Algorithmique Distribuée : TD 4, algorithmes silencieux

Alain Cournier Stéphane Devismes

8 juin 2026

1 Ensemble Indépendant Maximal

Soit $G = (V, E)$ un graphe simple et non-orienté où V est l'ensemble des sommets et E l'ensemble des arêtes. Soit $I \subseteq V$.

— I est un *ensemble indépendant* (de G) si et seulement si pour toute paire de sommets distincts $\{p, q\}$ de I on a p et q qui ne sont pas voisins, c'est-à-dire, $\{p, q\} \notin E$.

— I est un *ensemble indépendant maximal* (de G) si et seulement si I est indépendant et tout sur-ensemble (propre) de I ne l'est pas.

Question 1. Nous considérons le graphe donné en figure 1. Pour chacun des sous-ensembles suivants dites s'il est indépendant et indépendant maximal. Justifiez les réponses négatives.

	Indépendant	Indépendant maximal	Justification
$\emptyset$	oui	non	on peut ajouter n'importe quel sommet
$\{5\}$	oui	non	on peut ajouter 2, par exemple
$\{5, 6\}$	oui	non	on peut ajouter 2, par exemple
$\{5, 6, 8\}$	oui	non	on peut ajouter 2, par exemple
$\{1, 2, 5\}$	non	non	$\{1, 5\} \in E$
$\{3, 6, 7\}$	oui	oui	
$\{2, 5, 6, 8\}$	oui	oui	
$\{1, 2, 8, 9\}$	oui	non	on peut ajouter 4, par exemple
$\{1, 3, 7, 9\}$	oui	oui	
$\{2, 5, 8, 9\}$	non	non	$\{5, 9\} \in E$
$\{1, 2, 4, 8, 9\}$	oui	oui	
$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$	non	non	$\{1, 6\} \in E$

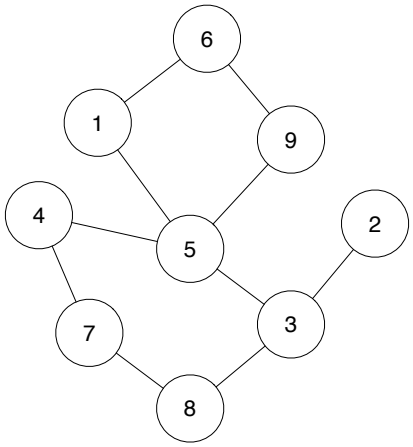

FIGURE 1 – Réseau quelconque.

2 Algorithme Autostabilisant calculant un Ensemble Indépendant Maximal

2.1 Dans un Réseau Anonyme

Nous rappelons que dans un réseau anonyme les processus sont indistinguables sauf lorsque leur degré est différent. Donc, ils ont le même algorithme avec en particulier les mêmes variables.

Question 2. Proposez un exemple de réseau anonyme et de démon pour lesquels il n'existe aucun algorithme déterministe autostabilisant (dans le modèle à états) calculant un ensemble indépendant maximal. Justifiez.

On choisit un réseau en anneau et un démon synchrone. S'il existait un tel algorithme, alors il devrait pouvoir stabiliser même lorsque tous les processus ont un état identique dans la configuration initiale de l'anneau. Or dans une telle configuration soit tous les processus sont dans l'ensemble et celui-ci n'est pas indépendant, soit aucun et l'ensemble n'est pas maximal.

Dans les deux cas, tous les processus sont activables et donc activés lors de la prochaine étape de calcul. L'algorithme étant déterministe, ils sont à nouveau dans un état identique, *etc.*. L'exécution n'atteint jamais une configuration où il y a un ensemble indépendant maximal de défini.

Question 3. Généralisez la réponse précédente.

L'exemple précédent se généralise aux graphes réguliers d'au moins deux nœuds. Ensuite, un démon distribué fortement équitable, faiblement équitable ou inéquitable peut se comporter comme un démon synchrone.

Plus généralement, il n'existe pas d'algorithme autostabilisant déterministe pour le calcul d'un ensemble indépendant maximal qui fonctionne dans des graphes quelconques avec un démon distribué.

2.2 Dans un Réseau Identifié

Nous considérons maintenant des réseaux bidirectionnels (l'hypothèse connexe n'est pas nécessaire) $G = (V, E)$ avec identité. $\forall p \in V$, id_p désigne l'identité de p.

2.2.1 Les Algorithmes Silencieux

Dans le modèle à états, un algorithme autostabilisant est *silencieux* si toutes ses exécutions atteignent une configuration dites *terminale* où plus aucun processus n'est activable. Généralement, les algorithmes silencieux sont utilisés pour calculer des structures sur le réseaux, comme par exemple un arbre couvrant.

