INFORMATIQUE
Sciences
Université Paris Cité

Spanning Tree Construction

MPRI

Lélia Blin

lelia.blin@irif.fr

2023

Spanning Tree Problem Specification

Let $G(V, E)$ a connected undirected graph. An acyclic subgraph that connects all the nodes of G is called a spanning tree of G, denoted by $ST(G)$.

Pointer notion

In a spanning tree, the acyclic property ensures that there are no loops or cycles, while the requirement for each node to have a unique pointer to a neighbor guarantees a distinct and unambiguous path between nodes

Self-stabilizing Spanning Tree construction

Main difficulties

- Breaking cycle
- Empty Node pointer

Spanning Tree under constraint

- BFS Spanning Tree
- DFS Spanning Tree
- Minimum spanning Tree
- Minimum Degree Spanning Tree
- Minimum Diameter Spanning Tree

A self-stabilizing algorithm for constructing a spanning trees

1991 Chen Yu Huang IPL

Model

- Semi-uniform → root node denoted by r
- Anonymous
- Knowledge: n
- Scheduler : central (Only one node activate at each step)

Local variables

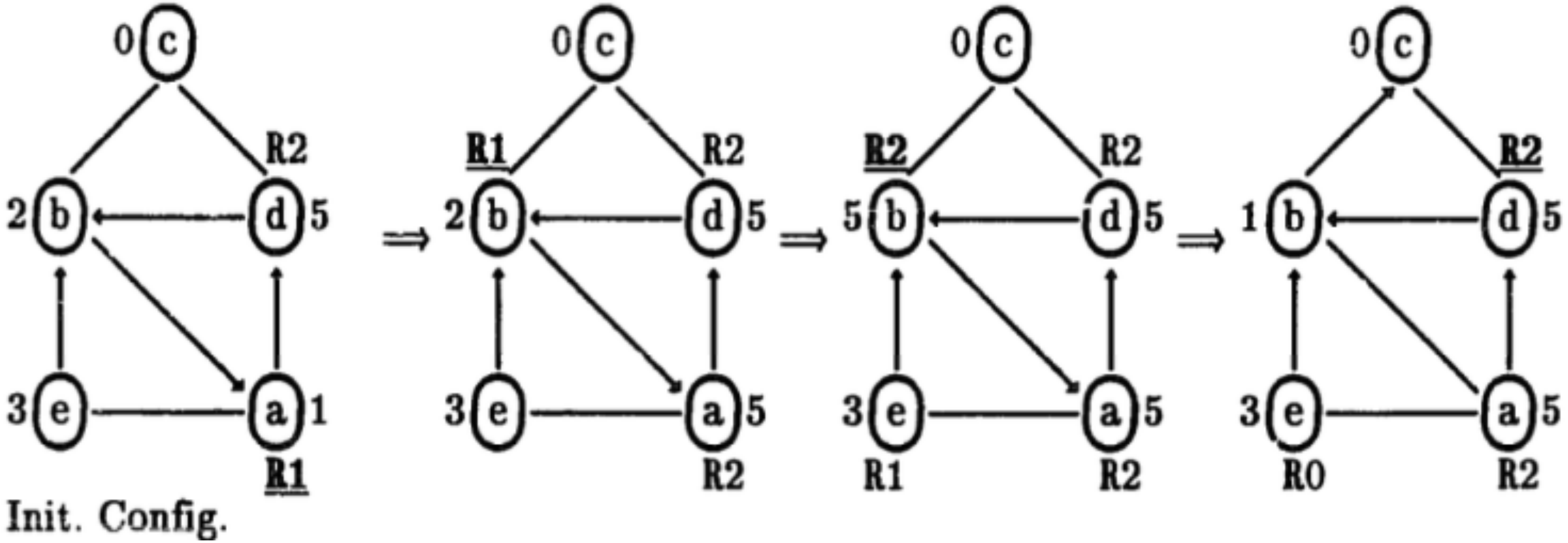
> - Level : $L_v \in \{1, dots, n\}$
> - Parent : $p_v \in \{0, \ldots, n\}$

- Remark: r has no parent and is level is zero
- These variables uses $O(\log)$ bits of memory by node

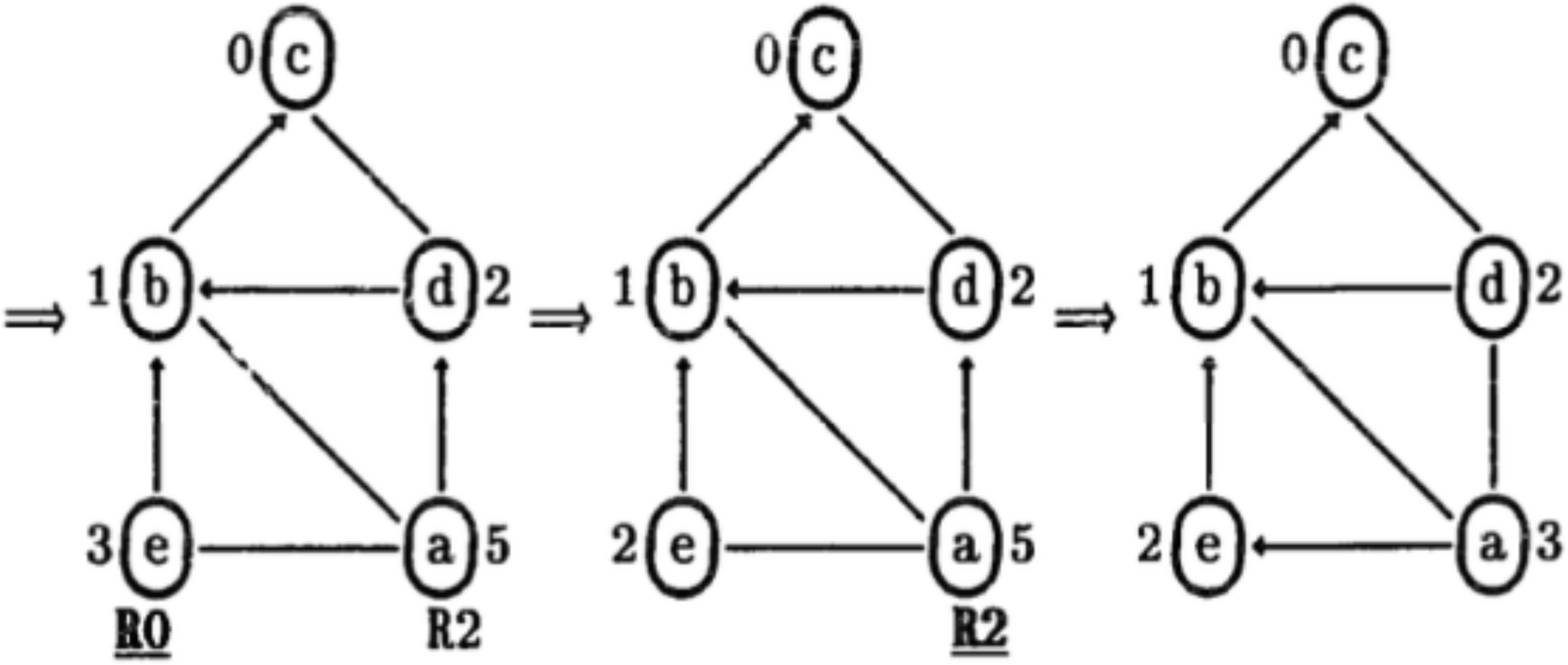
Algorithm

$$R_0 : L_v \neq n \wedge L_v \neq L_{p_v} + 1 \wedge L_{p_v} \neq n \longrightarrow L_v = L_{p_v} + 1$$
$$R_1 : L_v \neq n \wedge L_{p_v} = n \longrightarrow L_v = n$$
$$R_2 : L_v = n \wedge \exists u \in N(v) | L_u < n - 1 \longrightarrow L_v = L_u + 1; p_v = u$$

