INFORMATIQUE
Sciences
Université Paris Cité

Construction d'arbre couvrants Auto-stabilisants

M2

Lélia Blin

lelia.blin@irif.fr

Spécification du problème de l'arbre couvrant

Soit $G(V, E)$ un graphe non-dirigé connecté. Un sous-graphe acyclique qui relie tous les noeuds de G est appelé un arbre couvrant de G, noté $ST(G)$.

Notion de pointeur

Dans un arbre couvrant, la propriété acyclique garantit l'absence de cycles, tandis que l'obligation pour chaque nœud d'avoir un pointeur unique vers un voisin garantit un chemin distinct et sans ambiguïté entre les nœuds.

Construction d'un arbre couvrants auto-stabilisants

Principales difficultés

- Casser de cycle
- Pointeur de nœud vide (Fausses racines)

Arbres couvrants sous contrainte

- Arbre couvrant BFS
- Arbre couvrant DFS
- Arbre couvrant de poids minimum
- Arbre couvrant de degré minimum
- Arbre couvrant à diamètre minimum
- ....

Un algorithme auto-stabilisant pour la construction d'arbres couvrants

1991 Chen Yu Huang IPL

Modèle

- Semi-uniforme → nœud racine noté r
- Anonyme
- Connaissance : n
- Ordonnanceur : central (un seul nœud est activé à chaque étape)

Variables locales

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

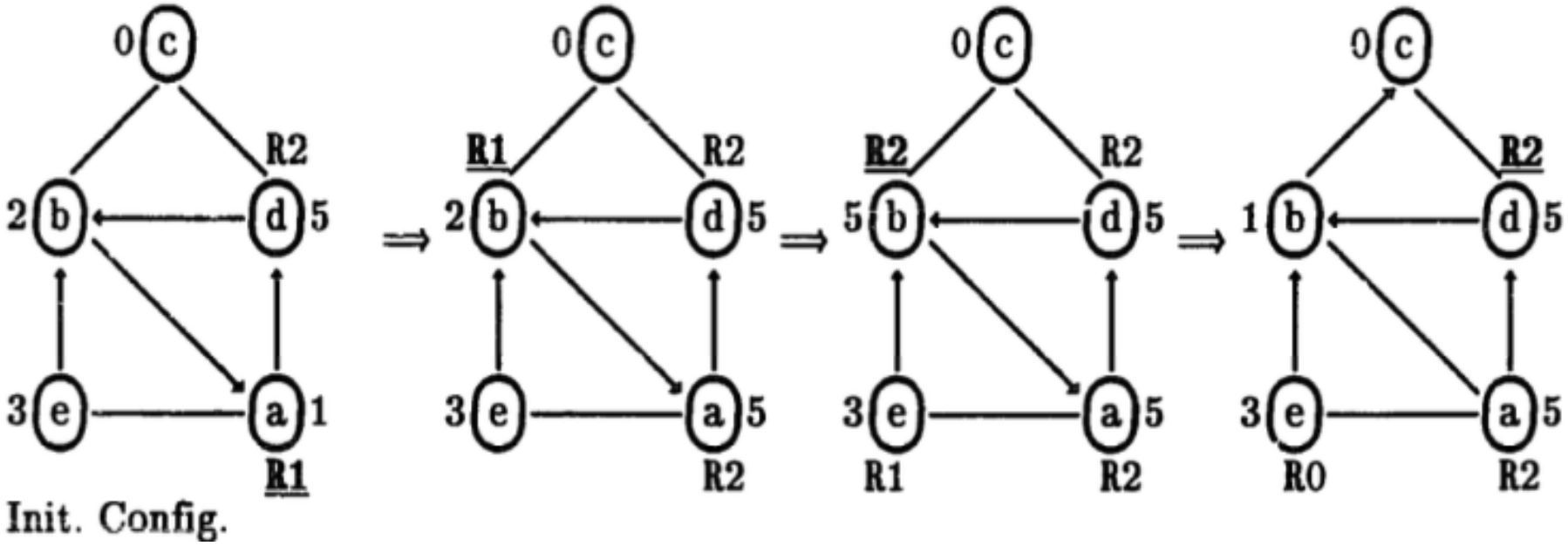
- Remarque : r n'a pas de parent et son niveau est zéro.
- Ces variables utilisent $O(\log_2 n)$ bits de mémoire par noeud

