Algorithme Auto-Stabilisant Compact d'Election pour les Graphes Arbitraires

Lélia Blin, Sébastien Tixeuil

Algorithme Auto-Stabilisant Compact d'Election pour les Graphes Arbitraires

Lélia Blin[1,2] et Sébastien Tixeuil [2]

[1] *Université d'Evry-Val-d'Essonne.*
[2] *Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606*

Nous présentons le premier algorithme auto-stabilisant d'élection pour les réseaux de topologie arbitraire dont la complexité en espace est de $O(\max\{\log\Delta, \log\log n\})$ bits par nœud, où n est la taille du réseau et Δ son degré. Cette complexité en espace est sous-logarithmique en n, tant que $\Delta = n^{o(1)}$.

Mots-clefs : Auto-stabilisation, Election, Mémoire compacte.

1 Introduction

Ce papier aborde le problème de la conception d'algorithmes auto-stabilisants efficaces en mémoire pour le problème de l'élection. *L'auto-stabilisation* [Dol00] est un paradigme général permettant de concevoir des algorithmes distribués tolérants aux fautes transitoires. De manière intuitive, un algorithme est auto-stabilisant si il est capable de retrouver un fonctionnement normal après l'apparition de fautes transitoires, et ceci sans intervention extérieure. *L'élection* est un problème fondamental de l'algorithmique distribuée, et consiste à distinguer de tous les autres un unique nœud du réseau, capable par la suite d'exécuter des actions spécifiques. L'élection est spécialement importante dans le contexte de l'auto-stabilisation puisque de nombreux protocoles supposent qu'un nœud distingué existe dans le réseau, même après l'apparition de fautes. Un mécanisme d'élection auto-stabilisant permet à de tels protocoles d'être exécutés dans des réseaux où aucun nœud n'est distingué a priori. *L'efficacité en mémoire* est relative à la quantité d'informations qui est envoyée aux voisins pour permettre la stabilisation. Utiliser un espace mémoire réduit induit une quantité d'informations envoyée de plus petite taille, ce qui *(1)* réduit l'échange d'informations quand il n'y a pas de fautes, ou après stabilisation [ANT12], et *(2)* facilite l'utilisation conjointe de l'auto-stabilisation et de la redondance [HP00].

Un résultat fondamental concernant la complexité en espace dans le contexte d'algorithmes auto-stabilisants silencieux † établi par Dolev et al. [DGS99], montre que, dans un réseau à n nœuds, $\Omega(\log n)$ bits de mémoire par nœud sont nécessaires pour résoudre des tâches telles que l'élection. En conséquence, seuls des algorithmes *bavards* peuvent atteindre une complexité en espace de $o(\log n)$ bits pour ce problème. Plusieurs tentatives pour concevoir des algorithmes compacts (en mémoire) auto-stabilisants pour l'élection (*i.e.*, des algorithmes dont la complexité en espace est $o(\log n)$ bits par nœud) ont été présentées sur des topologies en anneau. Les algorithmes de Mayer *et al.* [MOOY92], de Itkis et Levin [IL94], et de Awerbuch et Ostrovsky [AO94] utilisent un nombre constant de bits par nœud, mais guarantissent seulement une stabilisation *probabiliste* (au sens Las Vegas). Des algorithmes *déterministes* pour les anneaux ont été d'abord proposés par Itkis *et al.* [ILS95] pour les anneaux de taille première. Beauquier *et al.* [BGJ99] considèrent les anneaux de taille arbitraire, mais supposent que les identifiants des nœuds dans les anneaux de taille n sont bornés supérieurement par $n+k$, où k est une petite constante. Un résultat récent de Blin *et al.* [BT17] montre que les deux contraintes précédentes dans le cadre déterministe ne sont pas necéssaires, en présentant un algorithme d'élection déterministe pour des anneaux de taille arbitraire où les identifiants sont bornés par un polynôme en n, avec un espace en mémoire de $O(\log\log n)$ bits.

†. Un algorithme est *silencieux* si chacune de ses executions converge vers une configuration après laquelle l'état des nœuds ne change pas. Un algorithme non-silencieux est dit *bavard* (cf. [BT17]).

