Le pouvoir séparateur d'une pièce de monnaie

Quentin Bramas, Sébastien Tixeuil

Le pouvoir séparateur d'une pièce de monnaie †

Quentin Bramas[1] et Sébastien Tixeuil[1,2]

[1] *Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, F-75005, Paris, France*
[2] *Institut Universitaire de France*

Nous considérons le problème qui consiste à éparpiller n robots mobiles dans un plan Euclidien, à partir d'une situation initiale quelconque où en particulier, deux robots peuvent occuper la même position. Comme ce problème n'admet pas de solution déterministe (deux robots identiques initialement à la même place ont un comportement identique et donc ne se séparent jamais), les solutions sont nécessairement probabilistes. Nous étudions le nombre de lancers de pièce de monnaie (*alias* le nombre de bits aléatoires) nécessaire à une séparation totale des robots. Nous montrons tout d'abord que $n \log n$ lancers sont nécessaires pour éparpiller n robots. Puis, nous donnons une condition nécessaire et suffisante pour qu'un algorithme soit optimal en nombre de lancers. Comme il s'avère que les algorithmes existants vérifient cette condition, ils sont optimaux en nombre de lancers pour le problème de l'éparpillement. Ensuite nous étudions la complexité en temps des algorithmes d'éparpillement, lorsque la détection forte de la multiplicité (*i.e.* la capacité à compter le nombre de robots sur une position précise) n'est pas disponible. La séparation en temps constant étant impossible, nous présentons une famille d'algorithmes qui éparpille n robots en $O(f(n))$ pour toute fonction f non bornée supérieurement et, parmi cette famille, nous proposons un algorithme qui est à la fois optimal en temps et en nombre de lancers. Le détail des résultats peut être trouvé dans le rapport technique associé [BT13].

Keywords: Algorithmes probabilistes, Complexité, Réseaux de Robots, Éparpillement

1 Introduction

Nous considérons un système distribué constitué de robots autonomes qui peuvent se déplacer dans un plan Euclidien, s'observer et effectuer des calculs. Les robots ne communiquent pas explicitement entre eux et sont totalement autonomes. De tels réseaux peuvent être déployés dans des environnements inaccessibles aux humains pour effectuer des taches collaboratives comme les opérations de recherche et de sauvetage, de collecte de données, de surveillance environnementale et même d'exploration extra-terrestre. Le problème de l'éparpillement (ou séparation) de robots consiste, à partir d'une configuration initiale quelconque où en particulier, deux robots peuvent occuper la même position, à faire en sorte qu'au terme de l'exécution, tous les robots occupent des positions distinctes.

Modèle. Nous modélisons n robots comme autant de points dans un plan Euclidien. Nous appelons *configuration* le multi-ensemble contenant les positions des robots au temps t et leur multiplicité (lorsque plusieurs robots partagent la même position). Le temps est discrétisé et, à chaque instant, un sous-ensemble de robots s'activent et effectuent de manière atomique un cycle *Regarde-Calcule-Se Déplace*. A chaque cycle un robot activé *Regarde* autour de lui et obtient un *instantané* des positions des autres robots. Basé uniquement sur cette information visuelle, le robot *Calcule* une destination, et *se Déplace* vers cette destination. Comme tous les robots sont identiques, ils exécutent le même algorithme.

Lorsqu'un robot regarde son environnement, il obtient un instantané du plan, et distingue un point vide d'un point occupé par un ou plusieurs robots. Les robots sont capables de *détecter la multiplicité* s'il peuvent distinguer un point occupé par un robot d'un point occupé par plus d'un robot. Cette détection est dite *forte* si les robots sont capables de détecter le nombre exact de robots qui occupent un point, sinon elle est *faible*. Il faut remarquer que sans détection forte de la multiplicité les robots peuvent ne pas connaître le nombre total de robots présents dans leur environnement. De plus, les robots n'ont aucune mémoire persistante (la mémoire est réinitialisée au début de chaque cycle), donc l'instantané obtenu est la seule donnée utilisable par l'algorithme exécuté sur chaque robot.

† Ce travail est soutenu en partie par le LINCS, Paris, France

On suppose que le sous-ensemble des robots activés ainsi que leur orientation (*i.e.* leur système de coordonnées egocentré) sont choisis par un *adversaire* à chaque cycle. Dans le cas où l'algorithme exécuté est déterministe, l'adversaire peut décider d'activer l'ensemble des robots à chaque cycle, avec le même système de coordonnées. Il est alors impossible de séparer deux robots initialement présents à la même position. De fait, toutes les solutions précédentes au problème que nous considéront utilisent une approche probabiliste.

