Rendez-vous avec un lapin

Quentin Bramas, Anissa Lamani, Sébastien Tixeuil

Rendez-vous avec un lapin[†]

Quentin Bramas[1] et Anissa Lamani[1] et Sébastien Tixeuil[2]

[1] *Université de Strasbourg, ICUBE, CNRS, France*
[2] *Sorbonne Université, CNRS, LIP6, France*

Nous nous intéressons à une version du problème du rassemblement de robots mobiles où certains d'entre eux peuvent poser un lapin aux autres. Étant donné un ensemble de robots mobiles : si aucun des robots ne tombe en panne alors tous les robots doivent se rassembler à la même position, non connue à l'avance, en un temps fini. Si un, ou plusieurs robots tombe(nt) en panne à la même position (et posent donc un lapin aux autres), alors tous les autres robots encore en fonction doivent se rassembler à la même position que ceux qui sont en panne.

Motivés par les résultats d'impossibilité dans le cadre semi-synchrone, nous présentons dans ce papier, le premier algorithme résolvant le problème dans un environnement synchrone, en supposant le modèle vision-calcul-déplacement, sans hypothèses supplémentaires : les robots sont amnésiques, désorientés, ne peuvent pas détecter de multiplicité, et peuvent commencer leur exécution à n'importe quelle position (y compris celles avec des points de multiplicité).

Mots-clefs : Réseaux de robots, rassemblement, tolérance aux pannes

1 Introduction

Initiées par Suzuki et Yamashita [5], plusieurs recherches étudient les essaims de robots mobiles d'un point de vue algorithmique et s'intéressent à la puissance de calcul d'un ensemble de robots autonomes évoluant dans un espace euclidien bidimensionnel. L'objectif principal est de déterminer quelles sont les tâches (*p.e.*, formation de motifs, dispersion, rassemblement, exploration, patrouille, etc.) qui peuvent être effectuées par les robots dans un modèle donné. La simplicité des modèles proposés et la complexité des solutions obtenues a favorisé de nombreux résultats dans la communauté des systèmes distribués [4].

Le rassemblement constitue l'une des tâches fondamentales que les robots peuvent effectuer, *p.e.* pour collecter les données acquises ou pour mettre à jour l'algorithme des robots. Pour résoudre le problème du rassemblement (ou du rendez-vous quand seulement deux robots sont impliqués), les robots doivent rejoindre le même emplacement géographique, inconnu à l'avance, en un temps fini. Malgré la simplicité de sa définition, ce problème est notoirement impossible à résoudre sous certaines hypothèses.

Lorsque le nombre de robots augmente, certains robots peuvent tomber en panne, et deux variantes du rassemblement avec pannes ont été proposées. Dans la variante faible, seuls les robots corrects (toujours fonctionnels) doivent se rassembler, tandis que dans la variante forte, tous les robots (y compris ceux en panne) doivent se rassembler au même point. Bien sûr, la variante forte n'est possible que si tous les robots défectueux tombent en panne au même endroit. La variante faible du problème peut être résolue de manière déterministe même quand les robots ne sont pas synchronisés [2].

Dans cet article, nous considérons la version forte du problème de rassemblement qu'on appellera RaL (Rassemblement, ou Rendez-vous quand seuls deux robots participent, avec un Lapin). Nous montrons que la synchronisation des robots est la clef pour obtenir une solution. D'un côté nous prouvons que le problème ne peut pas être résolu si les robots ne sont pas synchrones, et d'un autre côté nous donnons un algorithme qui résout le problème avec des robots synchrones, sans aucune hypothèse supplémentaire. Plus de détails sont disponibles dans la version longue de l'article [1].

[†]Ce travail a été financé en partie par le projet ANR project SAPPORO, ref. 2019-CE25-0005-1 et par le projet ANR ESTATE, ref. ANR-16-CE25-0009-03.