Algorithme

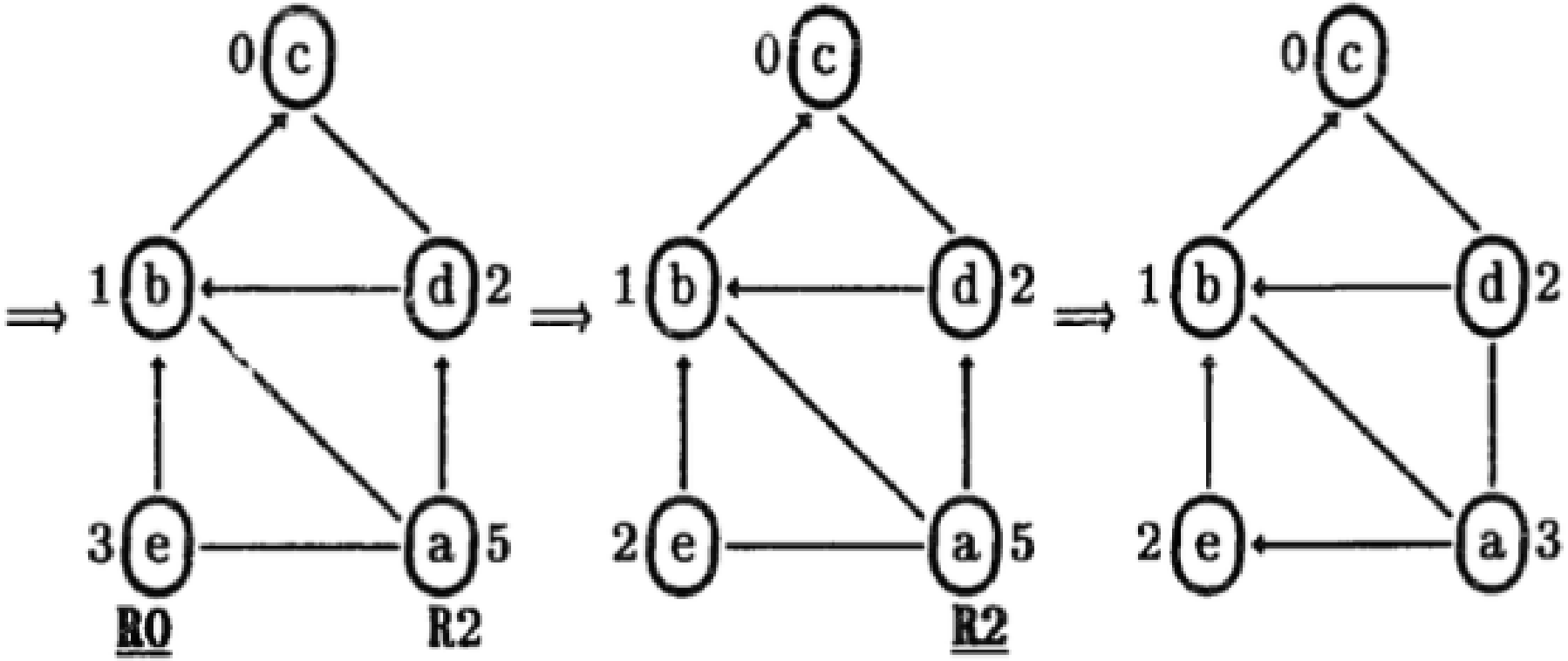
$$R_0 : L_v \neq n \wedge L_v \neq L_{p_v} + 1 \wedge L_{p_v} \neq n \longrightarrow L_v = L_{p_v} + 1$$
$$R_1 : L_v \neq n \wedge L_{p_v} = n \longrightarrow L_v = n$$
$$R_2 : L_v = n \wedge \exists u \in N(v) | L_u < n - 1 \longrightarrow L_v = L_u + 1; p_v = u$$

Lélia Blin

Exemple



Exemple

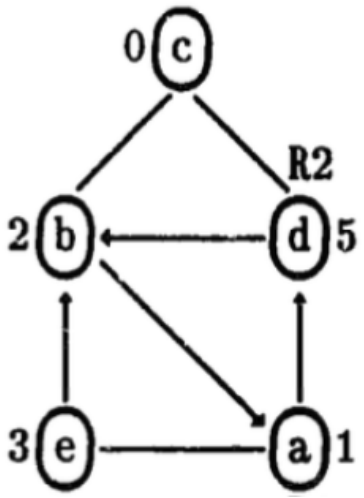


Configuration légale

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

Définitions

- Un pointeur parent est dit **Well-Formed** (WF) pointeur si
  $L_v \neq n \wedge L_{p_v} \neq n \wedge L_v = L_{p_v} + 1$
- $S_v^{L_v}$ pour désigner l'ensemble WF enraciné en v (structure arborescente)
  - Ex Fig1 initial $S_c^0 = \{c\}, S_a^1 = \{a, b, e\}, S_d^5 = \{d\}$.



interblocage 5no deadlock)

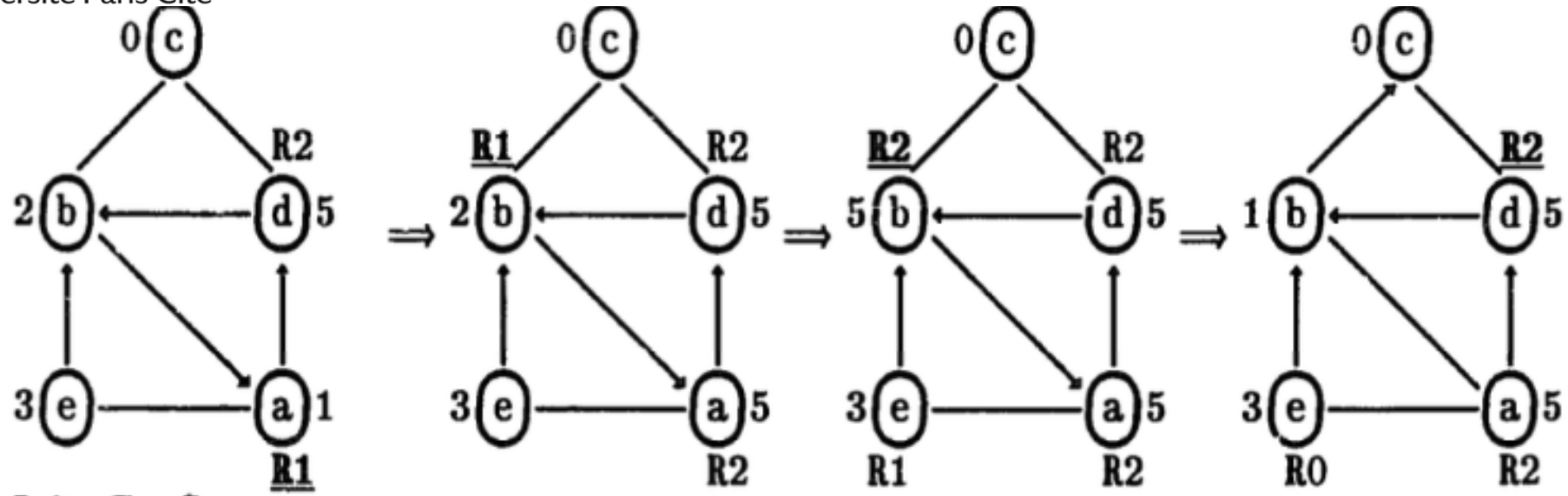
Lemme 1 : Avant que GST soit vrai, il existe au moins un noeud activable.

