checking on network traces).
- → Formal proofs ? (Coq)

Q : How far beyond toy examples ?

Classifying further

(C., Flocchini, Quattrociocchi, Santoro, 2012)

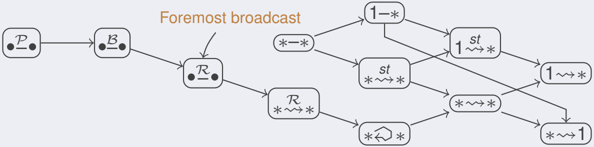
Extending the hierarchy



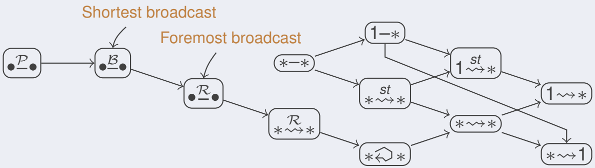
Classifying further

(C., Flocchini, Quattrociocchi, Santoro, 2012)

Extending the hierarchy

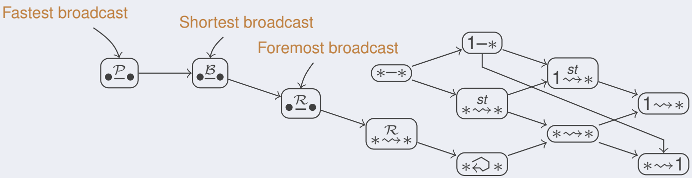


Extending the hierarchy



(C., Flocchini, Quattrociocchi, Santoro, 2012)

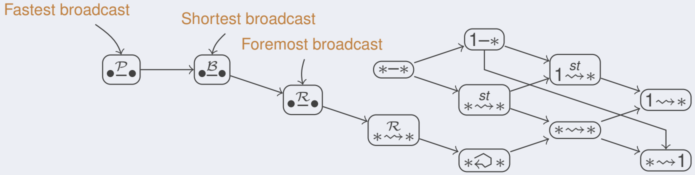
Extending the hierarchy



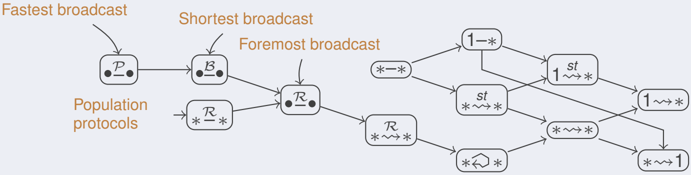
Classifying further

(C., Flocchini, Quattrociocchi, Santoro, 2012)

Extending the hierarchy



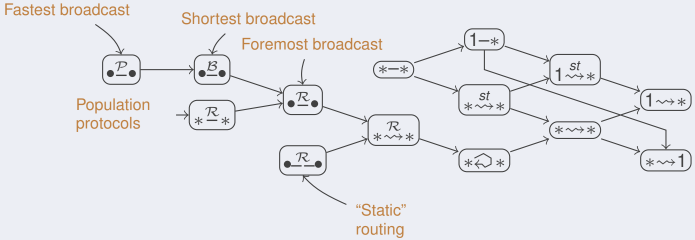
Extending the hierarchy



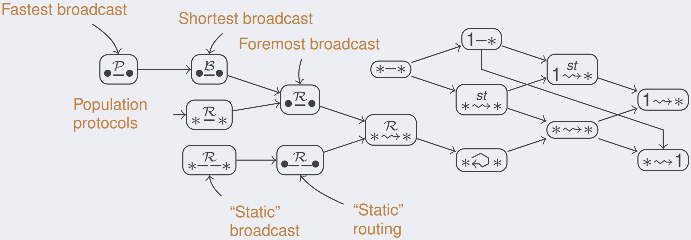
Classifying further

(C., Flocchini, Quattrociocchi, Santoro, 2012)