Example


0 c
R2
2 b
d 5
3 e
a 1
R1
Init. Config.
0 c
R1
R2
2 b
d 5
3 e
a 5
R2
0 c
R2
R2
5 b
d 5
3 e
a 5
R1
R2
0 c
R2
1 b
d 5
3 e
a 5
R0
R2

Example

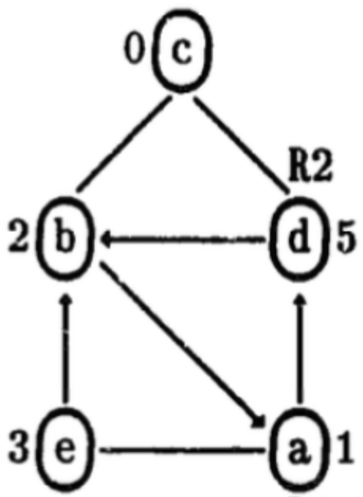


Legal Configuration

$$GST \equiv (\forall v \in V \setminus \{r\} | L_v = Lp_v + 1)$$

Definitions

- A parent pointer is called Well-Formed (WF) pointer if $L_v \neq n \wedge L_{p_v} \neq n \wedge L_v = L_{p_v} + 1$
- $S_v^{L_v}$ to denote the WF set rooted at v (tree structure)
    - Ex Fig1 initial $S_c^0 = \{c\}$,$S_a^1 = \{a, b, e\}$, $S_d^5 = \{d\}$.



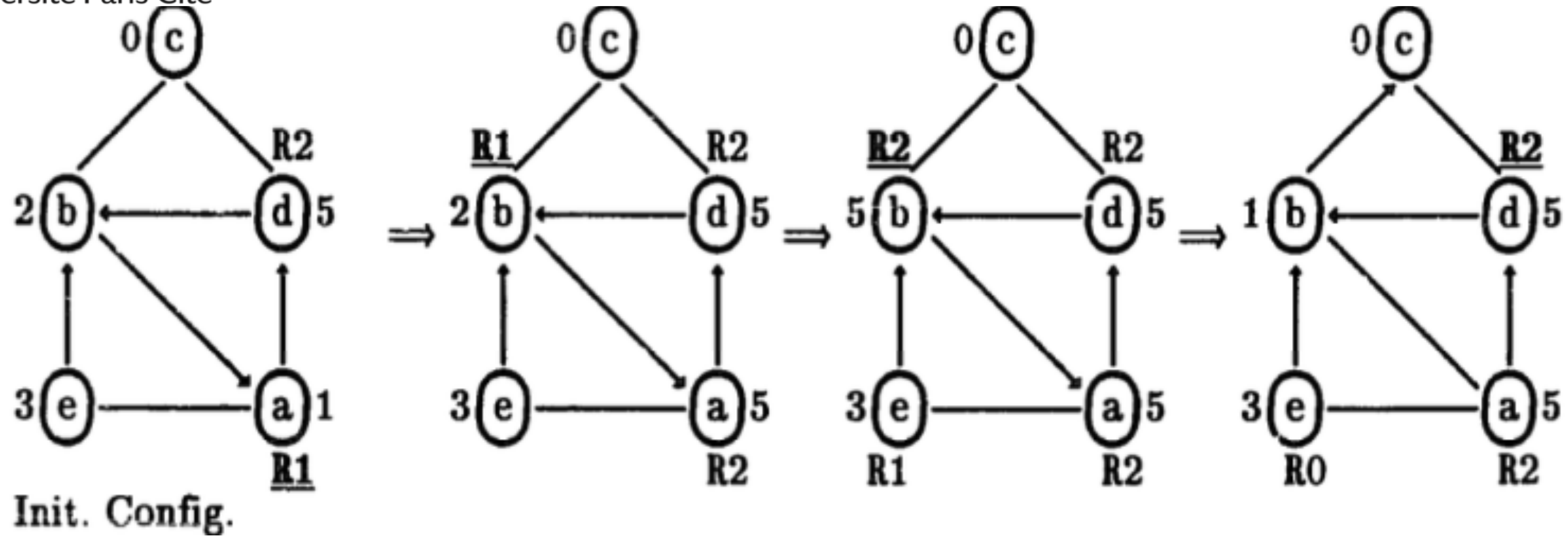
Lemma 1: Before GST is true, the algorithm does not terminate.

Proof: Before GST is true, there must exist some WF set $S_v^{L_v}$, with $v \neq r$. Two possible cases need to be considered