Preuve : Avant que GST soit vrai, il doit exister un ensemble WF $S_v^{L_v}$, avec $v \neq r$. Deux cas à considérés

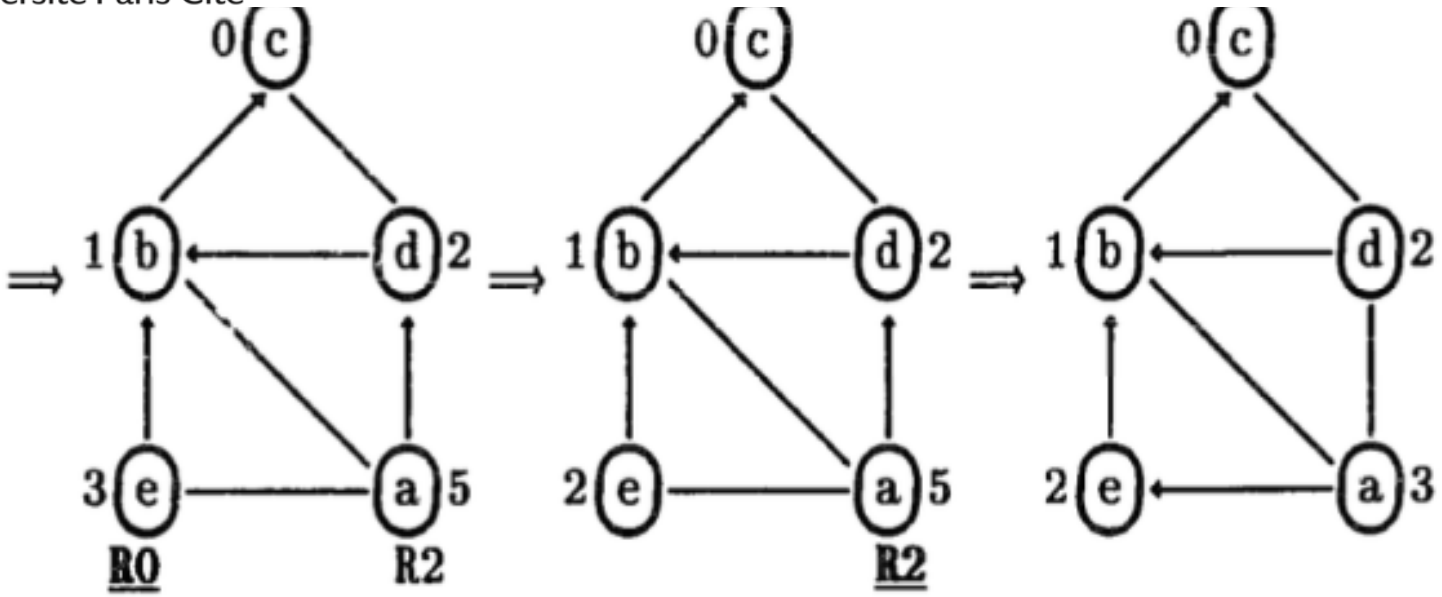
1. Nous considérons $v \neq r$ si $L_v \neq n$ v est activable par R_0 ou R_1 parce que $L_{p_v} \neq n$ ou $L_{p_v} = n$.
2. $L_v = n$ pour tout ensemble WF $S_v^{L_v}, v \neq r$. Puisque G est connexe, il existe au moins une arête entre un noeud u dans S_r^0 et un noeud $v, v \neq r$ et $S_v^{L_v} = S_v^n = \{v\}$. Comme $L_r = 0$ et $|S_r^0| < n$, nous avons $L_u < n - 1$. Par conséquent, le nœud v peut appliquer R_2 pour effectuer un déplacement. □

Définitions de la fonction F

- Soit γ une configuration et soit $t_k : 0 \leq k \leq n$ le nombre d'ensembles WF $S_v^{L_v}$ tels que $L_v = k$.
- $F(\gamma) = (t_0, t_1, \ldots, t_n)$ avec $0 \leq i < n$
- La comparaison de F est lexicographique.
- F est une fonction bornée dont la valeur maximale est $(1, n_1, 0, \ldots, 0)$ et la valeur minimale $(1, 0, \ldots, 0)$.



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:	$S_c^0 = \{c, b, d, e, a\}$	$F = (1, 0, 0, 0, 0, 0)$

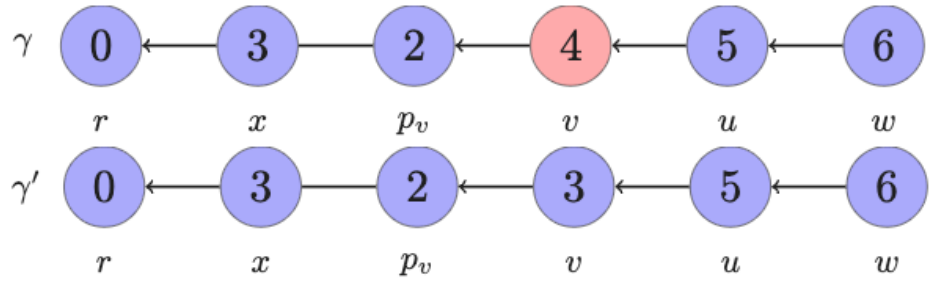
Règle R_0

Lemme 2 : F diminue monotonement à chaque fois que la règle R_0 est appliquée.

Lélia Blin

Exemple d'exécution

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



$\gamma : S_r^0 = \{r\}, S_{p_v}^2 = \{p_v\}, S_x^3 = \{x\}, S_v^4 = \{v, u, w\}$
$\longrightarrow F(\gamma) = (1, 0, 1, 1, 1, 0, 0)$
$\gamma' : S_r^0 = \{r\}, S_{p_v}^2 = \{p_v, v\}, S_x^3 = \{x\}, S_u^5 = \{u, w\}$
$\longrightarrow F(\gamma') = (1, 0, 1, 1, 0, 1, 0)$
conclusion $F(\gamma') < F(\gamma)$

Preuve du lemme 2

- Considérons un noeud v, avec $k = L_v(\gamma)$ tel que $v \in S_v^k$ dans γ de sorte que t_k n'est pas nul dans γ ;
- Soit p_v le parent de v dans la configuration γ, si v peut exécuter R_0 : $k_p = L_{p_v}(\gamma) \neq k - 1$.
- Si le nœud v exécute R_0 dans la configuration γ, il devient un élément de $S_{p_v}^{k_p}$, de sorte que S_v^k disparaît dans γ' et $t_k(\gamma') < t_k(\gamma)$.