2.2.2 Algorithme Silencieux calculant un Ensemble Indépendant Maximal

Nous considérons le modèle à états avec un démon distribué inéquitable. L'algorithme consiste à calculer une variable de sortie Booléenne $S_p \in \{Dominant, dominé\}$ de telle manière que l'ensemble $I = \{p \in V, S_p = Dominant\}$ est un ensemble indépendant maximal.

L'idée générale de l'algorithme est la suivante : Un processus doit être *Dominant* si et seulement si chacun de ses voisins est soit *dominé*, soit d'identité plus grande.

L'algorithme comporte deux règles *Leave* et *Join*. Un processus exécute *Leave* pour quitter l'ensemble indépendant. Un processus exécute *Join* pour entrer dans l'ensemble indépendant. L'ensemble des voisins d'un processus p donné sera noté $\mathcal{N}_p$.

Question 4. Écrivez les deux règles de l'algorithme.

$$Leave \ :: \ S_p = Dominant \wedge (\exists q \in \mathcal{N}_p, S_q = Dominant \wedge id_q < id_p) \mapsto S_p \leftarrow dominé$$

$$Join \ :: \ S_p = dominé \wedge (\forall q \in \mathcal{N}_p, S_q = dominé \vee id_q > id_p) \mapsto S_p \leftarrow Dominant$$

Question 5. Les actions *Leave* et *Join* d'un processus donné sont elles mutuellement exclusives ? Justifiez.

Oui, car l'opérande gauche dans la conjonction de la garde de *Join* est vraie si et seulement si l'opérande gauche dans la conjonction la garde de *Leave* est fausse. (vrai aussi pour les opérandes droites).

2.2.3 Preuve de l'algorithme

La preuve de correction d'un algorithme silencieux en supposant un démon distribué inéquitable comporte deux étapes majeures :

1. D'abord, montrer que dans toute configuration terminale, les variables définissent une solution du problème considéré (ici les variables doivent définir un ensemble indépendant maximal).
2. Puis, montrer que toute exécution termine, c'est-à-dire, atteint une configuration terminale, en un nombre fini d'étapes de calcul.

Question 6. Montrez, par contradiction, que dans toute configuration terminale, I est un ensemble indépendant.

Supposons que I ne soit pas un ensemble indépendant. Alors, il existe $p, q \in I$ tels que $p \neq q$, p est voisin de q et $S_p = S_q = \mathit{Dominant}$ dans γ_t. Supposons sans perte de généralité que $id_p < id_q$. Alors, l'action *Leave* est activable à q dans γ_t, contradiction.

Question 7. Montrez, par contradiction, que dans toute configuration terminale, I est un ensemble indépendant maximal.

Supposons que I soit un ensemble indépendant mais non maximal. Alors il existe $p \in V \setminus I$ tel que $I \cup \{p\}$ est un ensemble indépendant. Puisque $p \in V \setminus I$, $S_p = \mathit{dominé}$ dans γ_t. De plus, puisque $I \cup \{p\}$ est un ensemble indépendant, on a $\forall q \in \mathcal{N}_p$, $q \notin I$, *i.e.*, $S_q = \mathit{dominé}$ dans γ_t. Par suite, l'action *Join* est activable à p dans γ_t, contradiction.

D'où

Théorème 1. *Dans toute configuration terminale, I est un ensemble indépendant maximal.*

Nous allons maintenant démontrer que toute exécution de notre algorithme termine. Pour cela, nous allons utiliser la notion de *RANG*.

Soit $RANG(p) = |\{q \in V, id_q \leq id_p\}|$.

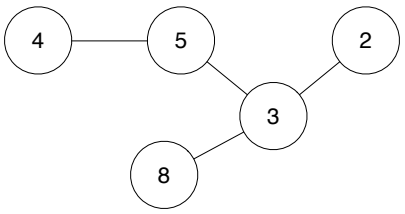

FIGURE 2 – Réseau identifié quelconque. (les identités sont les numéros à l'intérieur des nœuds.

Question 8. Dans la figure 2, qui est le processus de rang 1 et quel est le rang du processus d'identité 5 ? Expliquer informellement la notion de rang. Pourquoi est-ce un bonne mesure pour faire une récurrence.

Le processus de rang 1 a l'identité 2. Le processus d'identité 5 a le rang 4. $RANG(p)$ = position de p dans l'ordre des identités. Utile pour les récurrences car commence à 1 et pas de trous.

Question 9. Dans le pire des cas, combien d'actions exécute le processus de rang 1 ?

Au plus une

Question 10. Prouvez par récurrence sur le rang, que tout processus ne peut changer d'état qu'un nombre fini de fois.