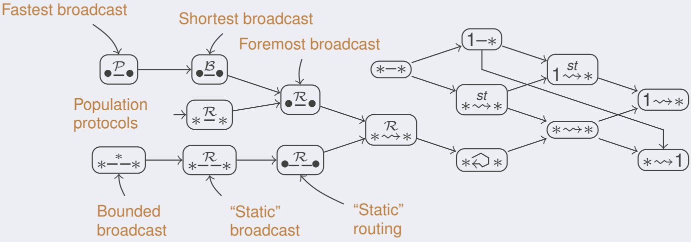
Extending the hierarchy



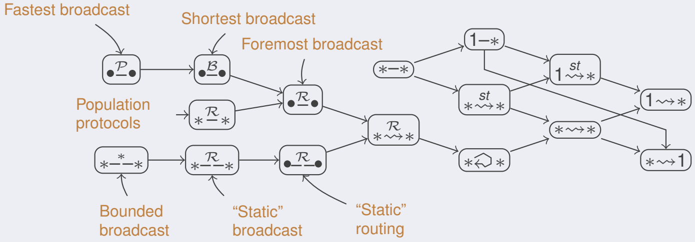
Extending the hierarchy



Extending the hierarchy



Extending the hierarchy



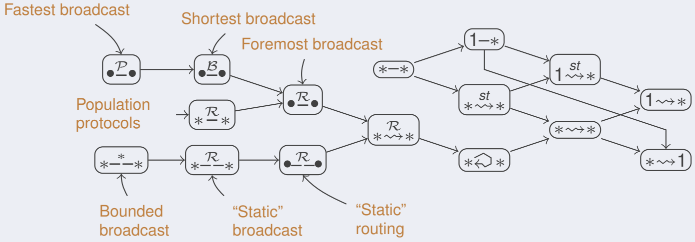
Ex : Bounded broadcast in $(* \overset{*}{-} *)$ *(O'Dell and Wattenhofer, 2005)*

The graph is arbitrarily dynamic, as long as every G_i remains connected :

Classifying further

(C., Flocchini, Quattrociocchi, Santoro, 2012)

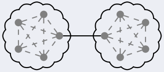
Extending the hierarchy



Ex : Bounded broadcast in $(*\overset{*}{-}*)$ *(O'Dell and Wattenhofer, 2005)*

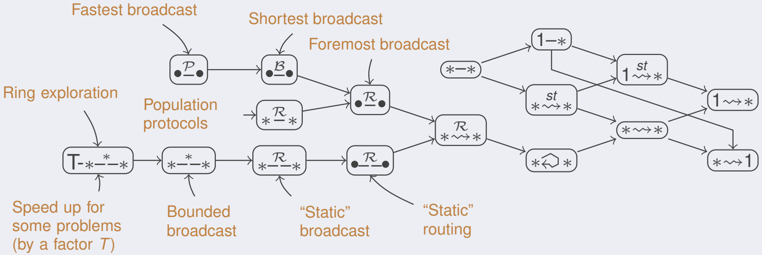
The graph is arbitrarily dynamic, as long as every G_i remains connected :

Min cut of size 1 between informed and uninformed nodes :
$\longrightarrow$ At least one new node informed in each step.

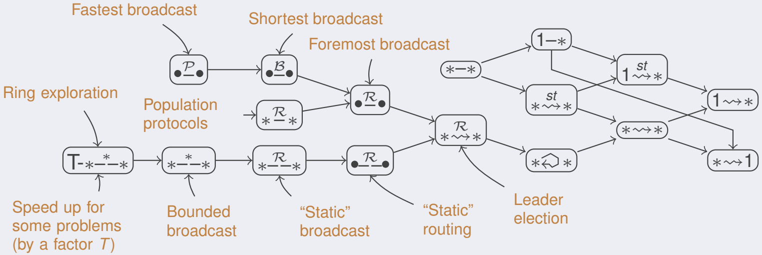


 (C., Flocchini, Quattrociocchi, Santoro, 2012)

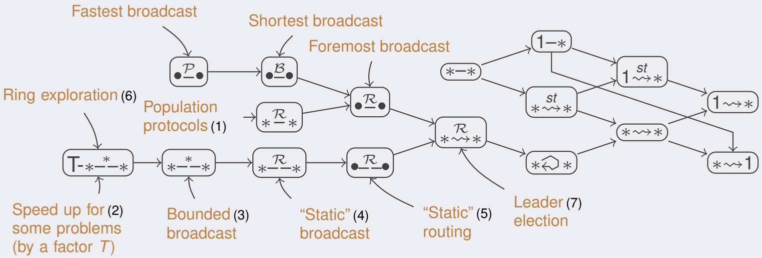
Extending the hierarchy



Extending the hierarchy



Extending the hierarchy



(1) Complete graph of interaction *(Angluin, Aspnes, Diamadi, Fischer, Peralta, 2004)*