Dans les réseaux dont la topologie est arbitraire, l'élection auto-stabilisante est étroitement liée à la construction d'arbre auto-stabilisante. D'un coté, l'existence d'un nœud distingué permet la conception d'algorithmes efficaces en temps et en espace pour construire un arbre couvrant [Dol00]. D'un autre coté, faire pousser et unifier des arbres constitue la technique principale pour l'élection dans un réseau général, car le nœud distingué est simplement la racine de l'arbre ainsi construit [Dol00]. A notre connaissance, tous les algorithmes qui ne supposent *pas* la pré-existence d'un nœud distingué utilisent $\Omega(\log n)$ bits par nœud. Cette complexité en mémoire élevée est due à l'implémentation de deux techniques principales, utilisées par tous les algorithmes, et rapellées ci-après.

La première technique est celle qui consiste à utiliser des variables de type "pointeurs-vers-le-voisin", qui permet de désigner de manière univoque un voisin particulier d'un nœud. Dans la perspective de la construction d'un arbre, de telles variables sont utilisées pour désigner le nœud parent dans l'arbre construit. De manière spécifique, le parent de chaque nœud est désigné sans ambigüité par son identifiant, ce qui requiert $\Omega(\log n)$ bits pour chacune de ces variables. En principe, il serait possible de réduire la mémoire utilisée à $O(\log \Delta)$ bits par variable pointeur dans les réseaux de degré maximum Δ, en utilisant un coloriage des nœuds à distance 2 à la place des identifiants globaux pour désigner les voisins. Cependant, ceci nécessiterait l'existence d'un algorithme de coloriage à distance 2 auto-stabilisant qui utilise lui-même $o(\log n)$ bits par nœud. Or, les algorithmes auto-stabilisants existants pour le coloriage à distance 2 utilisent des variables de grande taille. Par exemple, l'algorithme de Herman *et al.* [HT04] demande à chaque nœud de communiquer son voisinage à distance 3 à tous ses voisins, induisant une complexité en mémoire de $O(\Delta^3 \log n)$ bits par nœud. Johnen *et al.* [GJ01] effectuent un tirage aléatoire des couleurs dans l'intervalle $\{0,...,n^2\}$, ce qui induit une complexité en espace de $O(\log n)$ bits par nœud. Enfin, même si l'algorithme déterministe de Blair *et al.* [BM12] réduit la complexité en espace à $O(\log \Delta)$ bits par nœud, cette performance est atteinte en ignorant le coût pour stocker une variable "pointeur-vers-le-voisin" par nœud. En l'absence de coloriage à distance 2 (ce que leur algorithme [BM12] est justement censé produire), leur implémentation requiert toujours $\Omega(\log n)$ bits par nœud. A ce jour, aucun algorithme auto-stabilisant ne permet d'implémenter des variables "pointeur-vers-le-voisin" avec une complexité en espace $o(\log n)$ bits dans les réseaux de topologie arbitraire.

La deuxième technique fondamentale pour la construction d'arbre (ou l'élection) est l'utilisation d'une variable "distance" qui permet de stocker la distance de chaque nœud au nœud distingué. Une telle variable est utilisée dans le cadre de la construction d'arbre auto-stabilisante pour casser les cycles qui pourraient résulter d'une configuration initiale arbitraire (cf. [Dol00]). Clairement, stocker la distance au nœud distingué dans un réseau à n-nœuds requiert $\Omega(\log n)$ bits par nœud. Il existe quelques algorithmes auto-stabilisants d'élection qui n'utilisent pas explicitement de variables "distance" [JGJDL02, DDT06]), mais leur complexité en espace est $O(n \log n)$ bits [DDT06] ou $O(\log n + \Delta)$ [JGJDL02]. L'utilisation du principe des variables "distance" avec une complexité en espace en dessous de $\Theta(\log n)$ bits a été proposé par Awerbuch *et al.* [AO94], et par Blin *et al.* [BT17]. Ces travaux distribuent l'information des distances jusqu'au nœud distingué parmi les nœuds selon plusieurs mécanismes, permettant de stocker seulement $o(\log n)$ bits par nœud. Cependant, ces mécanismes sophistiqués ont seulement été proposés pour des topologies en anneau. A ce jour, aucun algorithme auto-stabilisant ne permet d'implémenter des variables "distance" avec une complexité en espace en $o(\log n)$ bits par nœud dans les réseaux de topologie arbitraire.

