Synthèse d'algorithmes pour robots mobiles : le cas du regroupement sur un anneau

Laure Millet, Maria Potop-Butucaru, Nathalie Sznajder, Sébastien Tixeuil

Synthèse d'algorithmes pour robots mobiles : le cas du regroupement sur un anneau.

Laure Millet[1,2], Maria Potop-Butucaru[1,2], Nathalie Sznajder[1,2], et Sébastien Tixeuil[1,2,3]

[1] *Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France*
[2] *CNRS, UMR 7606, LIP6, F-75005, Paris, France*
[3] *Institut Universitaire de France*

Les tâches susceptibles d'être exécutées par des robots mobiles sont de plus en plus nombreuses et en complexité croissante. Jusqu'à présent, les réseaux de robots ont été étudiés de manière empirique et la plupart des résultats ont été validés principalement manuellement, par des simulations ou des preuves partielles où l'optimalité est rarement prise en compte. Nous proposons une méthode basée sur les jeux d'accessibilité qui permet de générer automatiquement des algorithmes distribués optimaux pour des réseaux de robots autonomes. Cette méthode est testée sur le problème de rassemblement de ces robots sur un point d'un anneau (précis et non connu initialement).

Keywords: algorithm synthesis, autonomous mobile robots, game

1 Introduction

Nous considérons le modèle introduit par Suzuki & Yamashita [**?**] où les robots sont anonymes et identiques, ils ne peuvent être distingués les uns des autres et exécutent le même protocole. Ils n'ont pas de mémoire (donc leurs actions ne dépendent pas des actions précédentes), ils n'ont aucun sens de l'orientation et ne peuvent communiquer entre eux. Cependant, ils ont la capacité d'observer leur environnement et de voir l'agencement des autres robots. Ces robots fonctionnent selon un cycle de trois phases : *Look*, *Compute* et *Move*. Pendant la première phase, les robots examinent le monde qui les entoure. Les informations ainsi collectées leur permettent de calculer (phase *Compute*) un futur mouvement qui sera rendu effectif dans la dernière phase. Dans ce modèle robotique on peut distinguer deux types de modèles d'exécution en fonction du niveau de synchronisme voulu : le modèle synchrone et le modèle asynchrone. On s'intéresse ici au modèle synchrone qui possède deux variantes : FSYNC ou *Fully-Synchronous* où chaque cycle est exécuté de manière atomique par tous les robots, et SSYNC ou *Semi-Synchronous* où pour chaque cycle un sous-ensemble de robots effectue les trois phases de façon atomique.

Bien que ces robots évoluent dans un espace euclidien à deux dimensions, nous travaillons sur une version discrète de cet espace. Cette représentation consiste alors en un graphe où l'espace est divisé tel que : chaque noeud du graphe représente un emplacement, et chaque arc représente la possibilité pour un robot de se déplacer d'un emplacement vers un autre.

L'un des problèmes étudiés dans la littérature est le rassemblement de tous les robots en un point non défini au préalable, quelles que soient les positions initiales des robots. Une fois rassemblés les robots devront rester immobiles. Sur un anneau ce problème a été prouvé impossible dans certains cas : s'il n'y a que deux robots ou lorsque la configuration initiale est périodique, ou lorsque la configuration initiale contient un axe de symétrie ne passant par aucun noeud. Le rassemblement est aussi impossible si les robots ne sont pas capables de détecter la présence de plusieurs robots sur un même noeud. Des stratégies pour les cas restants ont été proposées mais elles ne prennent pas en compte les performances et leur validité n'est pas vérifiée.

La seule tentative d'automatiser la recherche d'algorithmes pour de tels robots est à notre connaissance [**?**] qui se restreint aux configurations asymétriques et qui interdit le déplacement de plusieurs robots à partir d'une même configuration, mais ne traite pas le problème dans sa globalité.