(2) T-interval connectivity *(Kuhn, Lynch, Oshman, 2010)*

(3) Constant connectivity *(O'Dell and Wattenhofer, 2005)*

(4) Eventual instant connectivity *(Ramanathan, Basu, and Krishnan, 2007)*

(5) Eventual instant routability *(Ramanathan, Basu, and Krishnan, 2007)*

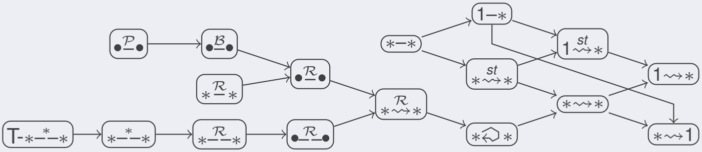
(6) T-interval connectivity *(Ilcinkas, Wade, 2013)*

(7) Recurrent temporal connectivity *(Arantes, Greve, Sens, Simon, 2013)* *(Gòmez-Cazaldo, Lafuente, Larrea, Raynal, 2013)*

Classifying further

(C., Flocchini, Quattrociocchi, Santoro, 2012)

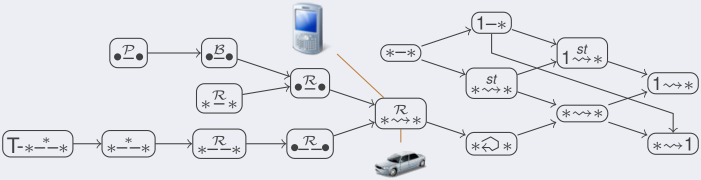
Real mobility contexts



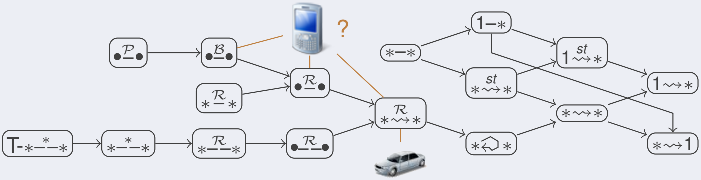
Classifying further

(C., Flocchini, Quattrociocchi, Santoro, 2012)

Real mobility contexts

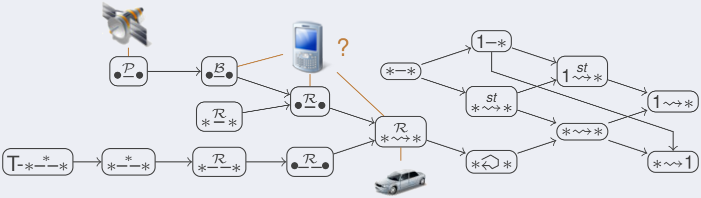


Real mobility contexts



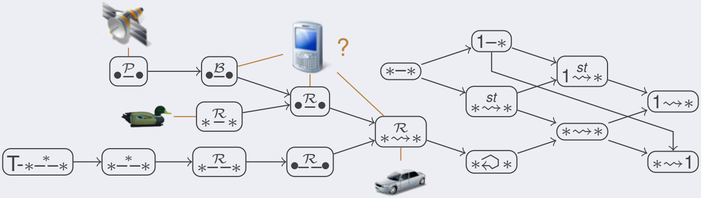
Classifying further *(C., Flocchini, Quattrociocchi, Santoro, 2012)*

Real mobility contexts



 (C., Flocchini, Quattrociocchi, Santoro, 2012)

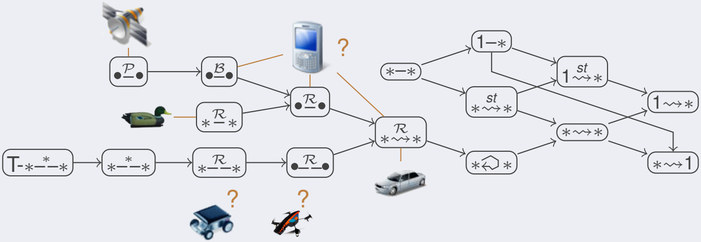
Real mobility contexts



Classifying further

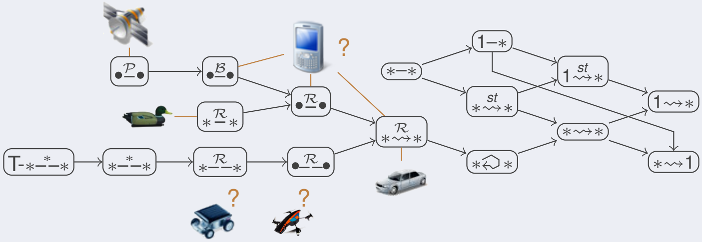
(C., Flocchini, Quattrociocchi, Santoro, 2012)