Travaux connexes. Le premier algorithme d'éparpillement probabiliste de robots à été donné par Dieudonné et Petit [DP09]. Cet algorithme consiste, pour tous les robots qui partagent la même position, à générer deux points qui ne peuvent pas être choisis par des robots se trouvant à une autre position, et à choisir l'une des deux destinations en lançant une pièce de monnaie. Les robots se divisent alors (en moyenne) en deux groupes de proportion égale, localisés sur deux positions distinctes. L'algorithme se poursuit jusqu'à ce que chaque groupe ne contienne qu'un seul robot. Clément *et al.* [CDPB$^+$10] ont montré que cet algorithme éparpille n robots en $O(\log(n)\log\log(n))$ rondes en moyenne (une ronde étant la plus petite portion d'exécution pendant laquelle tout robot à l'opportunité d'exécuter au moins un cycle). Dans le même article, ils donnent un autre algorithme d'éparpillement de robots qui sépare n robots en moyenne en temps constant. Pour fonctionner, cet algorithme fait l'hypothèse que le nombre total de robots est connu, ce qui revient à supposer que les robots sont capables de détection forte de la multiplicité. Le principe de l'algorithme est le même que [DP09], mais au lieu de choisir une destination aléatoirement parmi un ensemble de deux points, la destination est choisie aléatoirement parmi un ensemble de $2n^2$ points (où n est le nombre total de robots).

Contributions. Dans cet article nous étudions le nombre de lancers de pièce de monnaie (ou de bits aléatoires) nécessaires à l'éparpillage de n robots, *i.e.* la somme des lancers utilisés par l'ensemble des robots pour s'éparpiller. Nous donnons une condition nécessaire et suffisante pour qu'un algorithme utilise en moyenne le nombre minimum de lancers. Ensuite, nous étudions la complexité en temps des algorithmes d'éparpillement qui ne disposent *pas* de la détection forte de la multiplicité. La séparation en temps constant étant impossible, nous proposons une famille d'algorithmes qui éparpille en moyenne n robots en $O(f(n))$ pour toute fonction f croissante et non bornée supérieurement et, parmi cette famille, nous présentons un algorithme qui est à la fois optimal en temps et en nombre de lancers.

Algorithme d'éparpillement de robots canonique. En observant les algorithmes existants, nous proposons un algorithme d'éparpillement canonique. Nous remarquons que le nombre de destinations possibles caractérise un algorithme d'éparpillement, et influence directement le nombre de lancers utilisés par les robots.

Définition 1.1 *$\mathcal{A}$ est un algorithme d'éparpillement de robots* canonique *s'il est de la forme :*

Algorithme 1 : Algorithme d'éparpillement de robots, exécuté par un robot r

1 *$C \leftarrow$ Configuration observée ; $P \leftarrow$ Position observée de r.*
2 *Générer un ensemble Pos de $k_{\mathcal{A}}(C,P)$ destinations possibles, tel que chaque point dans Pos ne peut pas être choisi par un robot qui n'est pas localisé au point P.*
3 *Se déplacer vers un point de Pos choisi aléatoirement de manière uniforme.*

La fonction $k_{\mathcal{A}}$ qui donne le nombre de destinations possibles en fonction de la configuration et d'une position est appelée fonction de destination de l'algorithme $\mathcal{A}$.

La ligne 2 de l'algorithme 1 implique qu'un algorithme d'éparpillement canonique doit s'assurer que la cardinalité des points de multiplicité n'augmente jamais, c'est à dire que deux robots qui se trouvent à positions distinctes au temps t resteront à positions distinctes au temps $t' > t$. Les algorithmes précédents sont canoniques, et utilisent les diagrammes de Voronoï pour s'assurer que la cardinalité des points de multiplicité est décroissante.

- L'algorithme donné dans [DP09] est canonique avec la ligne 2 remplacée par : *Générer un ensemble Pos de 2 points distincts arbitraires dans la cellule de r dans le diagramme de Voronoï.*
- L'algorithme donné dans [CDPB$^+$10] est canonique avec la ligne 2 remplacée par : *Générer un ensemble Pos de $2|C|^2$ points distincts arbitraires dans la cellule de r dans le diagramme de Voronoï.*

2 Caractérisation de l'optimalité en nombre de lancers de pièce

Nombre minimum de lancers de pièce nécessaires à l'éparpillement de n robots. Pour séparer deux robots, un algorithme doit faire en sorte que leur destination soit différente. Comme l'algorithme, commun aux deux robots, prend en entrée le même instantané, il doit nécessairement utiliser un lancer de pièce pour avoir une chance de les séparer. Dans le meilleur des cas, un lancer de pièce par robot (donc deux lancers au total) suffira. Cette observation est la base de notre raisonnement par récurrence : $2\log_2(2)$ lancers sont nécessaires pour séparer deux robots.