2 Modèle

Soit $\mathcal{R} = \{r_1, r_2, \dots r_k\}$ l'ensemble des $k \geq 2$ robots modélisés comme des points dans un espace euclidien bidimensionnel. Les robots sont supposés être anonymes (ils sont indiscernables), uniformes (ils exécutent tous le même algorithme) et amnésiques (ils ne peuvent pas se souvenir des actions passées).

Soit Z un système de coordonnées global et soit $p_i(t) \in \mathbb{R}^2$ les coordonnées d'un robot ou d'un ensemble de robots localisés sur le même point dans Z au temps t. Étant donné un instant t, une *configuration* à l'instant t est définie comme étant un ensemble $C_t = \{p_1(t), p_2(t), \dots p_m(t)\}$ $(m \leq k)$ où m est le nombre de points occupés dans le plan et $p_i(t) \neq p_j(t)$ pour tout $i, j \in \{1, \dots, m\}$, $i \neq j$. Notons que les robots n'ont pas connaissance de Z. En effet, chaque robot r_i a son propre système de coordonnées Z_i centré sur la position courante de r_i. Nous supposons des robots *désorientés*, c'est-à-dire qu'ils ne s'accordent sur la direction d'aucun axe et ils ne partagent pas une unité de distance commune. Observons cependant que le plus petit cercle englobant (SEC) est indépendant du système de coordonnées.

Nous considérons le modèle synchrone (*FSYNC*) dans lequel à chaque instant t, appelé *ronde*, chaque robot r_i observe, calcule puis se déplace vers la destination calculée par rapport à son propre système de coordonnées Z_i. Pendant la phase d'observation, r_i observe les positions de tous les autres robots et construit l'ensemble $V_i(t) = \{p_1(t), \dots, p_m(t)\}$ $(m \leq k)$ où m est le nombre de points occupés dans le plan et $p_i(t) \in \mathbb{Z}^2$ est la position d'un ou plusieurs robots dans le système Z_i (translaté par $-r_i$ pour que r_i soit toujours au centre). On appelle $V_i(t)$ la *vue locale* de r_i au temps t. Notez que les robots n'ont aucun moyen de distinguer les positions occupées par un seul robot de celles occupées par plus d'un robot, c'est-à-dire qu'ils ne sont pas dotés de détection de multiplicité.

Un algorithme A est une fonction qui associe à chaque vue locale une destination. Lorsque r_i est activé au temps t, l'algorithme A affiche la destination de r_i, notée d, dans son système de coordonnées local Z_i. Le robot r_i se déplace alors vers la destination calculée. Nous supposons des mouvements *non-rigides* : lorsqu'un robot se déplace vers une destination calculée, un adversaire peut l'empêcher d'atteindre cette destination en arrêtant le robot sur un point du segment reliant sa position initiale de sa destination. Nous supposons qu'en cas d'interruption, le robot parcourt au moins $\delta > 0$. La valeur de δ est commune à tous les robots mais elle n'est pas connue. Sa valeur peut être arbitrairement petite et ne change pas pendant l'exécution de l'algorithme. Une exécution $\mathcal{E} = (C_0, C_1, \dots)$ de A est une séquence de configurations, où C_0 est la configuration initiale, et où toute configuration C_{t+1} est obtenue de C_t en appliquant A.

Un robot est dit *en panne* au temps t s'il n'est activé à aucun temps $t' \geq t$; un robot en panne arrête d'exécuter son algorithme et reste indéfiniment à la même position. La position sur laquelle se trouve un robot en panne est appelée *point de panne*. Puisque nous considérons des robots amnésiques et une configuration initiale arbitraire, nous pouvons supposer, sans perte de généralité, que si un robot tombe en panne alors cette panne se produit au début de l'exécution.

Le problème du RaL Un algorithme A résout le problème du Rassemblement (ou Rendez-vous dans le cas de deux robots) avec un Lapin (RaL) si, pour toute configuration initiale C_0 et pour toute exécution $\mathcal{E} = (C_0, C_1, \dots)$ de A en présence d'au plus un point de panne, il existe une ronde t et un point p tels que $C_{t'} = \{p\}$ pour tout $t' \geq t$. En d'autres termes, s'il existe un (unique) point de panne, tous les robots encore fonctionnels rejoignent ce point ; sinon (s'il n'existe pas de point de panne), tous les robots se rassemblent en un temps fini.