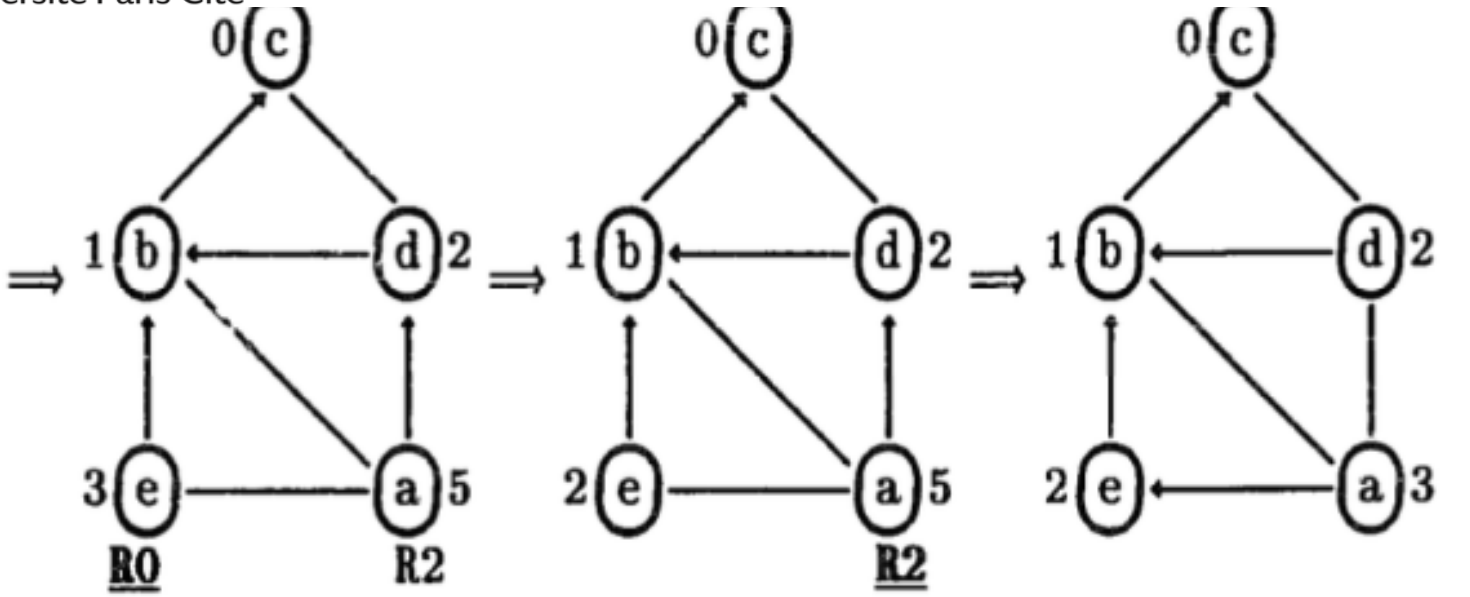
1. We consider $v \neq r$ if $L_v \neq n$ v is activatable by R_0 or R_1 because either $L_{p_v} \neq n$ or $L_{p_v} = n$
2. $L_v = n$ for every WF set $S_v^{L_v}$, $v \neq r$. Since G is connected, there exists at least one edge between a node u in S_r^0 and somme node v, $v \neq r$ and $S_v^{L_v} = S_v^n = \{v\}$. Because $L_r = 0$ and $|S_r^0| < n$, we have $L_u < n - 1$. Hence, node v can apply R_2 to make a move. $\square$

Definitions of function F

- Let γ be a configuration and let t_k: $0 \leq k \leq n$ be the the number of WF set $S_v^{L_v}$ such that $L_v = k$.
- $F(\gamma) = (t_0, t_1, \ldots, t_n)$ with $0 \leq i < n$
- The comparaison of F is lexicographic.
- F is bounded function with the maximum value $(1, n_1, 0, \ldots, 0)$ and minimum $(1, 0, \ldots, 0)$.



Step 0:	$S_c^0 = \{c\},\ S_a^1 = \{a, b, e\},\ S_d^5 = \{d\}$	$F = (1, 1, 0, 0, 0, 1)$
Step 1:	$S_c^0 = \{c\},\ S_b^2 = \{b, e\},\ S_a^5 = \{a\},\ S_d^5 = \{d\}$	$F = (1, 0, 1, 0, 0, 2)$
Step 2:	$S_c^0 = \{c\},\ S_e^3 = \{e\},\ S_a^5 = \{a\},\ S_b^5 = \{b\},\ S_d^5 = \{d\}$	$F = (1, 0, 0, 1, 0, 3)$
Step 3:	$S_c^0 = \{c, b\},\ S_e^3 = \{e\},\ S_a^5 = \{a\},\ S_d^5 = \{d\}$	$F = (1, 0, 0, 1, 0, 2)$
Step 4:	$S_c^0 = \{c, b, d\},\ S_e^3 = \{e\},\ S_a^5 = \{a\}$	$F = (1, 0, 0, 1, 0, 1)$
Step 5:	$S_c^0 = \{c, b, d, e\},\ S_a^5 = \{a\}$	$F = (1, 0, 0, 0, 0, 1)$
Step 6:	$S_c^0 = \{c, b, d, e, a\}$	$F = (1, 0, 0, 0, 0, 0)$



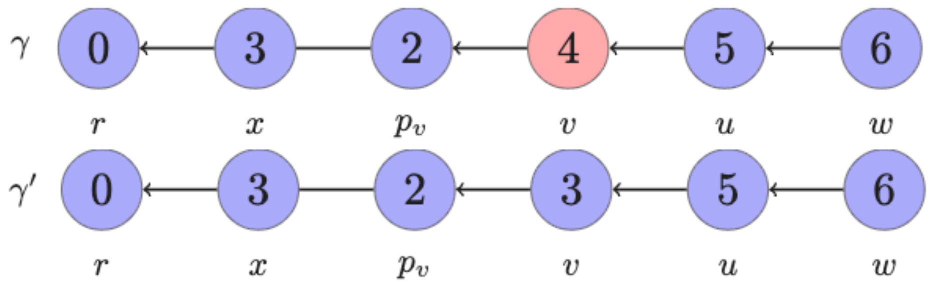
Step 0:	$S_c^0 = \{c\}, S_a^1 = \{a, b, e\}, S_d^5 = \{d\}$	$F = (1, 1, 0, 0, 0, 1)$
Step 1:	$S_c^0 = \{c\}, S_b^2 = \{b, e\}, S_a^5 = \{a\}, S_d^5 = \{d\}$	$F = (1, 0, 1, 0, 0, 2)$
Step 2:	$S_c^0 = \{c\}, S_e^3 = \{e\}, S_a^5 = \{a\}, S_b^5 = \{b\}, S_d^5 = \{d\}$	$F = (1, 0, 0, 1, 0, 3)$
Step 3:	$S_c^0 = \{c, b\}, S_e^3 = \{e\}, S_a^5 = \{a\}, S_d^5 = \{d\}$	$F = (1, 0, 0, 1, 0, 2)$
Step 4:	$S_c^0 = \{c, b, d\}, S_e^3 = \{e\}, S_a^5 = \{a\}$	$F = (1, 0, 0, 1, 0, 1)$
Step 5:	$S_c^0 = \{c, b, d, e\}, S_a^5 = \{a\}$	$F = (1, 0, 0, 0, 0, 1)$
Step 6:	$S_c^0 = \{c, b, d, e, a\}$	$F = (1, 0, 0, 0, 0, 0)$

Rule R_0

Lemma 2: F monotonically decreases each time when rule R_0 is applied.

Example of execution

$$R_0 : L_v \neq n \wedge L_v \neq L_{p_v} + 1 \wedge L_{p_v} \neq n \longrightarrow L_v = L_{p_v} + 1$$



$\gamma : S_r^0 = \{r\}, S_{p_v}^2 = \{p_v\}, S_x^3 = \{x\}, S_v^4 = \{v, u, w\} \longrightarrow F(\gamma) = (1, 0, 1, 1, 1, 0, 0)$

γ':

$S_r^0 = \{r\}, S_{p_v}^2 = \{p_v, v\}, S_x^3 = \{x\}, S_u^5 = \{u, w\} \longrightarrow F(\gamma') = (1, 0, 1, 1, 0, 1, 0)$

$\rightarrow F(\gamma) < F(\gamma')$

Proof of lemma 2

- Let us consider a node v, with $k = L_v(\gamma)$ such that $v \in S_v^k$ in γ so t_k is not null in γ;
- Let p_v be the parent of v in configuration γ, if v can execute R_0: $k_p = L_{p_v}(\gamma) \neq k - 1$.
- If the node v executes R_0 in configuration γ, it becomes element of $S_{p_v}^{k_p}$, so S_v^k disappears in γ' and $t_k(\gamma') < t_k(\gamma)$.