Preuve du lemme 2

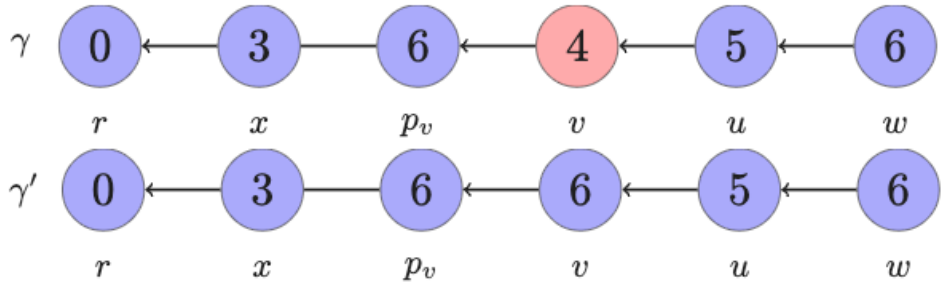
- Soit dénotés par u_i les enfants de v dans la configuration γ telle que $L_{u_i} = k_i = k + 1$, tous les enfants étaient dans $S_v^k(\gamma)$ mais dans γ' l'enfant $u_i \in S_{u_i}^{k_i}$, donc $t_{k_i}(\gamma') > t_k(\gamma)$, rappelons que $k_i = k + 1$.
- En conséquence, $F(\gamma) = (1, \ldots, t_k(\gamma), t_{k_i}(\gamma), \ldots) < F(\gamma') = (1, \ldots, t_k(\gamma'), t_{k_i}(\gamma'), \ldots)$ car $t_k(\gamma) < t_k(\gamma')$ et $t_{k_i}(\gamma) > t_{k_i}(\gamma')$.
  □

Règle R_1

Lemme 3 : F diminue monotonement à chaque fois que la règle R_1 est appliquée.

Exemple d'exécution

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



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

Preuve du lemme 3

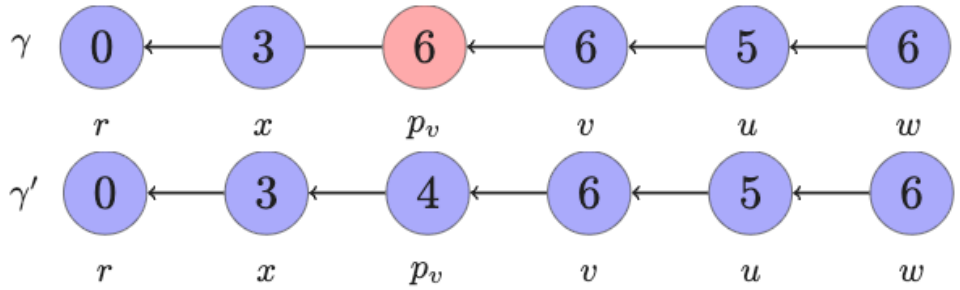
- Considérons un nœud v, avec $L_v(\gamma) = k, k < n$ et $L_{p_v} = n$
- Dans γ nous avons donc $v \in S_k^v$ et $p_v \in S_{p_v}^n$ par conséquent $t_k(\gamma)$ et $t_n(\gamma)$ ne sont pas nuls ;
- Si le noeud v exécute R_1 dans la configuration γ, il devient dans γ' un élément de S_v^n, et S_v^k disparaît donc $t_k(\gamma') < t_k(\gamma)$ et $t_n(\gamma') > t_n(\gamma)$.
- Comme $k < n$ on obtient $F(\gamma') < F(\gamma)$.
  □

Règle R_2

Lemme 4 : F diminue monotonement à chaque fois que la règle R_2 est appliquée.

Exemple d'exécution de R_2.

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



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

Preuve du lemme 4

La preuve du lemme 4 : Considérons un nœud v tel que $v \in S_v^n$ dans γ de sorte que $t_n(\gamma)$ n'est pas nul dans γ ;
Après l'exécution de la règle R_2 par le noeud v dans γ S_v^n disparaît et $t_n(\gamma') = t_n(\gamma) - 1$, maintenant v est dans S_u^k et $t_k(\gamma') = t_k(\gamma')$ parce que v atteint un parent avec le bon niveau.
Par définition de R_2 $k < n - 1$ on obtient donc $F(\gamma') < F(\gamma)$.
□

Théorème

> Le système finit par atteindre une configuration légitime.

La preuve : Puisque la valeur initiale de F est finie, et que la plus petite valeur possible pour F est $(1, 0, \ldots, 0)$, par les Lemmas 2,3 et 4 les règles ne peuvent être appliquées qu'un nombre fini de fois. Par conséquent, d'après le lemme 1, GST est finalement vrai.□

Conclusion

- Algorithme silencieux auto-stabilisant
- Ordonnanceur centralisé
- Connaissance : n
- $O(\log_2 n)$ bits de mémoire par noeud
- Complexité temporelle non fournie

> Question : L'espace mémoire est-il optimal ?

Algorithme auto-stabilisant pour la construction d'arbres BFS.

Chen Huang 1992

Modèle

- Semi-uniforme → nœud racine désigné par r
- Anonyme
- Connaissance : n
- **Ordonnanceur : Ordonnanceur équitable distribué**

Variables locales

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

- Remarque : r n'a pas de parent et son niveau est zéro
- Ces variables utilisent $O(\log_2 n)$ bits de mémoire par noeud

Algorithme

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

Configuration legale

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

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

Interblocage

Lemma 1 : avant que BFT soit vrai, le système ne provoque jamais de blocage.

Autrement dit au moins un noeud est activable.

Preuve

- si $n \geq 2$ la preuve est triviale. Considérons pour $n > 2$.
- Preuve par contradiction, supposons que BFT est faux et qu'aucun noeud ne peut appliquer une règle.
- Pour tout v dans V, il y a un minimum de règles : $\neg R_0(v) \wedge \neg R_1(v) = true$
- $BTF(v) \equiv L_v = Lp_v + 1 \wedge L_{p_v} = \min\{L_u | u \in N(v)\} = false$
  1. $L_v \neq L_{p_v} + 1$ et $\neg R_0(v)$ → $L_{p_v} = n$
  2. $L_{p_v} > \min\{L_u | u \in N(v)\}$ et $\neg R_1(v)$ contradiction

  → (1) tous les noeuds v ont $L_v = n, L_r = 0$ donc les voisins de r peuvent appliquer R_1.