Soit $k \in \mathbb{N}$, nous savons qu'il faut au moins $\log_2(k)$ lancers pour affecter à un robot une destination aléatoire choisie dans un ensemble à k éléments (*i.e.,* choisir un nombre aléatoirement entre 1 et k). Si n robots partagent la même position, en faisant la somme sur chaque robot, $n\log_2(k)$ lancers sont nécessaires pour répartir aléatoirement n robots dans k destinations. Dans le meilleur des cas, les n robots seront divisés en k groupes distincts de n/k robots. Par induction, si nous supposons que chaque groupe de n/k robots nécessite $n/k\log_2(n/k)$ lancers, alors le nombre total de lancers utilisés pour séparer les n robots est bien : $n\log_2(k)+k\times n/k\log_2(n/k) = n\log_2 n$. Dans le cas ou n n'est pas divisible par k, la concavité de la fonction log permet d'obtenir le résultat de manière similaire. D'où le théorème suivant :

Théorème 2.1 *L'éparpillement de n robots nécessite au moins $n\log_2 n$ lancers.*

Condition nécessaire et suffisante d'optimalité. Si les robots exécutent un algorithme d'éparpillement canonique avec plusieurs destinations possibles (*i.e.*$|Pos| > 1$), alors ces derniers ont une probabilité positive de se séparer. Plus généralement, nous pouvons montrer de manière surprenante que si à chaque ronde le nombre de destinations possibles est polynomial en n, alors il suffit de $O(n\log n)$ lancers de pièce en moyenne pour éparpiller n robots, *i.e.* l'algorithme est optimal en nombre de lancers. D'où le théorème suivant :

Théorème 2.2 *Un algorithme canonique d'éparpillement de robots $\mathcal{A}$ est optimal en nombre de lancers (i.e., utilise $\Theta(n\log n)$ lancers en moyenne) si et seulement si $\exists K \in \mathbb{N}$ tel que $k_{\mathcal{A}}(C,P) = O(|C|^K)$, quelque soit le type de détection de la multiplicité utilisé.*

Comme les deux algorithmes existants vérifient la condition précédente, nous venons de prouver qu'ils sont optimaux en nombre de lancers, malgré des complexités en temps différentes ($O(\log(n)\log\log(n))$ pour le premier [DP09] et constant pour le deuxième [CDPB$^+$10]).

3 Complexité en temps sans détection forte de la multiplicité

Vitesse de séparation des algorithmes d'éparpillement sans détection forte de la multiplicité. Nous savons déjà que l'hypothèse de la détection forte de la multiplicité permet une complexité de $O(1)$ rondes en moyenne (voir l'algorithme de Clément *et al.* [CDPB$^+$10]). Le théorème suivant montre que cette borne n'est pas atteignable sans cette hypothèse :

Théorème 3.1 *Il n'existe pas d'algorithme sans détection forte de la multiplicité qui éparpille n robots en $O(1)$ rondes en moyenne.*

Malgré ce résultat négatif, nous présentons maintenant une famille d'algorithmes $(\mathrm{SA}_f)_{f\in\mathcal{F}}$ n'utilisant pas la détection de la multiplicité, où $\mathcal{F}$ est l'ensemble des fonctions de $\mathbb{N}$ dans $\mathbb{N}$ croissantes et surjectives.

Algorithme SA_f : Algorithme d'éparpillement sans détection de multiplicité exécuté par r.

1 Générer le diagramme de Voronoï de la configuration observée.
2 Soit u le nombre de points occupés par au moins un robot.
3 $Cell \leftarrow$ Cellule de Voronoï de r ; $\quad x \leftarrow f^{-1}(f(u)+1)$; $\quad k \leftarrow \max(2^{250}, 16x^4, u^3)$
4 Soit Pos un ensemble de k positions distinctes dans $Cell$.
5 Se déplacer vers un point de Pos choisi aléatoirement de manière uniforme.