Proof of lemma 2

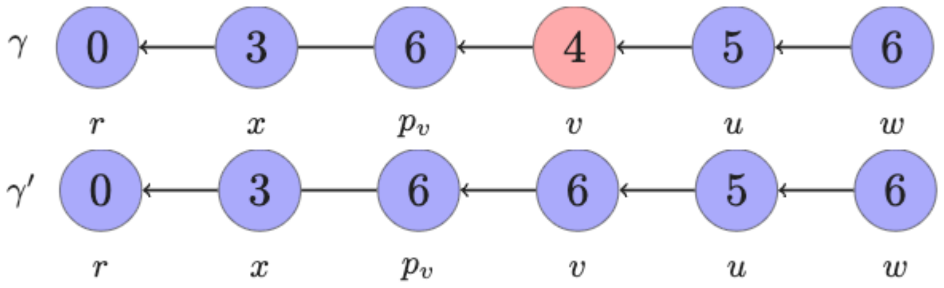
- Let denoted by u_i the children of v in configuration γ such that $L_{u_i} = k_i = k+1$, all the children was in $S_v^k(\gamma)$ but in γ' the child $u_i \in S_{u_i}^{k_i}$, so $t_{k_i}(\gamma') > t_k(\gamma)$, remember that $k_i = k+1$.
- As a consequence,
  $F(\gamma) = (1, \ldots, t_k(\gamma), t_{k_i}(\gamma), \ldots) < F(\gamma') = (1, \ldots, t_k(\gamma'), t_{k_i}(\gamma'), \ldots)$
  because $t_k(\gamma) < t_k(\gamma')$ and $t_{k_i}(\gamma) > t_{k_i}(\gamma')$.
  $\square$

Rule R_1

Lemma 3: F monotonically decreases each time when rule R_1 is applied.

Example of execution

$$R_1 : L_v \neq n \wedge L_{p_v} = n \longrightarrow L_v = n$$



γ: $S_r^0 = \{r\}, S_x^3 = \{x\}, S_v^4 = \{v, u, w\}, S^6 = \{p_v\} \longrightarrow F(\gamma) = (1, 0, 0, 1, 1, 0, 1)$

γ' :

$S_r^0 = \{r\}, S_x^3 = \{x\}, S_u^5 = \{u, w\}, S_{p_v}^6 = \{p_v\}, S_v^6 = \{v\} \longrightarrow F(\gamma') = (1, 0, 0, 1, 0, 1, 2$

$\rightarrow F(\gamma) < F(\gamma')$

Proof of lemma 3

- Let us consider a node v, with $L_v(\gamma) = k$, $k < n$ and $L_{p_v} = n$
- So in γ we have $v \in S_k^v$ and $p_v \in S_{p_v}^n$ as a consequence $t_k(\gamma)$ and $t_n(\gamma)$ are not nul;
- If the node v executes R_1 in configuration γ, it becomes in γ' element of S_v^n, and S_v^k disappears so $t_k(\gamma') < t_k(\gamma)$ and $t_n(\gamma') > t_n(\gamma)$.
- Like $k < n$ we obtain $F(\gamma') < F(\gamma)$.

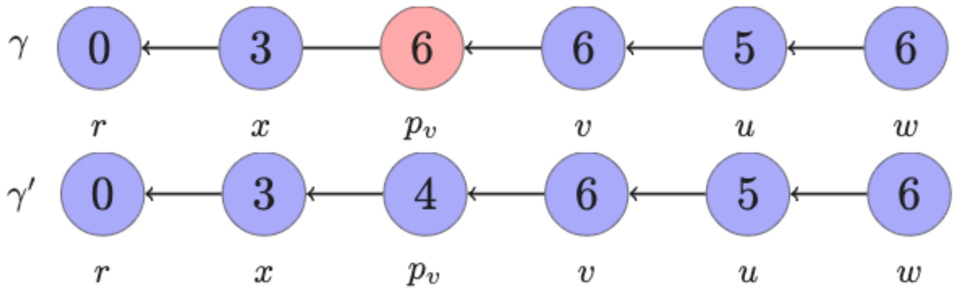
$\square$

Rule R_2

Lemma 4: F monotonically decreases each time when rule R_2 is applied.

Example of Execution of R_2

$$R_2 : L_v = n \wedge \exists u \in N(v) | L_u < n - 1 \longrightarrow L_v = L_u + 1; p_v = u$$



γ :

$S_r^0 = \{r\}, S_x^3 = \{x\}, S_u^5 = \{u, w\}, S_{p_v}^6 = \{p_v\}, S_v^6 = \{v\} \longrightarrow F(\gamma') = (1, 0, 0, 1, 0, 1, 2$

γ :

$S_r^0 = \{r\}, S_x^3 = \{x, p_v\}, S_u^5 = \{u, w\}, S_v^6 = \{v\} \longrightarrow F(\gamma') = (1, 0, 0, 1, 0, 1, 1)$

$\rightarrow F(\gamma) < F(\gamma')$

Proof of lemma 4

Proof: Let us consider a node v such that $v \in S_v^n$ in γ so $t_n(\gamma)$ is not null in γ;
After execution of rule R_2 by the node v in γ S_v^n disappears and $t_n(\gamma') = t_n(\gamma) - 1$, now v is in S_u^k and $t_k(\gamma') = t_k(\gamma')$ because v reaches a parent with the good level.
By definition of R_2 $k < n - 1$ so we obtain $F(\gamma') < F(\gamma)$.
$\square$

Theorem

> **Theorem** Eventually, the system reaches a legitimate configuration.

Proof: Since the initial value for F is finite, and the smallest possible value for F is $(1, 0, \ldots, 0)$, by Lemmas 2,3 and 4 the rules can only be applied a finite number of times. Hence, by Lemma 1, eventually GST is true.$\square$

Conclusion

- Silent self-stabilizing algorithm
- Centralized Scheduler
- Knowledge: n
- $O(\log_2 n)$ bits of memory per node
- Time complexity not provided

> Question: Is it space optimal?