2 Election Compacte et Auto-stabilisante

Nous présentons un nouvel algorithme auto-stabilisant pour l'élection dont la complexité en espace est $O(\max\{\log \Delta, \log\log n\})$ bits par nœud dans les réseaux de topologie arbitraire de n nœuds avec un degré maximum Δ. Cet algorithme est la première solution auto-stabilisante pour ce problème dans les réseaux arbitraires dont la complexité en mémoire est en $o(\log n)$ (dès que $\Delta = n^{o(1)}$). L'algorithme est conçu pour le modèle standard d'exécution (le modèle à états) et s'exécute avec l'ordonnanceur le plus général, distribué et inéquitable (l'ordonnanceur peut sélectionner à chaque étape un sous-ensemble arbitraire de nœuds activables). La conception de notre algorithme requiert de surmonter plusieurs verrous, y compris la manipulation de "pointeurs-vers-le-voisin" et de variables "distance" en utilisant $o(\log n)$ bits par nœud dans des réseaux arbitraires. Ces déblocages sont dus à de nouvelles sous-routines, dont chacune mérite qu'on s'y intéresse indépendamment, décrites ci-après.

Notre algorithme auto-stabilisant d'élection est basé sur une construction d'arbre enraciné sur un nœud de degré maximum, sans utiliser de variables "distance". Si plusieurs nœuds de degré maximum existent, les candidats sont sélectionnés suivant leur couleur à distance 2, et si nécessaire, par leur identifiant global.

Theorem 1. *L'algorithme* **C-LE** *résout le problème de l'élection de manière auto-stabilisante et bavarde dans tout réseau de n nœuds, en utilisant le modèle à états et un ordonnanceur distribué inéquitable, avec* $O(\max\{\log\Delta, \log\log n\})$ *bits de mémoire par nœud, où* Δ *dénote le degré maximum du graphe.*

Notre algorithme auto-stabilisant bavard réutilise et étend aux graphes arbitraires une technique initialement présentée pour les anneaux[BT17] afin d'obtenir des identifiants compacts de taille $O(\log\log n)$ bits par nœud, en publiant les identifiants des nœuds bit par bit, en partant du bit de poids fort. Par exemple, pour l'identifiant 10 dont la représentation binaire est 1010, le nœud signalera que la taille de son identifiant est 4, donc que son premier bit à 1 est à la position 4, et son deuxième bit à la position 2. Nous appelons les étapes successives de publications des bits des *phases*. Globalement, le processus d'élection consiste à exécuter plusieurs couches algorithmiques de priorités décroissantes :

Un algorithme auto-stabilisant silencieux pour le coloriage à distance 2 qui permet d'implémenter des variables "pointeur-vers-le-voisin" avec $o(\log n)$ bits par nœud. Contrairement aux algorithmes auto-stabilisants précédents de coloriage à distance 2 nous n'utilisons pas les identifiants globaux pour encoder les "pointeurs-vers-le-voisin", mais nous utilisons à la place notre représentation compacte des identifiants pour briser les symétries. Ceci rend possible la conception d'un encodage compact pour l'arbre couvrant. Les nœuds en conflit à distance un échangent leurs identifiants bit par bit, et le nœud dont l'identifiant est le plus petit incrémente sa couleur de un. Une variable est dédiée à la diffusion de Δ, le degré du graphe, afin de contraindre la couleur choisie à un intervalle $[1, \Delta(v)^2 + 1]$. Lorsqu'un un nœud dépasse cet intervalle en incrémentant sa couleur, il met sa couleur à un. Un nœud v qui n'a pas de conflit à distance deux, mais qui détecte un conflit à distance 2 entre deux de ses voisins u_1 et u_2, devient un intermédiaire et relaie l'identifiant maximum entre u_1 et u_2, afin que le voisin d'identifiant minimum change sa couleur.