Impossibilité dans le cas semi-synchrone Dans le modèle semi-synchrone, un sous-ensemble non vide arbitraire des robots est activé et exécute l'algorithme à chaque ronde. Dans ce cas, le problème du RaL n'a pas de solution, sans hypothèse supplémentaire. En effet, supposons deux robots r et r' à des positions distinctes. On considère l'exécution semi-synchrone où à chaque ronde, (a) si l'action d'un robot consiste à rester immobile, alors uniquement ce robot est activé à l'infini (l'autre robot est considéré comme en panne), (b) si les robots bougent et ont des destinations différentes alors ils sont activé simultanément, (c) si les robots bougent et ont la même destination, un seul robot est activé. Dans cette exécution, le RaL n'est jamais réalisé, et au plus un robot n'est jamais activé (et donc considéré en panne). Cette preuve s'étend à un nombre quelconque de robots, en prenant une configuration initiale constituée de deux groupes de robots de tailles arbitraires, un groupe agissant comme r et l'autre comme r'.

3 Solution proposée

L'algorithme proposé se base sur le nombre de positions (appelées points dans la suite) observées, leur alignement, et, dans le cas où uniquement 2 points sont visibles, leur distance. En effet, même si les système de coordonnées des robots sont arbitraires, il est important de noter qu'ils sont fixes et ne changent pas au cours de l'exécution [5]. Cela implique que deux robots qui se rapprochent peuvent effectuer des actions différentes en fonction de la distance qui les sépare. Ainsi, nous partitionnons l'ensemble $Conf$ de toutes les configurations possibles comme suit : $Conf = \mathcal{T}_{SEC} \cup \mathcal{T}_{ext} \cup Conf_2$

— $\mathcal{T}_{ext}(p)$ (ou simplement $\mathcal{T}_{ext}$ lorsque p est arbitraire) dénote l'ensemble des configurations où tous les points sont alignés, et un seul robot extrême p est à distance e du robot le plus proche, avec $\frac{e}{d} \in \left\{\frac{1}{9}, \frac{1}{8}, \frac{9}{80}, \frac{10}{81}, \frac{10}{18}, \frac{9}{16}, \frac{80}{81}\right\}$, où d désigne la distance entre les deux points aux extrémités.

— $\mathcal{T}_{SEC}$ désigne l'ensemble des configurations qui ont plus de deux points, et qui ne sont pas dans $\mathcal{T}_{ext}$.

— $Conf_2$ désigne les configurations constituées de deux points seulement. On partitionne cet ensemble en fonction de la distance entre d_r les deux points (vu par le robot r). Cette partition est définie en fonction du *niveau* l_r d'un robot r, qui est le nombre tel que $d_r \in [2^{-l_r}, 2^{-l_r+1})$. On a alors $\forall k \geq 0$:

$$\begin{aligned}
\mathcal{A}^k &= \{C \in Conf_2 \mid S_k \leq l_r < S_k + k\} \\
\mathcal{B}^k &= \{C \in Conf_2 \mid S_k + k \leq l_r < S_k + 2k\} \\
C_1^k &= \{C \in Conf_2 \mid S_k + 2k \leq l_r < S_k + 2k + 1\} \\
C_2^k &= \{C \in Conf_2 \mid S_k + 2k + 1 \leq l_r < S_k + 2k + 2\} \\
C_3^k &= \{C \in Conf_2 \mid S_k + 2k + 2 \leq l_r < S_k + 2k + 3\}
\end{aligned}$$

Où $S_k = k(k+2)$, de sorte que $S_{k+1} = S_k + 2k + 3$ et tous les niveaux ≥ 0 sont considérés. Pour des raisons de simplicité, au lieu de définir un ensemble vide, nous fixons par convention $\mathcal{A}^0 = \{C \in Conf_2 \mid l_r \leq 1\}$. Ainsi, la séquence infinie $\left(\mathcal{A}^k \cup \mathcal{B}^k \cup C_1^k \cup C_2^k \cup C_3^k\right)_{k \geq 0}$ est une partition de $Conf_2$. On remarque que la partition déduite dépend du robot qui observe. Ainsi, deux robots différents peuvent voir une configuration donnée C dans différents sous-ensembles de $Conf_2$ au même temps t étant donné qu'ils peuvent avoir des unités de distance différentes.