Théorème 3.2 *Soit $f \in \mathcal{F}$. SA_f est un algorithme d'éparpillement canonique qui n'utilise pas la détection de la multiplicité, éparpille n robots en $O(f(n))$ rondes en moyenne et utilise $\Theta\left(n\log\left(f^{-1}\left(f(n)+1\right)\right)\right)$ lancers de pièce en moyenne.*

Chaque exécution de l'algorithme SA_f a pour but de diviser la multiplicité maximum des points occupés par plusieurs robots par un nombre x qui dépend de f. Pour cela, nous avons montré qu'il est suffisant de prendre $16x^4$ destinations différentes pour chaque groupe de robots, avec au minimum un certain nombre de destinations, borné de manière très large par 2^{250} et par u^3.

Si f n'est pas surjective, mais reste croissante et divergente, nous pouvons construire $g \in \mathcal{F}$ tel que $g(n) \in O(f(n))$. Ceci permet d'avoir SA_g qui éparpille n robots en $O(f(n))$ rondes en moyenne. Nous pouvons donc construire, pour chaque temps moyen de séparation souhaité f, un algorithme d'éparpillement. Prenons par exemple $f = \log^*$, l'algorithme $SA_{\log^*}$ éparpille n robots en $O(\log^*(n))$ rondes en moyenne. De plus, comme $\log(f^{-1}(f(n)+1)) = \log\left({}^{(\log^*(n)+1)}2\right) = {}^{\log^* n}2 = n$, (avec ${}^n a = \underbrace{a^{a^{\cdot^a}}}_{n}$) l'algorithme $SA_{\log^*}$ utilise $O(n^2)$ lancers (il n'est donc pas optimal comme celui de [CDPB+10], mais il éparpille beaucoup plus rapidement). Des algorithmes encore plus rapides peuvent être construits en prenant par exemple la fonction inverse d'Ackermann A^{-1}, $SA_{A^{-1}}$, avec une complexité en temps moyenne de $O(A^{-1}(n)) = o(log^* log^* log^* log^* n)$.

Algorithme optimal en temps et en nombre de lancers, sans détection de la multiplicité. Si nous voulons maintenant que notre algorithme SA_f soit optimal en nombre moyen de lancers, f doit satisfaire : $n\log(f^{-1}(f(n)+1)) = O(n\log(n))$. En prenant $f = \log\circ\log$, nous avons : $n\log(f^{-1}(f(n)+1)) = n\log 2^{2^{\log\log(n)+1}} = 2n\log(n) = O(n\log(n))$. Donc $SA_{\log\circ\log}$ est optimal en nombre de lancers de pièce et éparpille n robots en $O(\log\log n)$ rondes en moyenne. De plus, nous pouvons montrer qu'il n'existe pas d'algorithme optimal en nombre de lancers séparant n robots plus vite que $O(\log\log(n))$ rondes en moyenne, sans détection forte de la multiplicité. D'où le théorème suivant.

Théorème 3.3 *L'algorithme $SA_{\log\circ\log}$ est optimal en temps et en nombre de lancers.*

4 Conclusion

Nous nous sommes intéressés au nombre moyen de lancers de pièce qu'un algorithme distribué d'éparpillement de robots doit utiliser, et avons donné une condition nécessaire et suffisante pour qu'un algorithme soit optimal en nombre de lancers. Par ailleurs, l'hypothèse de la détection forte de la multiplicité s'avère nécessaire pour obtenir une séparation en temps moyen constant. Malgré ce résultat, nous avons donné une famille d'algorithmes qui permet d'obtenir un temps de séparation aussi proche que souhaité du temps constant, au prix d'une complexité en nombre de lancers qui n'est pas toujours optimale. Pour finir, nous avons trouvé dans cette famille, un algorithme optimal en nombre de lancers qui éparpille n robots en $O(\log\log n)$ rondes en moyenne. Cette vitesse de séparation est optimale et améliore d'un facteur $\log(n)$ le résultat précédent [DP09].

A ce jour et à notre connaissance, aucun algorithme d'éparpillement de robots n'a été trouvé dans le cas d'un modèle totalement asynchrone (où chaque action d'un cycle a une durée arbitraire). Nous pensons qu'un tel algorithme existe, avec des conditions supplémentaires, et qu'une grande partie de nos résultats pourrait se généraliser au cas asynchrone.

Références

[BT13] Quentin Bramas and Sébastien Tixeuil. The random bit complexity of mobile robots scattering. *CoRR*, abs/1309.6603, 2013.

[CDPB+10] Julien Clément, Xavier Défago, Maria Gradinariu Potop-Butucaru, Taisuke Izumi, and Stéphane Messika. The cost of probabilistic agreement in oblivious robot networks. *Inf. Process. Lett.*, 110(11) :431–438, 2010.

[DP09] Yoann Dieudonné and Franck Petit. Scatter of robots. *Parallel Processing Letters*, 19(1) :175–184, 2009.