Un algorithme auto-stabilisant silencieux pour éliminer les cycles et les arbres illégitimes repris de travaux précédents [BT17].

Un algorithme auto-stabilisant silencieux pour la détection des cycles qui n'utilise *pas* de variables "distance" jusqu'à la racine de l'arbre. A la place, nous détectons un cycle en nous basant sur l'unicité de chaque identifiant dans le réseau. De manière remarquable, cette technique peut être implémentée de manière silencieuse et auto-stabilisante en utilisant une mémoire en $O(\max\{\log\Delta, \log\log n\})$ bits par nœud. Plus précisément, nous décrivons en premier le principe en utilisant les identifiants globaux ($O(\log n)$ bits) avant d'expliquer son adaptation compacte ($O(\log\log n)$ bits). Nous utilisons une variable m_v dédiée à la collecte de l'identifiant minimum, en partant des feuilles vers la racine. Si un nœud reçoit son propre identifiant où que deux de ses enfants proposent le même identifiant minimum, c'est qu'il existe un cycle et une erreur est alors détectée. Afin de contourner le problème posé par la présence dans m_v d'un identifiant minimum inexistant dans le graphe, un nœud qui partage le même identifiant minimum avec un de ses enfants et son parent, ré-initialise sa variable m_v avec son propre identifiant.

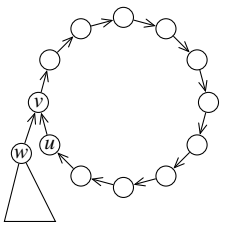

FIGURE 1: Structure couvrante

Avec des identifiants compacts, nous diffusons les identifiants bit par bit, au fur et à mesure de la diffusion, les nœuds d'identifiants (localement) maximum deviennent passifs, les autres nœuds restent actifs. Seuls les nœuds actifs peuvent incrémenter leur phase. De plus les nœuds actifs incrémentent leur phase si leur parent et au moins un de leurs enfants ont la même information qu'eux. Seuls les nœuds passifs redémarrent le processus en utilisant leur identifiant. Le système converge vers une configuration où seul le nœud d'identifiant minimum reste actif. Considérons un nœud v avec deux enfants, l'enfant u qui appartient à un cycle et l'enfant w qui n'est pas dans un cycle (comme les nœuds ont un seul parent, ils ne peuvent être impliqués que dans un seul cycle). Si le nœud d'identifiant minimum (disons v) est dans le cycle, à la dernière phase de la diffusion bit par bit, le nœud u lui annonce la même position que lui et v détecte le cycle. Si le nœud d'identifiant minimum est dans le sous-arbre de w où w lui

même, le nœud v détecte une erreur car l'algorithme converge vers une configuration où u et w annoncent la même position à la dernière phase.

Un algorithme auto-stabilisant bavard pour la construction d'un arbre couvrant , qui n'utilise pas non plus de variables "distance". En particulier, cet algorithme utilise une technique nouvelle qui évite la création de nouveaux cycles, sans utiliser de "distances", et crée ainsi une forêt couvrante, qui se réduit ultimement à un arbre enraciné au un nœud de degré maximum. Notre implémentation permet une complexité mémoire en $O(\max\{\log\Delta, \log\log n\})$ bits par nœud. Cet algorithme de construction d'arbre est trivialement modifié pour obtenir un algorithme d'élection. Plus précisément, l'algorithme construit un arbre couvrant enraciné au nœud de degré maximum, de couleur maximum, et d'identifiant maximum (dans cet ordre de priorité). Dans chaque configuration, un invariant est maintenu : tout nœud v qui a un parent doit avoir son voisin u soit comme parent soit comme enfant, où u est le voisin de v dont le degré est maximum, la couleur maximum, l'identifiant maximum. Tout nœud sans parent (racine) est un leader potentiel qui diffuse (bit par bit) son identifiant de la racine vers les feuilles. Un nœud v dans un sous-arbre T_v qui a un voisin u dans un sous-arbre T_u annonçant une meilleure racine (au sens $<$ degré, couleur, identifiant $>$) en informe la racine r_v de T_v. r_v choisit alors de fusionner avec le sous-arbre de meilleure racine en réorientant son arbre vers le nœud v qui fusionne avec le sous arbre T_u, et v choisit u comme parent. L'opération est répétée jusqu'à ce qu'un seul arbre couvrant persiste. La racine de cet arbre unique devenant l'unique leader. Suite à une configuration arbitraire initiale, il se peut que cette configuration soit un arbre couvrant enraciné en une mauvaise racine. La diffusion continue de l'identifiant bit par bit permet de détecter une telle erreur, mais rend la solution bavarde.