Lorsque la configuration courante ne comporte que deux points à distance d, nous utilisons `Aller` (e) pour indiquer au robot de se déplacer vers l'autre robot en parcourant une distance $e \times d$. Rappelons que les actions des robots dépendent de leur système de coordonnées respectif. Ainsi, lorsque la configuration courante C est dans $Conf_2$, un robot r oriente la ligne passant par les deux points occupés dans C à l'aide de son système de coordonnées. Si l'autre point (où r ne se trouve pas) est situé à l'est ou au nord (dans le cas où aucun point n'est à l'est de l'autre), alors r considère qu'il est à gauche de l'autre point, sinon r se voit à droite de l'autre point.

Notre algorithme est présenté dans la Figure 1. En fonction de la configuration vue par le robot r, et de sa position par rapport à l'autre robot (dans sa vue), le robot se déplace vers la destination indiquée par la flèche. Par exemple, si la configuration ne contient que deux points, et que le robot r se voit à gauche, et voit une distance d_r ayant un niveau l_r telle que la configuration est dans l'ensemble C_3^k, alors le robot se déplace au milieu des deux points.

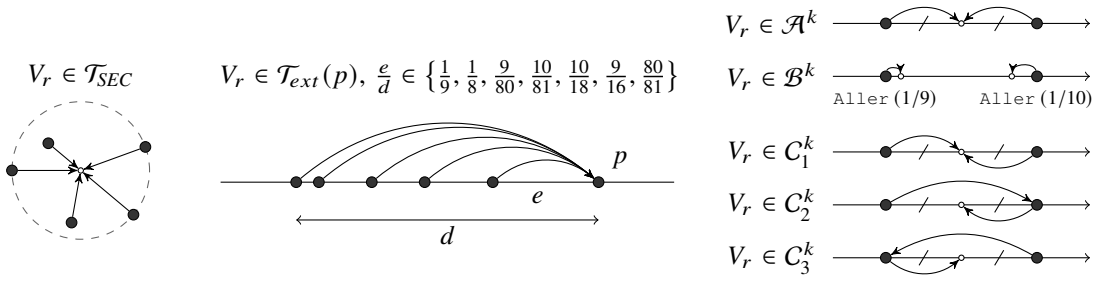

FIGURE 1: Algorithme tolérant un ou plusieurs lapins colocalisés. Exécuté par un robot r ayant une vue locale V_r. Lorsque $V_r \in Conf_2$, la destination dépend aussi de l'emplacement (droite ou gauche) auquel r se voit.

Théorème 1. *L'algorithme de la Figure 1 résout le problème du RaL avec $n \geq 2$ robots synchrones désorientés.*

Aperçu de la preuve. Tout d'abord, il est facile de voir que si les robots ne sont pas alignés, ils le deviennent après un nombre fini de rondes grâce à la règle du déplacement vers le centre du SEC (et ils restent alignés grâce aux autres règles). Supposons donc que les robots sont alignés. La suite d'ensembles de configurations qui partitionne l'ensemble $Conf_2$ est importante pour tolérer les unités de distances différentes des robots et le fait que les mouvements ne sont pas rigides. Cette séquence est construite de telle sorte que les robots sont assurés de se rapprocher tant que le regroupement n'est pas réalisé (les mouvement se font toujours en direction des autres robots). Par ailleurs, il est important de réaliser que le nombre de niveaux pris en charge par chaque ensemble de configurations ($\mathcal{A}^k$ par exemple) augmente avec k. Si le regroupement n'est pas réalisé avant, les robots seront suffisamment proches (et donc leur niveau suffisamment grand) pour que tous les robots se voient dans la même phase. En effet, la différence maximum Δ de niveau entre les robots ne change pas lorsque les robots se rapprochent, il existe donc un k tel que $\Delta < k$ et donc tous les robots se voient dans une phase $\mathcal{A}^k$ (puis un mélange de $\mathcal{A}^k$ et $\mathcal{B}^k$, puis tous en phase $\mathcal{B}^k$, etc...). Cela permet, dans le cas où tous les robots sont corrects de réaliser le regroupement en une ronde, une fois tous les robots en phase $\mathcal{A}^k$.