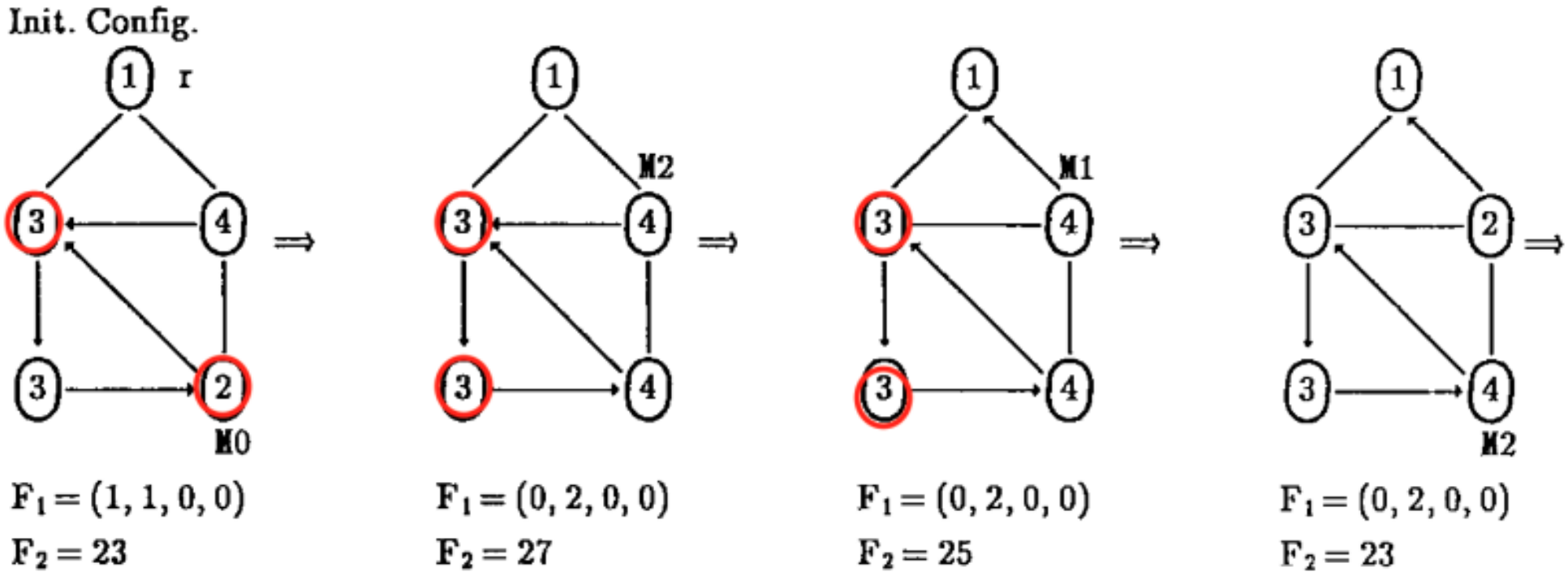
Modifications des règles de correction

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

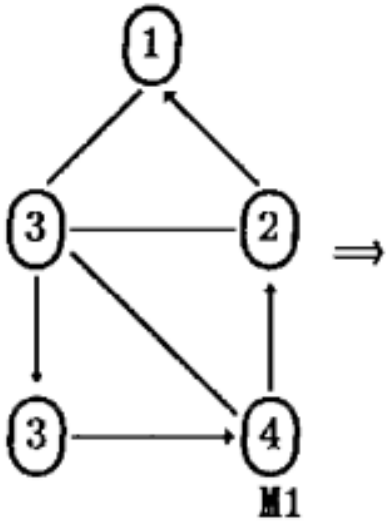
Fonction potentielle

- $F \equiv (F_1, F_2)$
- $F_1 = (t_2, t_3, \ldots, t_n)$ où t_i est un nombre de noeuds $v \in V$ tel que $L_v = i$ et $L_v \leq L_{p_v}$ le noeud v est appelé un $i - turn$.
- $F_2 = \sum_{v \in V \setminus r}(L_v + L_{p_v})$

Exemples

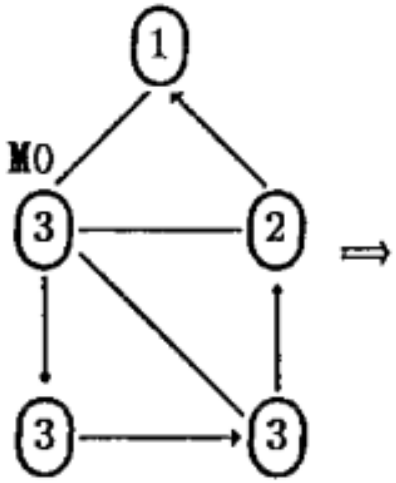

Init. Config.
1
r
3
4
3
2
M0
$F_1 = (1, 1, 0, 0)$
$F_2 = 23$
1
M2
3
4
3
4
$F_1 = (0, 2, 0, 0)$
$F_2 = 27$
1
M1
3
4
3
4
$F_1 = (0, 2, 0, 0)$
$F_2 = 25$
1
3
2
3
4
M2
$F_1 = (0, 2, 0, 0)$
$F_2 = 23$