Avec des robots synchrones, on peut voir la recherche d'un algorithme comme un jeu où l'algorithme (la coalition de robots) est un joueur qui pour chaque configuration choisit comment chaque robot se déplacera afin d'obtenir le rassemblement. Le second joueur (environnement) est dû à l'absence de sens de l'orientation des robots : quand un robot est sur un axe de symétrie, il est incapable de distinguer les directions de leurs symétriques, et donc l'environnement choisit le sens effectif.

2 Rappel : Jeux d'accessibilité, définitions

Un jeu d'accessibilité [**?**] est composé d'une arène et d'une condition d'acceptation définie par un ensemble d'états à atteindre. Une *arène* est un graphe $\mathcal{A} = (V,E)$ où l'ensemble des noeuds $V = V_p \uplus V_o$ est partitionné entre les noeuds du joueur/protagoniste V_p et les noeuds de l'environnement/opposant V_o. Les arcs $E \subseteq V \times V$ définissent la notion de successeur. On note V^∞ une sequence finie ou infinie d'éléments de V. Une *partie* dans cette arène est alors un chemin dans le graphe $\pi = v_0 v_1 \cdots \in V^\infty$, tel que pour tout $0 < i < |\pi|$, $(v_{i-1}, v_i) \in E$. Une *stratégie* pour pour le protagoniste est de choisir pour chacun de ces noeuds un successeur : c'est alors une fonction (partielle) $\sigma : V^* \cdot V_p \to V$ qui en fonction de l'historique de la partie choisit un successeur parmi ceux possibles dans l'arène. On dit qu'une stratégie est sans mémoire si elle ne dépend que de la position courante : on a alors $\sigma : V_p \to V$. Une partie π obéit à une stratégie σ si, pour tout $0 < i < |\pi|$, si $v_{i-1} \in V_p$, $v_i = \sigma(v_0 \cdots v_{i-1})$.

Si la condition d'acceptation pour le joueur est l'accès à $T \subseteq V$, l'ensemble des parties gagnantes est donné par $Reach(T) = \{\pi = v_0 v_1 \cdots \in V^\infty \mid \exists 0 \leq i < |\pi| : v_i \in T\}$. Une stratégie est dite *gagnante* pour une arène $\mathcal{A}$ à partir d'un noeud v, si toutes les parties pouvant être obtenues selon cette stratégie sont gagnantes.

Les noeuds à partir desquels il existe une stratégie gagnante sont appelés *positions gagnantes*.

Théorème 2.1. *Dans un jeu d'accessibilité, l'ensemble des positions gagnantes pour le joueur est calculé en temps linéaire à la taille de l'arène. À partir de n'importe quelle position, le protagoniste a une stratégie gagnante si et seulement si il a une stratégie gagnante sans mémoire.*

En utilisant les jeux afin d'obtenir une stratégie pour la coalition de robots, on obtiendra une stratégie centralisée, c'est notre encodage de l'arène qui va permettre d'obtenir des stratégies distribuables.

3 Représentation formelle et codage du modèle robotique sur un anneau

3.1 Configurations robotiques : symétries et équivalences

Considérons un modèle constitué de k robots sur un anneau de taille $n > k$. Une configuration du système est alors représentée par $C = (d_1, \cdots, d_k)$, tel que $\Sigma_{i=1}^{k} d_i = n - k$, et $d_i \in \{-1, 0, \cdots, n-1\}$, où d_i représente le nombre de noeuds vides entre le i^e robot et son voisin le plus proche dans le sens des aiguilles d'une montre. Ainsi $d_i = 0$ représente le fait que ces deux robots sont adjacents et $d_i = -1$ qu'ils forment une tour (superposition de robots sur un même noeud). On note $\mathcal{C}$ l'ensemble des configurations. Dans une configuration, chaque robot a une observation de l'anneau centrée sur sa propre position. Deux robots sur un même noeud devant se comporter de la même façon, on définit la *vue* d'un robot de façon à ce que tous les robots formant une tour aient la même vue : la vue d'un robot i de la configuration C est donnée par $view_i(C) = \{(d_i, \ldots, d_j), (d_j, \ldots, d_i)\}$, où $i < j$ sont respectivement le plus petit et le plus grand index tel que $d_i \neq -1$ (resp.$d_j \neq -1$). Les deux éléments de l'ensemble constituant la vue correspondent à l'observation que fait un robot de la configuration selon la direction dans laquelle il « regarde ». Un robot i est dit désorienté si $|view_i(C)| = 1$; en effet un robot dont la vue est un singleton ne peut se repérer car il voit la même chose dans les deux directions. Soit $\mathcal{V}$ l'ensemble des vues correspondant à $\mathcal{C}$.