Considérons maintenant le cas où il y a un point de panne, et que les robots sont suffisamment proches pour que leurs mouvements soient rigides (plus aucun mouvement ne peut être interrompu par un adversaire), et que leurs niveaux soient tous plus grands que Δ. Si la phase est $\mathcal{T}_{SEC}$ ou $\mathcal{T}_{ext}$, alors tous les robots ont la même destination, créant ainsi 2 points dont un point contenant tous les robots corrects. Chaque robot exécute alors un mouvement issu d'une des phases de $Conf_2$. La configuration obtenue est alors obligatoirement de type $\mathcal{T}_{ext}(p)$ avec p la position panne. En effet les robots corrects effectuent `Aller` $(1/2)$, `Aller` (1), `Aller` $(1/9)$, ou `Aller` $(1/10)$, et quelque soit la combinaison, si le regroupement n'est pas réalisé, le rapport de distance entre le point de panne et le robot correct le plus proche est un élément de l'ensemble $\left\{\frac{1}{9}, \frac{1}{8}, \frac{9}{80}, \frac{10}{81}, \frac{10}{18}, \frac{9}{16}, \frac{80}{81}\right\}$.

Si le point de panne est aussi occupé par des robots corrects, alors, soit il arrive un temps où aucun robot correct n'est sur le point de panne (le cas précédent permet alors de conclure), soit on peut montrer qu'après avoir exécuté successivement les phases présentent dans $Conf_2$, les robots corrects se déplacent tous vers la position panne. La preuve complète peut être trouvée dans la version longue de l'article [1]. □

4 Conclusion

Nous avons présenté la première solution au problème du RaL. Notre solution est auto-stabilisante (la configuration initiale peut inclure des points de multiplicité ou être bivalente), ne repose pas sur des hypothèses supplémentaires telles que la détection de multiplicité, une orientation commune, etc.

Les très faibles capacités des robots que nous avons considérés rendent le problème insoluble dans des modèles d'exécution plus relâchés, comme le modèle SSYNC. Cependant, il peut être possible de résoudre le RaL dans le modèle SSYNC avec un certain nombre d'hypothèses supplémentaires : configurations initiales non bivalentes (sinon, le résultat d'impossibilité du rendez-vous indulgent s'applique [1]), possibilité de détecter les points de multiplicité (sinon, le résultat d'impossibilité pour le rassemblement classique s'applique), et un système de coordonnées persistant (sinon, le résultat d'impossibilité de Defago et al. [3] à propos de RaL s'applique). Cette question ouverte est laissée pour de futures recherches.

Références

[1] Quentin Bramas, Anissa Lamani, and Sébastien Tixeuil. Stand up indulgent gathering. *Theor. Comput. Sci.*, 939 :63–77, 2023.

[2] Quentin Bramas and Sébastien Tixeuil. Wait-free gathering without chirality. In *International Colloquium on Structural Information and Communication Complexity*, pages 313–327. Springer, 2015.

[3] Xavier Défago, Maria Potop-Butucaru, and Philippe Raipin Parvédy. Self-stabilizing gathering of mobile robots under crash or byzantine faults. *Distributed Comput.*, 33(5) :393–421, 2020.

[4] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro (Eds.). *Distributed Computing by Mobile Entities-Current Research in Moving and Computing*. Lecture Notes in Computer Science, 11340. Springer, 2019.

[5] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots : Formation of geometric patterns. *SIAM J. Comput.*, 28(4) :1347–1363, 1999.