Exemples



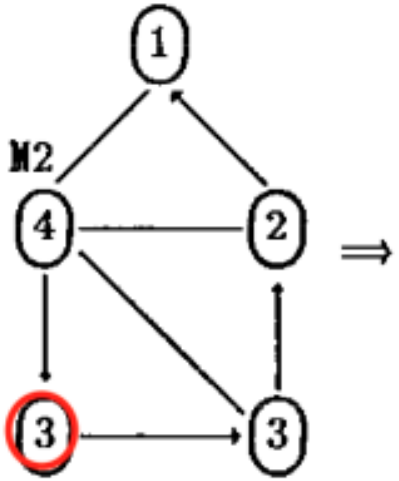
$F_1 = (0, 2, 0, 0)$

$F_2 = 22$



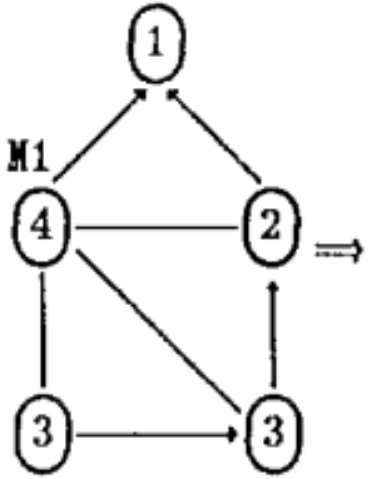
$F_1 = (0, 2, 0, 0)$

$F_2 = 20$



$F_1 = (0, 1, 0, 0)$

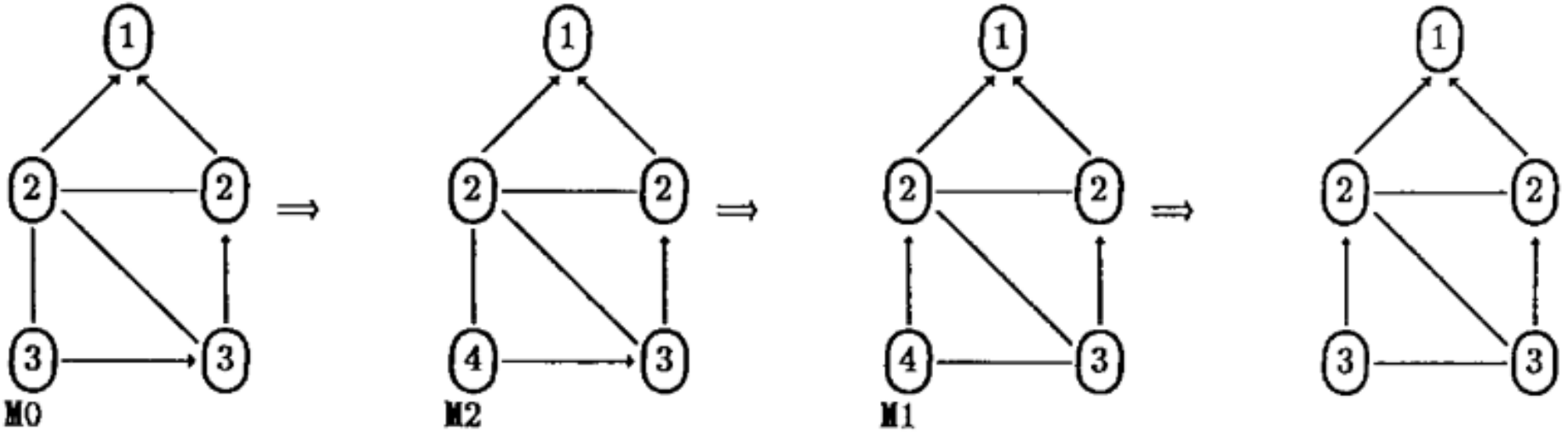
$F_2 = 21$



$F_1 = (0, 1, 0, 0)$

$F_2 = 19$

Exemples


1
2
2
3
3
M0
$F_1 = (0, 1, 0, 0)$
$F_2 = 17$
1
2
2
4
3
M2
$F_1 = (0, 0, 0, 0)$
$F_2 = 18$
1
2
2
4
3
M1
$F_1 = (0, 0, 0, 0)$
$F_2 = 17$
1
2
2
3
3
$F_1 = (0, 0, 0, 0)$
$F_2 = 16$

Remarque :

Pour calculer F_1 et F_2, seul le tuple parent du noeud est pris en compte.

Schéma 2 : F_1 diminue à chaque fois que la règle M_0 est appliquée.

Rappelez-vous :

- $M_0 : L_v \leq L_{p_v} < n \longrightarrow L_v = L_{p_v} + 1$
- $F_1 = (t_2, t_3, \ldots, t_n)$ où t_i est un nombre de noeuds $v \in V$ tel que $L_v = i$ et $L_v \leq L_{p_v}$.

Preuve du lemme 2 :

- Soit v un noeud $k-turn$ où $k = L_v(\gamma)$, donc v peut exécuter M_0 dans la configuration γ par définition de M_0 et t, après l'exécution du noeud v, v ne reste pas un noeud $k-turn$.
- Considérons maintenant un nœud u enfant du nœud v tel que
  - $L_u(\gamma) = L_v(\gamma) + 1$, donc dans γ u n'est pas un noeud $(k+1)-turn$ mais devient $(k+1)-turn$ après l'activation de v, mais $k+1 > k$ donc dans nous obtenons $F_1(\gamma) < F_1(\gamma')$
  - $L_u(\gamma) \leq L_v(\gamma)$
  - $L_u(\gamma) > L_v + 1$

Lemme 3 : F_2 diminue à chaque fois que la règle M_1 ou M_2 est appliquée.

Remember:

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

Preuve du lemme 3

Si un noeud v applique M_1 ou M_2 alors $L(v)$ diminue, la seule façon d'augmenter F_2 est d'utiliser M_0 mais dans ce cas grâce au lemme 2 F_1 diminue donc F diminue.

Lemma 4 : F_1 n'augmente pas à chaque fois que la règle M_1 ou M_2 est appliquée.

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 $k = \min\{L_u | u \in N(v)\}$ $k_{id} = \min\{id_u | u \in N(v) \wedge L_u = k\}$
- $F_2 = \sum_{v \in V \setminus r}(L_v + L_{p_v})$

> Théorème : Le système finit par atteindre un état légitime.

Preuve : Directe par les lemmes 2,3,4.