A self-stabilizing algorithm for constructing breadth-first trees

Chen Huang 1992

Model

- Semi-uniform → root node denoted by r
- Anonymous
- Knowledge: n
- Scheduler : Distributed fair scheduler

Local variables

- Level : $L_v \in \{1, dots, n\}$
- Parent : $p_v \in \{0, \ldots, n\}$

- Remark: r has no parent and is level is zero
- These variables uses $O(\log)$ bits of memory by node

Algorithm

- $R_0 : L_v \neq L_{p_v} + 1 \wedge L_{p_v} \neq n \longrightarrow L_v = L_{p_v} + 1$
- $R_1 : L_v > k \longrightarrow L_v = k + 1; p_v = k_{id};$
  - with $k = \min\{L_u | u \in N(v)\}$ $k_{id} = \min\{id_u | u \in N(v) \wedge L_u = k\}$

Configuration legale

$$BFT \equiv (\forall v \in V \setminus \{r\} | L_v = Lp_v + 1 \wedge L_{p_v} = \min\{L_u | u \in N(v)\})$$

Remark: $BTF(v) \equiv L_v = Lp_v + 1 \wedge L_{p_v} = \min\{L_u | u \in N(v)\}$

Lélia Blin

No Deadlock

Lemma 1: before BFT is true the system never causes a deadlock

Proof

- if $n \geq 2$ the proof is trivial. Let us consider for $n > 2$.
- Proof by contradiction, so assuming BFT is false and no node can apply a rule.
- $\forall v \in V \setminus \{r\} : \neg R_0(v) \wedge \neg R_1(v) = true$
- $BTF(v) \equiv L_v = Lp_v + 1 \wedge L_{p_v} = \min\{L_u | u \in N(v)\} = false$
  1. $L_v \neq L_{p_v} + 1$ and $\neg R_0(v)$ → $L_{p_v} = n$
  2. $L_{p_v} > \min\{L_u | u \in N(v)\}$ and $\neg R_1(v)$ contradiction
  → (1) all the node v have $L_v = n$, $L_r = 0$ so the neighbors of r can applied R_1.

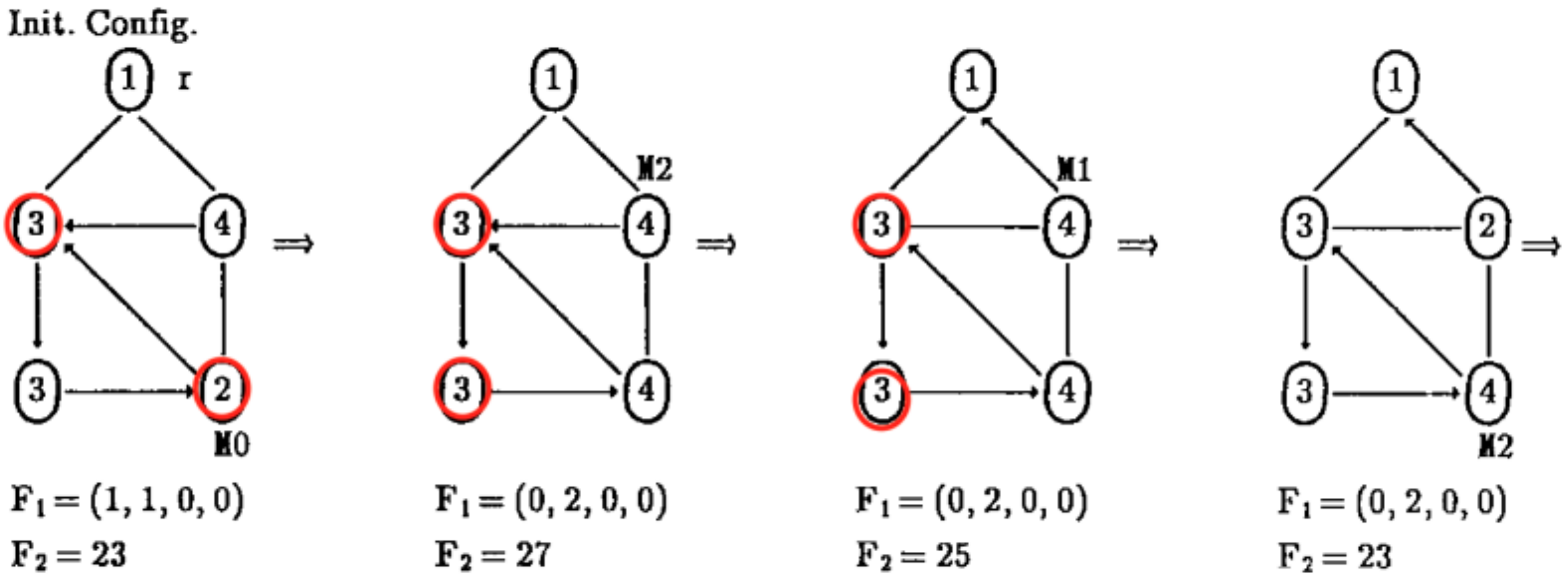
Modifications of the rules for the Correctness

- $M_0 : L_v \leq L_{p_v} < n \longrightarrow L_v = L_{p_v} + 1$
- $M_1 : L_v > L_{p_v} + 1 \longrightarrow L_v = L_{p_v} + 1$
- $M_2 : L_v \neq k \longrightarrow p_v = k_{id};$
  - with $k = \min\{L_u | u \in N(v)\}$ $k_{id} = \min\{id_u | u \in N(v) \wedge L_u = k\}$