En raison du manque de place, seuls les grands principes de l'algorithme sont présentés ici. Les détails et les preuves associées peuvent être trouvées dans le rapport associé (ref. arXiv :1702.07605).

Références

[ANT12] J. Adamek, M. Nesterenko, and S. Tixeuil. Using abstract simulation for performance evaluation of stabilizing algorithms : The case of propagation of information with feedback. In *SSS 2012*, LNCS. Springer, 2012.

[AO94] B. Awerbuch and R. Ostrovsky. Memory-efficient and self-stabilizing network reset. In *PODC*, pages 254–263. ACM, 1994.

[BGJ99] J. Beauquier, M. Gradinariu, and C. Johnen. Memory space requirements for self-stabilizing leader election protocols. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC 1999)*, pages 199–208, 1999.

[BM12] J. R. S. Blair and F. Manne. An efficient self-stabilizing distance-2 coloring algorithm. *Theor. Comput. Sci.*, 444 :28–39, 2012.

[BT17] L. Blin and S. Tixeuil. Compact deterministic self-stabilizing leader election on a ring : The exponential advantage of being talkative. *Distributed Computing*, page To be appear, 2017.

[DDT06] S. Delaët, B. Ducourthial, and S. Tixeuil. Self-stabilization with r-operators revisited. *Journal of Aerospace Computing, Information, and Communication (JACIC)*, 3(10) :498–514, 2006.

[DGS99] S. Dolev, M. G. Gouda, and M. Schneider. Memory requirements for silent stabilization. *Acta Inf.*, 36(6) :447–462, 1999.

[Dol00] S. Dolev. *Self-stabilization*. MIT Press, March 2000.

[GJ01] M. Gradinariu and C. Johnen. Self-stabilizing neighborhood unique naming under unfair scheduler. In Rizos Sakellariou, John Keane, John R. Gurd, and Len Freeman, editors, *Euro-Par 2001 : Parallel Processing, 7th International Euro-Par Conference Manchester, UK August 28-31, 2001, Proceedings*, volume 2150 of *Lecture Notes in Computer Science*, pages 458–465. Springer, 2001.

[HP00] T. Herman and S. V. Pemmaraju. Error-detecting codes and fault-containing self-stabilization. *Inf. Process. Lett.*, 73(1-2) :41–46, 2000.

[HT04] T. Herman and S. Tixeuil. A distributed tdma slot assignment algorithm for wireless sensor networks. In *Proceedings of the First Workshop on Algorithmic Aspects of Wireless Sensor Networks (AlgoSensors'2004)*, number 3121 in Lecture Notes in Computer Science, pages 45–58, Turku, Finland, July 2004. Springer-Verlag.

[IL94] G. Itkis and L. A. Levin. Fast and lean self-stabilizing asynchronous protocols. In *FOCS*, pages 226–239. IEEE Computer Society, 1994.

[ILS95] G. Itkis, C. Lin, and J. Simon. Deterministic, constant space, self-stabilizing leader election on uniform rings. In *WDAG*, LNCS, pages 288–302. Springer, 1995.

[JGJDL02] J.Beauquier, M. Gradinariu, C. Johnen, and J. O. Durand-Lose. Token-based self-stabilizing uniform algorithms. *J. Parallel Distrib. Comput.*, 62(5) :899–921, 2002.

[MOOY92] A. J. Mayer, Y. Ofek, R.l Ostrovsky, and M. Yung. Self-stabilizing symmetry breaking in constant-space (extended abstract). In *STOC*, pages 667–678, 1992.