Les robots étant anonymes, une configuration n'est pas modifiée par des rotations successives de l'anneau. Soit $\circlearrowleft \subseteq \mathcal{C} \times \mathcal{C}$ la relation rotation définie par : pour toutes configurations $C, C' \in \mathcal{C}$, $C \circlearrowleft C'$ *ssi* $C = (d_i, d_{i+1}, \cdots, d_{i+k-1})$ et $C' = (d_{i+1}, d_{i+2}, \cdots d_{i+k})$, où $+$ est une somme modulo k. Les robots n'ayant aucun sens de l'orientation, deux configurations peuvent représenter la même situation générale de robots si l'une est le miroir de l'autre. En effet si $C = (d_1, \cdots, d_k)$ et $C' = (d_k, \cdots, d_1)$, alors pour tout robot i, $view_i(C) = view_i(C')$. On définit la relation miroir $\sim \subseteq \mathcal{C} \times \mathcal{C}$ par $C \sim C'$ *ssi* $C = (d_1, \cdots, d_k)$ et $C' = (d_k, \cdots, d_1)$. À partir de ces deux relations, on définit la relation d'equivalence $\equiv \subseteq \mathcal{C} \times \mathcal{C}$ par : $\equiv \stackrel{\text{def}}{=} (\circlearrowleft \cup \sim)^*$. Cette relation identifie toutes les configurations où les robots se comportent de la même façon.

Soit $[C]_\equiv$ la classe d'équivalence de $C \in \mathcal{C}$, et soit l'application $rep : \mathcal{C}/\equiv \to \mathcal{C}$, telle que $rep([C]_\equiv) \in [C]_\equiv$, pour tout $C \in \mathcal{C}$, qui associe à chaque classe d'équivalence un représentant unique dans la classe.

3.2 Encoder les mouvements des robots dans les transitions entre configurations

Afin de définir les transitions du système on définit $\mathbb{M} = \{\curvearrowright, \curvearrowleft, \uparrow\}$ les différents mouvement des robots, $\curvearrowright$ et $\curvearrowleft$ indiquent respectivement un mouvement dans le sens horaire et dans le sens anti-horaire, alors que $\uparrow$ indique un non-mouvement. Quand un robot i bouge, il modifie les distance d_i et d_{i-1} (incrémentant l'une et décrémentant l'autre). L'ensemble des mouvements des robots pour un cycle peut alors être représenté par un vecteur pour chaque robot. Pour le robot $1 \leq i \leq k$, ce vecteur est noté respectivement $m^{i,\curvearrowleft}$, $m^{i,\curvearrowright}$ et m^0 si le robot effectue respectivement les mouvements : $\curvearrowleft$, $\curvearrowright$ et $\uparrow$. Ces vecteurs sont des k-tuples définis respectivement par $m_i^{i,\curvearrowleft} = 1$, $m_{i-1}^{i,\curvearrowleft} = -1$ et $m_j^{i,\curvearrowleft} = 0$ pour tout $1 \leq j \leq k$ avec $i \neq j$, et $m_i^{i,\curvearrowright} = -1$, $m_{i-1}^{i,\curvearrowright} = 1$ et $m_j^{i,\curvearrowright} = 0$ pour tout $1 \leq j \neq i \leq k$, et finalement $m^0 = m_j^0 = 0$, pour tout $1 \leq j \leq k$.