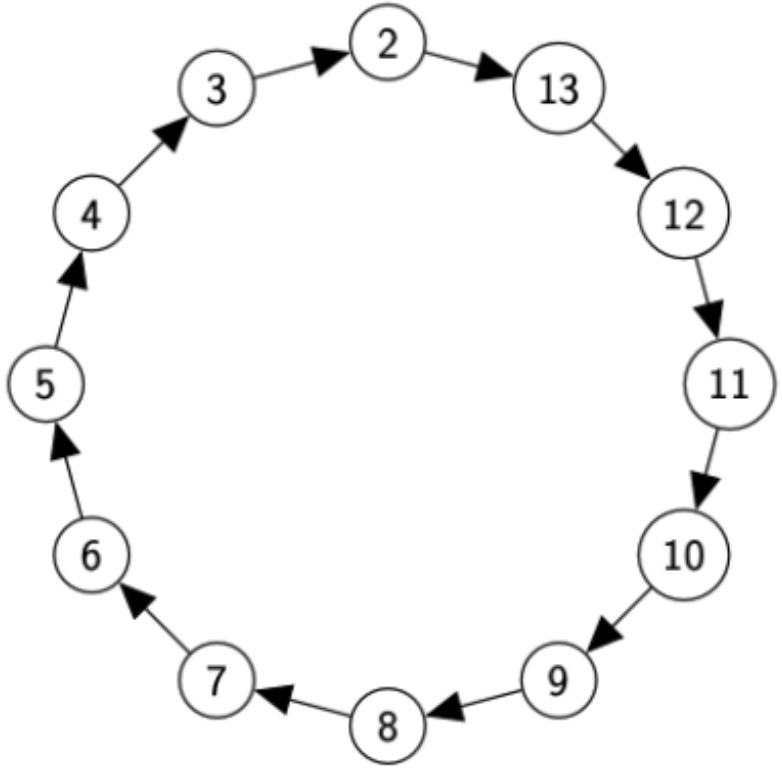
Conclusion

- Algorithme silencieux auto-stabilisant
- Ordonnanceur équitable distribué
- Connaissance : n
- $O(\log_2 n)$ bits de mémoire par noeud
- Complexité temporelle non fournie

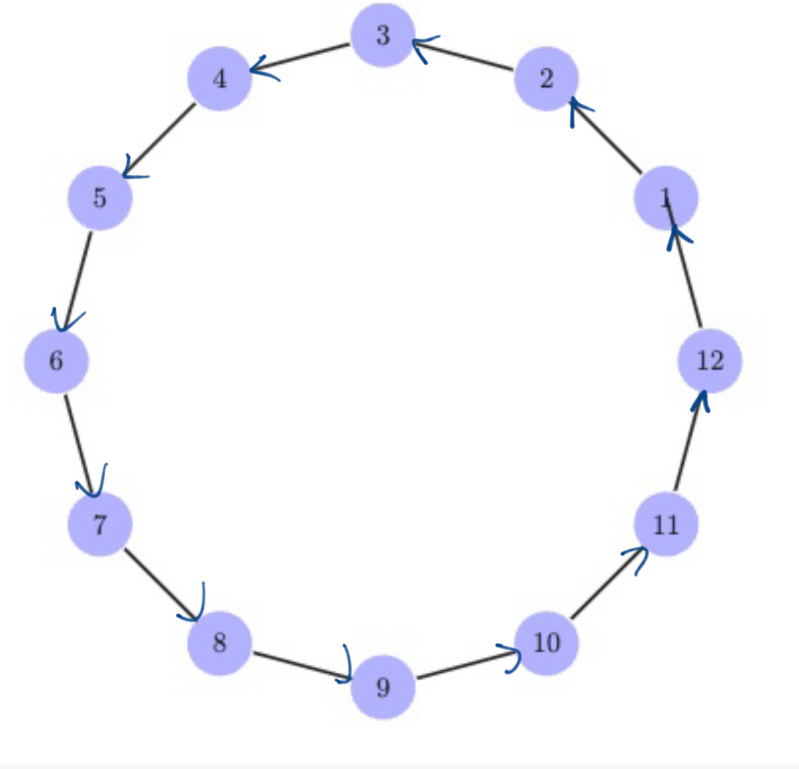
> Question : L'espace mémoire est-il optimal ?

Techniques pour casser les cycles

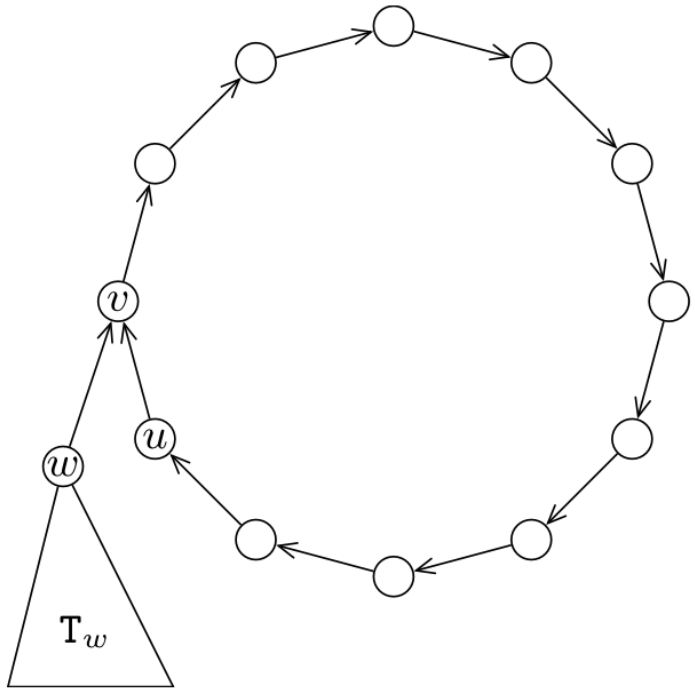
sans connaissance globale



Principale: Distance par rapport à la racine



Nombre d'enfants



Unicité de l'identité

Algorithme basé sur l'identité :