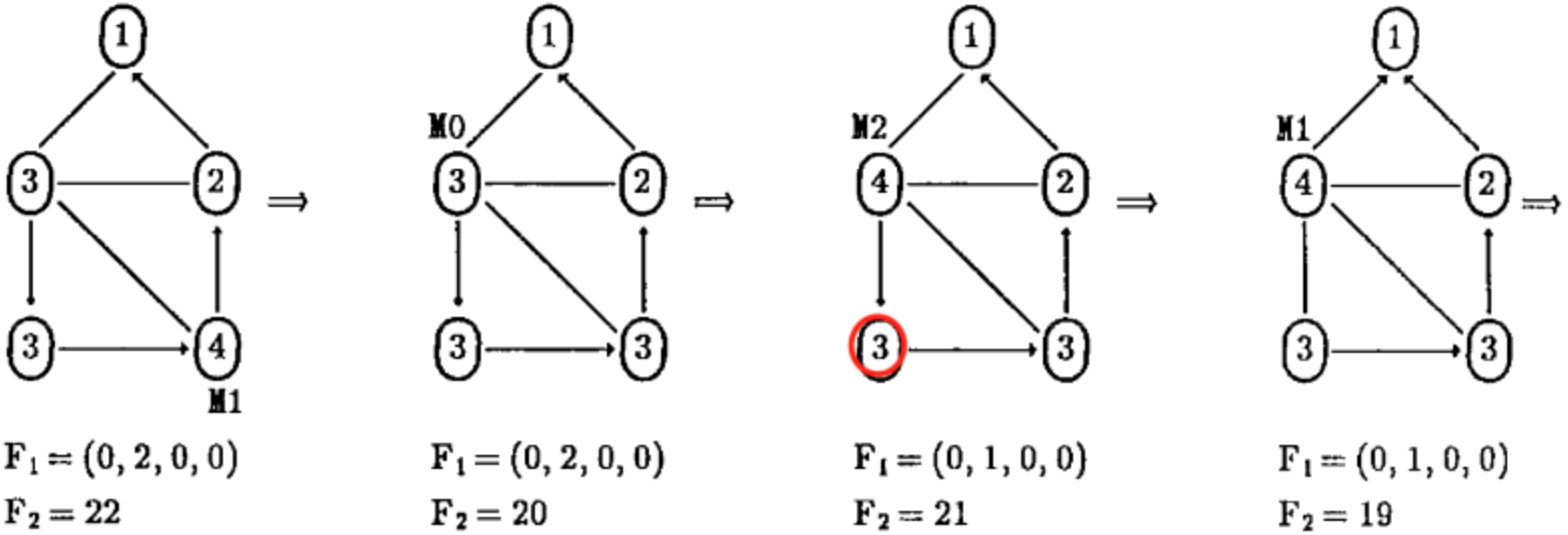
Potential Function

- $F \equiv (F_1, F_2)$
- $F_1 = (t_2, t_3, \ldots, t_n)$ where t_i is a number of nodes $v \in V$ such that $L_v = i$ and $L_v \leq L_{p_v}$ the node v is called an $i - turn$.
- $F_2 = \sum_{v \in V \setminus r} (L_v + L_{p_v})$

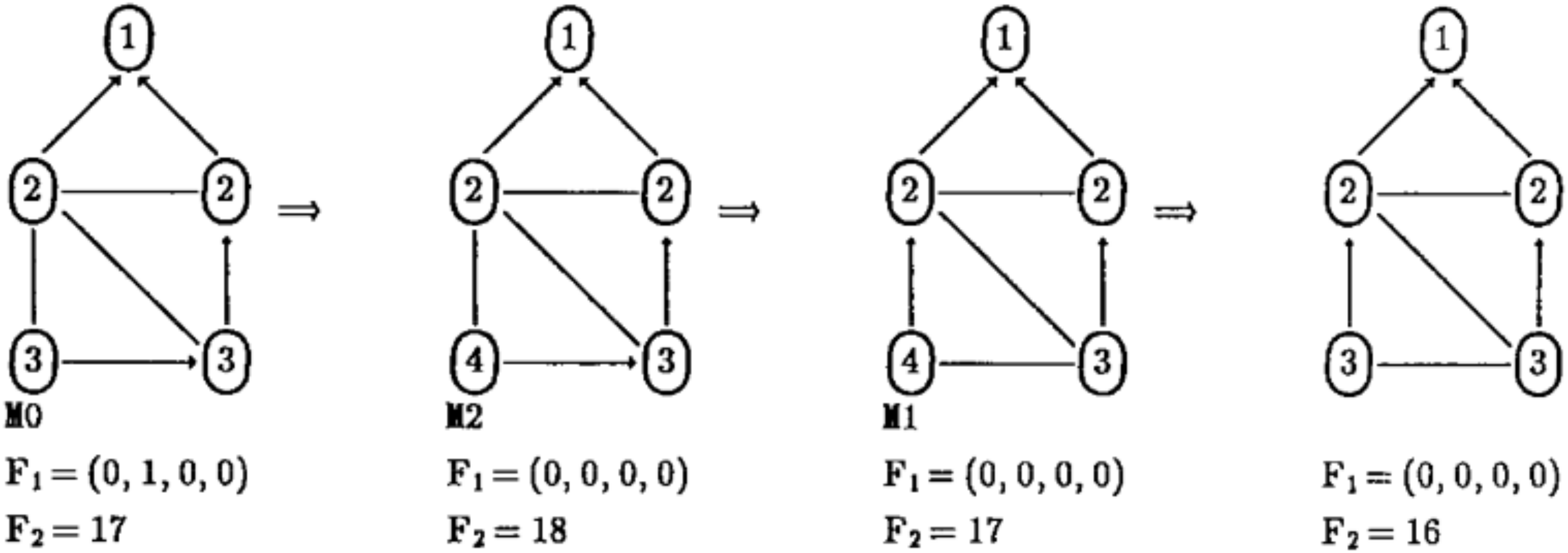
Examples


Init. Config.
1
r
3
4
3
2
M0
$F_1 = (1, 1, 0, 0)$
$F_2 = 23$
1
M2
3
4
3
4
$F_1 = (0, 2, 0, 0)$
$F_2 = 27$
1
M1
3
4
3
4
$F_1 = (0, 2, 0, 0)$
$F_2 = 25$
1
3
2
3
4
M2
$F_1 = (0, 2, 0, 0)$
$F_2 = 23$

Examples


1
3
2
3
4
M1
$F_1 = (0, 2, 0, 0)$
$F_2 = 22$
1
M0
3
2
3
3
$F_1 = (0, 2, 0, 0)$
$F_2 = 20$
1
M2
4
2
3
3
$F_1 = (0, 1, 0, 0)$
$F_2 = 21$
1
M1
4
2
3
3
$F_1 = (0, 1, 0, 0)$
$F_2 = 19$

Examples



Remark: To compute F_1 and F_2 only the tuple node parent of the node is considered.

Lemma 2: F_1 decreases each time when rule M_0 is applied.

Remember:

- $M_0 : L_v \leq L_{p_v} < n \longrightarrow L_v = L_{p_v} + 1$
- $F_1 = (t_2, t_3, \ldots, t_n)$ where t_i is a number of nodes $v \in V$ such that $L_v = i$ and $L_v \leq L_{p_v}$

Proof of lemma 2:

- Let v be a $k - turn$ node where $k = L_v(\gamma)$, so v can execute M_0 in configuration γ by definition of M_0 and t, after execution of node v , v does not stay a $k - turn$ node.
- Let now consider a node u child of node v such that
  - $L_u(\gamma) = L_v(\gamma) + 1$, so in γ u is not a $(k+1) - turn$ node but becomes $(k+1) - turn$ after activation of v, but $k + 1 > k$ so in we obtain $F_1(\gamma) < F_1(\gamma')$
  - $L_u(\gamma) \leq L_v(\gamma)$
  - $L_u(\gamma) > L_v + 1$

Lemma 3: F_2 decreases each time when rule M_1 or M_2 is applied.

Remember:

- $M_1 : L_v > L_{p_v} + 1 \longrightarrow L_v = L_{p_v} + 1$
- $M_2 : L_v \neq k \longrightarrow p_v = k_{id};$
  - with $k = \min\{L_u | u \in N(v)\}$ $k_{id} = \min\{id_u | u \in N(v) \wedge L_u = k\}$
- $F_2 = \sum_{v \in V \setminus r}(L_v + L_{p_v})$

Proof of Lemma 3

If a node v applied M_1 or M_2 then $L(v)$ decrease, the only way to increase F_2 is the use of M_0 but in this case thanks to lemma 2 F_1 decreases so F decreases.