Real mobility contexts



(C., Flocchini, Quattrociocchi, Santoro, 2012)

Real mobility contexts



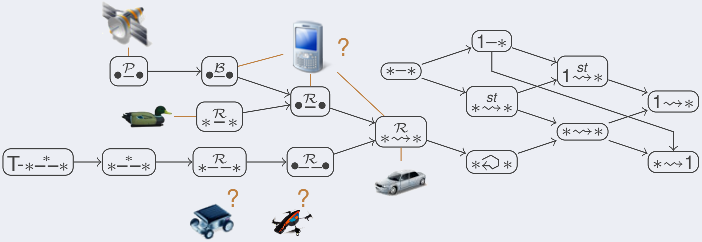
How to proceed ?

→ Generate connection traces

→ Test properties

Classifying further *(C., Flocchini, Quattrociocchi, Santoro, 2012)*

Real mobility contexts



How to proceed ?

→ Generate connection traces

→ Test properties

- Temporal Connectivity *(Whitbeck et al. 2012 ; Barjon et al., 2014)*
- T-Interval Connectivity *(C. et al., 2014)*

References (external) :

- I. Litovsky, Y. Métivier, and E. Sopena, Graph relabelling systems and distributed algorithms., *Handbook of graph grammars and computing by graph transformation, 1999.*
- *F. Marchand de Kerchove and F. Guinand, Strengthening Topological Conditions for Relabeling Algorithms in Evolving Graphs, Technical report, Université Le Havre, 2012.*
- *B. Bui-Xuan, A. Ferreira, and A. Jarry, Computing shortest, fastest, and foremost journeys in dynamic networks, JFCS 14(2) : 267-285, 2003.*
- *Y. Métivier, N. Saheb, A. Zemmari, Analysis of a randomized rendezvous algorithm, Information and Computation 184(1) :109-128, 2003.*
- *A. Ferreira, Building a Reference Combinatorial Model for manets, IEEE Network 18(5) : 24-29, 2004.*
- *R. O'Dell and R. Wattenhofer, Information dissemination in highly dynamic graphs, DIALM-POMC, 2005.*
- *F. Kuhn, N. Lynch, R. Oshman, Distributed computation in dynamic networks, STOC, 2010.*
- *R. Ramanathan, P. Basu, and R. Krishnan, Towards a Formalism for Routing in Challenged Networks, CHANTS, 2007.*
- *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.*
- *D. Ilcinkas and A. Wade, Exploration of the T-Interval-Connected Dynamic Graphs : the Case of the Ring, SIROCCO, 2013.*
- *J. Whitbeck, M. Dias De Amorim, V. Conan, J.-L. Guillaume, Temporal Reachability Graphs, PhD Thesis, University of Bordeaux, 2006.*
- *L. Arantes, F. Greve, P. Sens, V. Simon, Eventual Leader Election in Evolving Mobile Networks, OPODIS 2013*
- *C. Gomez, A. Lafuente, M. Larrea, M. Raynal, Fault-Tolerant Leader Election in Mobile Dynamic Distributed Systems, PRDC 2013.*

References (self) :

- A. Casteigts, P. Flocchini, W. Quattrociocchi, N. Santoro, Time-Varying Graphs and Dynamic Networks., *IJPEDS 27(5) : 387-408, 2012.*
- *A. Casteigts, S. Chaumette, A. Ferreira, Characterizing Topological Assumptions of Distributed Algorithms in Dynamic Networks, SIROCCO, 2009.*
- *A. Casteigts, P. Flocchini, B. Mans, N. Santoro, Measuring Temporal Lags in Delay-Tolerant Networks, IEEE Transactions on Computer, 63(2) :397-410, 2014.*
- *A. Casteigts, B. Mans, L. Mathieson, On the Feasibility of Maintenance Algorithms in Dynamic Graphs, CoRR abs/1107.2722, 2011.*
- *A. Casteigts, P. Flocchini, B. Mans, N. Santoro, Shortest, Fastest, and Foremost Broadcast in Dynamic Networks, IFIP-TCS 2010*
- *M. Barjon, A. Casteigts, S. Chaumette, C. Johnen, Y.M. Neggaz, Testing Temporal Connectivity in Sparse Dynamic Graphs, ALGOTEL 2014*
- *A. Casteigts, R. Klasing, Y.M. Neggaz, J. Peters, Efficiently Testing T-Interval Connectivity in Dynamic Graphs, Ref to be added, 2014.*