La configuration atteinte après le mouvement de tous les robots est obtenue par la somme des vecteurs des mouvements des robots avec la configuration courante. Pour toute configuration $C \in \mathcal{C}$, et tout mouvement $(m_i)_{1\leq i\leq k}$, on note $C' = C \oplus (m_i)_{i\in\{1,\dots,k\}}$ la configuration successeur de C. †

4 Encodage de l'arène pour un objectif de rassemblement

On construit une arène afin que le joueur ait une stratégie gagnante si et seulement si un algorithme distribué permet aux robots de se rassembler sur un même noeud quelle que soit la configuration initiale. Dans chaque configuration les robots ont le choix parmi les actions : $\Delta = \{\curvearrowright, \curvearrowleft, \uparrow, ?\}$, qui contient $\mathbb{M}$, l'ensemble des movements possibles, et "?" utilisée par les robots désorientés indiquant qu'ils veulent bouger, mais ne peuvent décider de la direction exacte. On note $\overline{\curvearrowright} = \curvearrowleft$, $\overline{\curvearrowleft} = \curvearrowright$, $\overline{\uparrow} = \uparrow$ et $\overline{?} = ?$.

Notre arène est $\mathcal{A}_{\text{gather}} = (V_p \uplus V_o, E)$, avec $V_p = (\mathcal{C}/\equiv)$ les états du joueur et $V_o = \mathcal{C} \times (\Delta^k)$ les états de l'environnement. La taille de l'arène est alors linéaire par rapport à n et exponentielle par rapport à k.

Les arcs de l'arène sont définis par la relation E comme une alternance stricte entre les deux joueurs : $E \subseteq (V_p \times V_o) \cup (V_o \times V_p)$.

$V_p \to V_o$. À partir d'un noeud du joueur, ce dernier choisit pour chaque robot un mouvement. Cela est possible uniquement si dans une classe d'équivalence deux robots avec la même vue prennent la même décision. Une *fonction de décision* f est une fonction qui propose un mouvement en fonction de la vue d'un robot. Elle est définie par $f : \mathcal{V} \to \Delta$ telle que, pour toute vue $V \in \mathcal{V}$, si $|V| = 1$ $f(V) \in \{\uparrow, ?\}$ et si $f(V) = ?$ alors $|V| = 1$, un robot désorienté ne pouvant choisir que de bouger ou de rester immobile. Quand on applique une fonction de décision, les mouvements des robots doivent être cohérents avec un sens de l'orientation commun. Étant donné $C = (d_1, \dots, d_k) \in \mathcal{C}$, et $f : \mathcal{V} \to \Delta$, pour tout $1 \leq i \leq k$, on définit $f(C,i) = f(view_i(C))$ si $(d_i, \cdots, d_k, d_1, \cdots d_{i-1})$ est le plus petit élément de $view_i(C)$ dans l'ordre lexicographique, alors, et $f(C,i) = \overline{f(view_i(C))}$ sinon. On a alors : pour tout $v \in V_p, v' \in V_o$, $(v,v') \in E$ *ssi* il existe une fonction de décision f telle que $v' = \big(C, (a_1, \dots, a_k)\big)$ avec $C = rep(v) = (d_1, \dots, d_k)$ et pour tout $1 \leq i \leq k$, $a_i = f(C,i)$.

$V_o \to V_p$. Le jeu continu d'une position de l'environnement où les choix du joueur sont mémorisés. Les décisions de l'environnement ramènent le jeu dans une position du joueur qui est la classe d'équivalence de la configuration résultante des mouvements des robots sur la configuration de l'état courant. Si un robot désorienté a décidé de bouger, l'environnement choisit le mouvement qu'effectuera le robot parmi $\{\curvearrowright, \curvearrowleft\}$.