Lemma 4: F_1 does not increase each time when rule M_1 or M_2 is applied.

Remember:

- $M_1 : L_v > L_{p_v} + 1 \longrightarrow L_v = L_{p_v} + 1$
- $M_2 : L_v \neq k \longrightarrow p_v = k_{id};$
  - with $k = \min\{L_u | u \in N(v)\}\ k_{id} = \min\{id_u | u \in N(v) \wedge L_u = k\}$
- $F_2 = \sum_{v \in V \setminus r}(L_v + L_{p_v})$

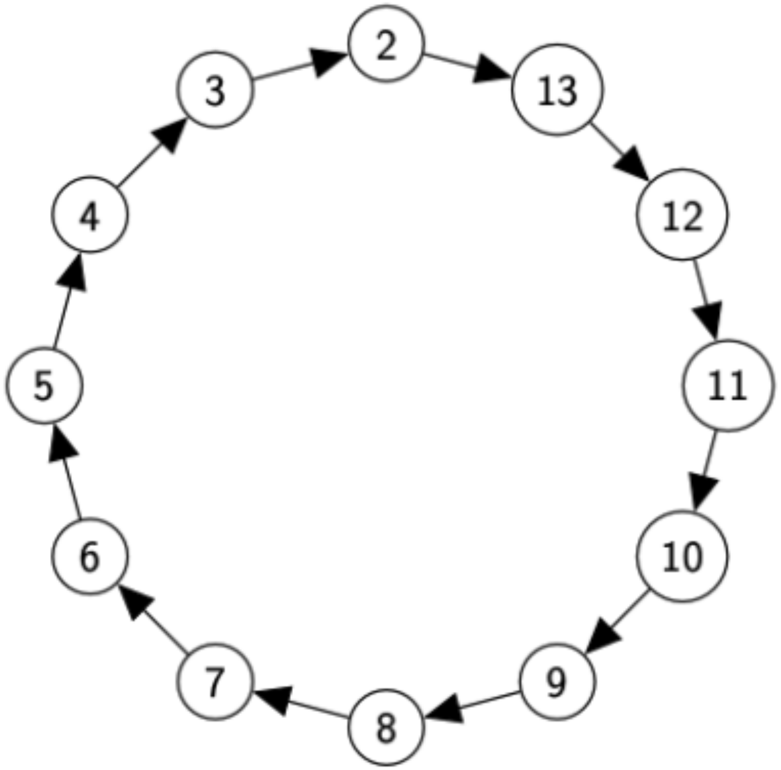
Theorem: Eventually, the system reaches a legitimate state.

Proof: Direct by lemmas 2,3,4.

Conclusion

- Silent self-stabilizing algorithm
- Distributed fair Scheduler
- Knowledge: n
- $O(\log_2 n)$ bits of memory per node
- Time complexity not provided

> Question: Is it space optimal?



Main technique to break cycles

Distance to the root

Freeze : Technic to destroy cycle

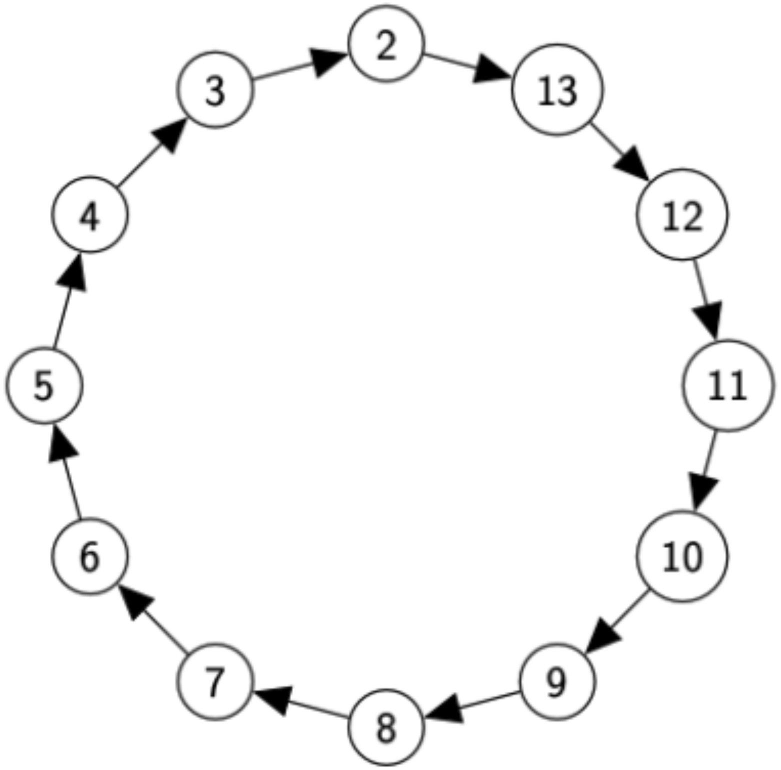
Blin Tixeuil 2017

Algorithm 4: Algorithm **Freeze**

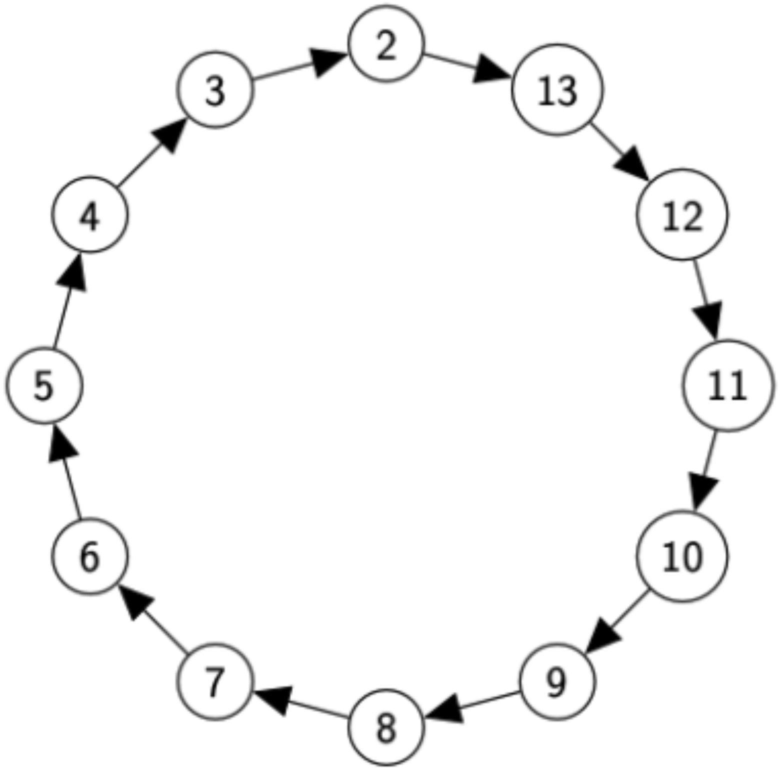
$$
\begin{array}{lll}
\mathbb{R}_{\texttt{Error}} & : & \mathrm{ErCycle}(v) \vee \mathrm{ErST}(v) \longrightarrow \mathsf{froz}_v := 1, \mathsf{p}_v := \emptyset; \\
\mathbb{R}_{\texttt{Froze}} & : & \neg\mathrm{ErCycle}(v) \wedge \neg\mathrm{ErST}(v) \wedge (\mathsf{frozp}_v = 1) \wedge (\mathsf{froz}_v = 0) \longrightarrow \mathsf{froz}_v := 1; \\
\mathbb{R}_{\texttt{Prun}} & : & \neg\mathrm{ErCycle}(v) \wedge \neg\mathrm{ErST}(v) \wedge (\mathsf{frozp}_v = 1) \wedge (\mathsf{froz}_v = 1) \wedge (\texttt{Ch}(v) = \emptyset) \longrightarrow \mathit{Reset}(v);
\end{array}
$$