Cas de base : Soit p le processus tel que $RANG(p) = 1$. Par définition, $\forall q \in \mathcal{N}_p$, $id_p < id_q$. Ainsi, si $S_p = Dominant$, alors p n'est pas activable. Sinon, p est activable pour changer d'état et après ce changement d'état il est inactivable pour toujours. Par suite, la récurrence est vérifiée dans ce cas.

Hypothèse de récurrence : Supposons que tout processus p tel que $RANG(p) \leq i$ ne peut changer d'état qu'un nombre fini de fois.

Pas de récurrence : Soit q un processus tel que $RANG(q) = i + 1$. Par hypothèse de récurrence, le système atteint nécessairement une configuration γ où $\forall q' \in \mathcal{N}_q$, si $id_{q'} < id_q$, alors q' ne change plus d'état. Ainsi, deux cas sont possibles dans γ :

- Supposons $\exists q' \in \mathcal{N}_q, S_{q'} = Dominant \wedge id_{q'} < id_q$. Dans ce cas, ce prédicat est vrai pour toujours par hypothèse de récurrence. Après au plus un changement d'état de q, on a S_q est égale à *dominé* et q est inactivable pour toujours. Ainsi, à partir de γ, q change d'état au plus une fois.
- Sinon, $\forall q' \in \mathcal{N}_q, id_{q'} < id_q \Rightarrow S_{q'} = dominé$. Dans ce cas, ce prédicat est vrai pour toujours par hypothèse de récurrence. Après au plus un changement d'état de q, on a S_q est égale à *Dominant* et q est inactivable pour toujours. Ainsi, à partir de γ, q change d'état au plus une fois.

Ainsi, q ne peut changer d'état qu'un nombre fini de fois.

Puisque tout processus change d'état qu'un nombre fini de fois, le nombre d'étapes de calcul est lui aussi fini et le théorème est vérifié.

D'après la question précédente, on a :

Théorème 2. *À partir d'une configuration quelconque, le système atteint une configuration terminale en un nombre fini d'étapes de calcul.*

Question 11. Proposez une exécution possible comportant $\Omega(n^2)$ étapes de calcul.

Borne inf sur le nombre d'étape de calcul : $\frac{n(n+1)}{2}$. Il suffit de considérer un réseau en « chaîne » où tous les nœuds sont initialement dans l'état *dominé* et où les nœuds sont identifiés de 1 à n de la gauche vers la droite. En activant les nœuds en priorité de la droite vers la gauche, on obtient la borne.

Question 12. Quel est le temps de stabilisation en rondes de l'algorithme ? Donnez un exemple d'exécution réalisant ce temps. Prouvez par récurrence sur le rang cette borne.

Borne exacte avec le réseau en chaîne précédent et avec les nœuds d'identités paire comme *Dominant*.

Théorème 3. *À partir d'une configuration quelconque, le système atteint une configuration terminale en au plus n rondes.*

Preuve. Nous montrons ce théorème par récurrence sur le rang.

Cas de base : Soit p le processus tel que $RANG(p) = 1$. Par définition, $\forall q \in \mathcal{N}_p$, $id_p < id_q$. Ainsi, si $S_p = Dominant$, alors p n'est pas activable. Sinon, p est activable pour changer d'état et après au plus une ronde, p exécute l'action *Join*. Par suite, la récurrence est vérifiée dans ce cas.

Hypothèse de récurrence : Supposons que tout processus p tel que $RANG(p) \leq i$ est inactivable pour toujours après au plus $RANG(p)$ rondes.

Pas de récurrence : Soit q un processus tel que $RANG(q) = i + 1$. Par hypothèse de récurrence, après au plus i rondes, $\forall q' \in \mathcal{N}_q$, si $id_{q'} < id_q$, alors q' est inactivable pour toujours, et deux cas sont possibles :

- Supposons $\exists q' \in \mathcal{N}_q, S_{q'} = Dominant \wedge id_{q'} < id_q$. Dans ce cas, ce prédicat est vrai pour toujours par hypothèse de récurrence, et une fois que S_q est égale à *dominé*, q est inactivable pour toujours. D'après la règle *Leave*, $S_q = dominé$ en au plus une ronde.
- Sinon, $\forall q' \in \mathcal{N}_q, id_{q'} < id_q \Rightarrow S_{q'} = dominé$. Dans ce cas, ce prédicat est vrai pour toujours par hypothèse de récurrence, et une fois que S_q est égale à *Dominant*, q est inactivable pour toujours. D'après la règle *Join*, $S_q = Dominant$ en au plus une ronde.

Ainsi, q est inactivable pour toujours en au plus $i + 1$ rondes et le théorème est vérifié.

□