Freeze : La technique pour détruire le cycle

Blin Tixeuil 2017

Algorithm 4: Algorithm **Freeze**

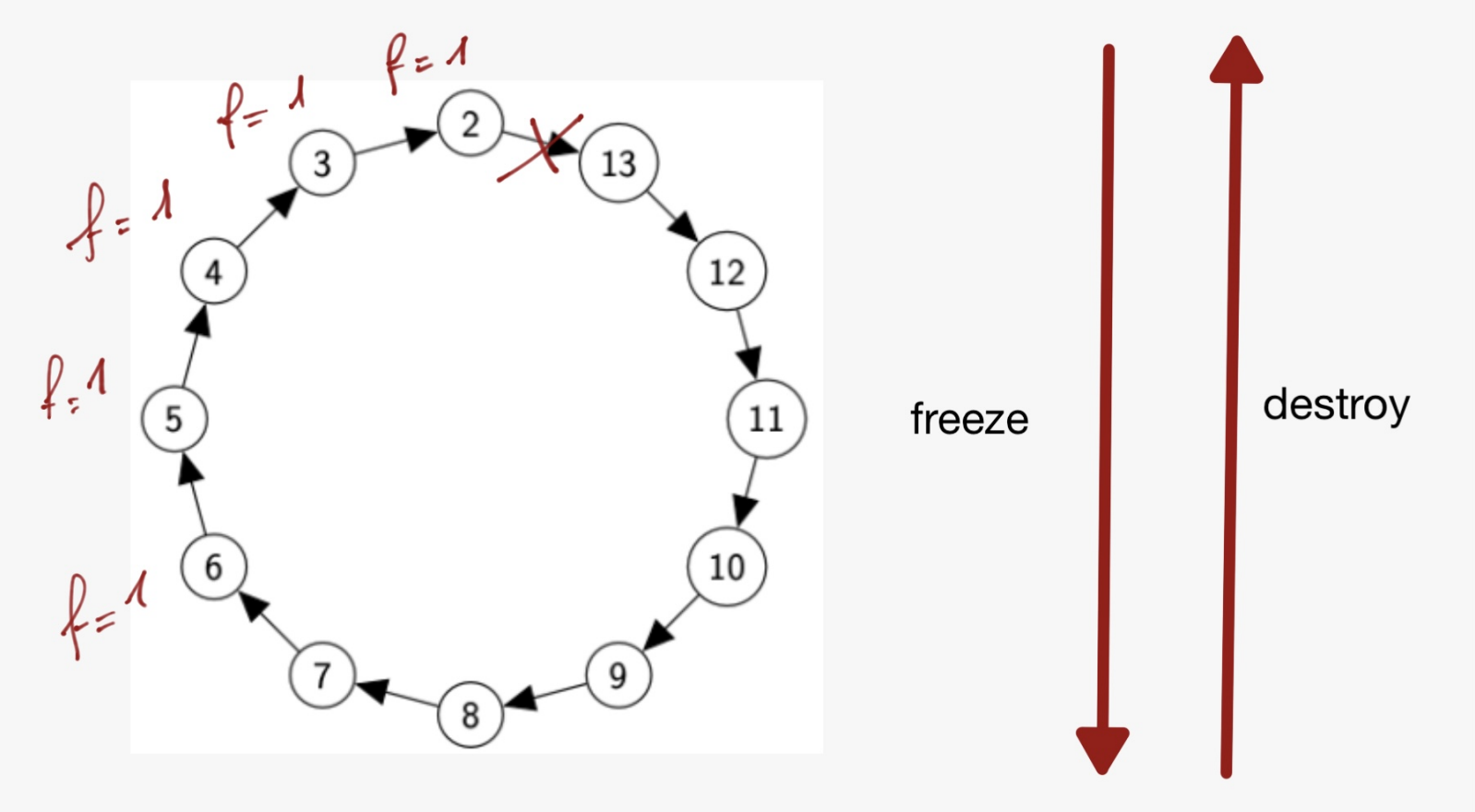
$$
\begin{array}{lcll}
\mathbb{R}_{\texttt{Error}} & : & \mathrm{ErCycle}(v) \vee \mathrm{ErST}(v) & \longrightarrow \mathsf{froz}_v := 1, \mathsf{p}_v := \varnothing; \\
\mathbb{R}_{\texttt{Froze}} & : & \neg\mathrm{ErCycle}(v) \wedge \neg\mathrm{ErST}(v) \wedge (\mathsf{froz}_{\mathsf{p}_v} = 1) \wedge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

▸ **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.*

Lélia Blin

Freeze



	Articles	Semi-unif.	Anonyme	Knowledge	Communications	Scheduler	Équité	Atomicité	Silent	Espace mémoire	Temps de convergence	Propriété
ST	ChenYuHuang91	✓		n	R	Central		$\oplus$	✓	$O(\log n)$	non fourni	
	AggarwalK93	X	X	X	R	C	f	$\oplus$	✓	$O(\log n)$	$O(D)$	dyn
DFS	HuangC93	✓	✓	n	R&D					$O(\log \Delta n)$		
	CollinD94	✓	✓		R	C	f	$\oplus$		$O(n \log \Delta)$	$O(Dn\Delta)$	
	HuangW97		✓	n	R	C	f			$O(\log n)$		
	DattaJPV00	✓	✓		R&D		f			$O(\log \Delta)$	$O(Dn\Delta)$	
BFS	DolevIM90	✓	✓		R	Central	f	$\oplus$		$O(\Delta \log n)$	$O(D)$	dyn
	AroraG90			n	R	C				$O(\log n)$	$O(n^2)$	dyn
	AfekKY91				R&D		f	$\oplus$		$O(\log n)$	$O(n^2)$	dyn
	HuangC92	✓	✓	n	R&D	Inéquitable		$\oplus$	✓	$O(\log n)$	non fourni	
	Dolev93				R	C	f	$\oplus$		$O(\Delta n \log n)$	$\Theta(D)$	dyn
	Johnen97	✓	✓		R			f	X	$O(\log \Delta)$	$O(n)$	
	AfekB98				M&D				✓	$O(\log n)$		$O(n)$
	HuangL02	✓	✓		R	C	I			$O(\log n)$		
	DattaLV08				R&D		f	$\oplus$		$O(\log n)$	$O(n)$	
	CournierRV19											
	DattaDJL23	✓	✓		R		U	$\oplus$	X	$O(\log \Delta)$	$O(D.n^2)$	

Espace mémoire

Avec la propriété de silence espace mémoire $\Omega(\log_2 n)$ bits (uniforme ou semi-uniforme)

Sans la propriété de silence: la meilleur complexité mémoire connue $O(\Delta + \log_2 \log_2 n)$ bits par noeud (uniforme).