▸ **Theorem 14.** *Algorithm* **Freeze** *deletes a cycle or an impostor-rooted sub spanning tree in n-nodes graph in a silent self-stabilizing manner, assuming the state model, and a distributed unfair scheduler. Moreover, Algorithm* **Freeze** *uses $O(1)$ bits of memory per node.*

▸ **Lemma 15.** *Algorithm* **Freeze** *converges in $O(n)$ steps.*

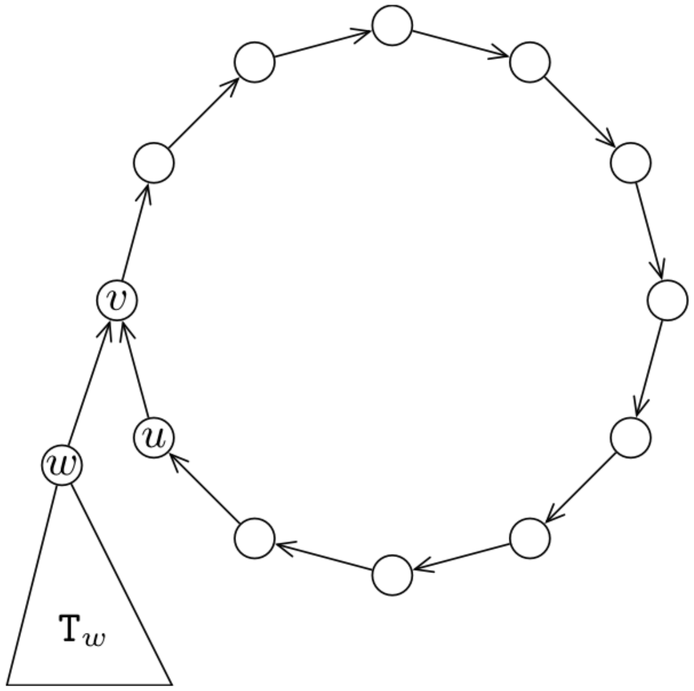


Freeze : Technic to destroy cycle



Other techniques

Number of children



Other techniques

> ID based algorithm: Unicity of the identity

Other technique: Non Silent, Semi-Uniform

Constant wave from the root

Fake subtree destruction

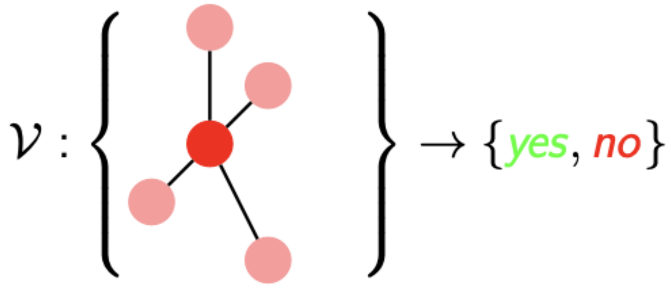
Space Optimality

Silent Self-Stabilizing Algorithm for Tree Construction

Proof Labeling Scheme vs Silent algorithm

Korman, Kutten, Peleg, 2007

- A proof-labeling scheme for a task is a pair such that:
- Prover assigns a certificate to every node.
- Verifier is a distributed algorithm such that



Configurations

Legal

(G, x) legal for $\mathcal{T} \Rightarrow \mathcal{P}$ assign certificates such that $\mathcal{V}$ accepts at all nodes.

Illegal

(G, x) illegal for $\mathcal{T} \Rightarrow$ for every certificates, $\mathcal{V}$ must reject in at least one node.

PLS : Spanning Tree

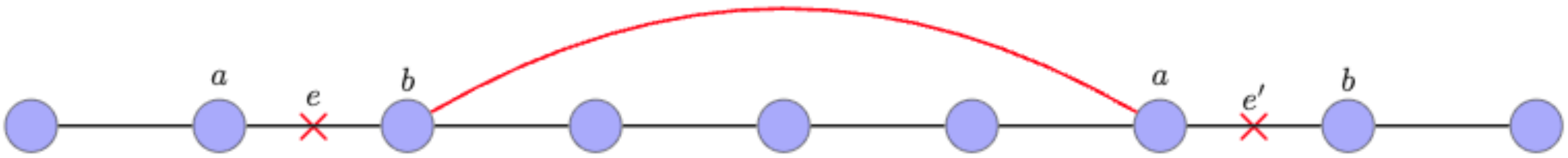
- $\mathcal{T} = \{(v, x(v)), v \in V\}$, $x(v)$ is the parent of v.
- $\mathcal{P}(v) = (Root, dis)$
- $\mathcal{V}(v) : \big(\forall u \in N(v) : (Root_u = Root_v)\big) \wedge (dis_v = dis_{p_v} + 1)$

Acyclicity

Theorem [Korman,Kutten,Peleg 2007]: Any PL for acyclicity requires certificate on $\Omega(\log_2 n)$ bits in n-node graph.

Proof

- Certificate k bits with $k \leq \frac{1}{2}\log_2 n - 1$
- $m = n - 1$ edges
- edges $\leftarrow \rightarrow$ (a, b): $2k$ bits
- different pair $2^{2k} = 2^{\log_2 n - 2} = \frac{1}{4}2^{\log_2 n} = \frac{n}{4}$
- $\exists e \neq e' | e \in E, e' \in E$ that correspond to the same pair



Silent Self-Stabilization vs. Proof-Labeling Scheme

Blin, Fraigniaud, Patt-Shamir 2014

	Size of registers	Number of rounds
Lower bound	$\Omega(\ell)$	
Algorithm 1	$O(\ell + \log n)$	$O(n2^{n\ell})$
Algorithm 2	$O(n^2 + nk)$	$O(n)$

- n-node networks,k-bit outputs, PLS of size at most ℓ bits

Question: Is there, for every task, a silent self-stabilizing algorithm for solving that task, that is simultaneously space-efficient and time-efficient?