Dans $v' = (C, (a_1, \dots, a_k)) \in V_o$, un ensemble de mouvements $(m_i)_{i\in\{1,\dots,k\}} \in \mathbb{M}^k$ est v'-*compatible* si : pour tout $1 \leq i \leq k$ tel que $a_i \neq ?$, $m_i = a_i$, et pour tout $1 \leq i \leq k$ tel que $a_i = ?$, $m_i \neq \uparrow$.

Le passage d'une position de l'opposant à une position du protagoniste est alors : pour tout $v \in V_p$, $v' = (C, (a_1, \dots, a_k)) \in V_o$, $(v', v) \in E$ *ssi* il existe un tuple v'-compatible $(m_i)_{i\in\{1,\dots,k\}}$ tel que $v = [C \oplus (m_i)_{i\in\{1,\dots,k\}}]_\equiv$.

Théorème 4.1. *Les positions gagnantes pour le joueur dans le jeu $(\mathcal{A}_{gather}, W)$ correspondent exactement aux configurations à partir desquelles les robots peuvent effectuer le rassemblement.*

†. Afin de prendre en compte toutes les subtilités du modèle, le calcul de $C \oplus (m_i)_{i\in\{1,\dots,k\}}$ est décrit précisément dans [**?**].

5 Synthèse d'un algorithme de rassemblement pour 3 robots

Cette construction d'arène permet de synthétiser un algorithme déterministe de rassemblement de k robots dans un anneau de taille n. Soit $T = [(-1,\cdots,-1,n-1)]_{\equiv} \in V_p$ la classe d'équivalence de toutes les configurations où tous les robots sont rassemblés sur un seul et même nœud. Nous avons implémenté l'arène pour trois robots pour différentes tailles d'anneau ($n \in [3,15]$ et $n = 100$) dans l'outil de résolution de jeux UPPAAL TIGA [‡]. Nous avons pu vérifier les résultats d'impossibilité à partir des configurations périodiques. Nous avons obtenu que pour toutes les configurations exceptées les périodiques il existe une stratégie gagnante.

On s'intéresse à l'obtention de stratégies optimales, pour cela nous avons ajouté des poids sur les arcs du graphe en fonction du nombre de robots qui bougent. On a alors obtenu des stratégies optimales pour le joueur. Une étude de ces stratégies nous a permis de regrouper ces données en fonction du type de configurations : si il existe une tour, si la configuration est symétrique, *etc*.

Stratégie		
V_p	φ	V_o
$[(-1,-1,n-1)]_{\equiv}$		$((-1,-1,n-1),(\uparrow,\uparrow,\uparrow))$
$[(-1,d_1,d_2)]_{\equiv}$	$d_1 < d_2$	$((-1,d_1,d_2),(\uparrow,\uparrow,\curvearrowleft))$
$[(-1,\frac{n-1}{2},\frac{n-1}{2})]_{\equiv}$		$((-1,\frac{n-1}{2},\frac{n-1}{2}),(\uparrow,\uparrow,?))$
$[(d_1,d_1,d_2)]_{\equiv}$	$rep = (d_1,d_1,d_2)$	$((d_1,d_1,d_2),(\curvearrowright,\uparrow,\curvearrowleft))$
$[(d_1,d_1,d_2)]_{\equiv}$	$rep = (d_2,d_1,d_1)$	$((d_2,d_1,d_1),(\uparrow,\uparrow,?))$
$[(d_1,d_2,d_3)]_{\equiv}$	$d_1 < d_2 < d_3$	$(\curvearrowright,\uparrow,\uparrow))$

Algorithme distribué		
$view_r$	φ	$f : view_r$
$\{(d_1,-1,d_2),(d_2,-1,d_1)\}$	$d_1 < d_2$	$\curvearrowright$
$\{(\frac{n-1}{2},-1,\frac{n-1}{2})\}$		?
$\{(d_1,d_1,d_2),(d_2,d_1,d_1)\}$	$d_1 < d_2$	$\curvearrowright$
$\{(d_1,d_2,d_1)\}$	$d_2 < d_1$	?
$\{(d_2,d_1,d_3),(d_3,d_1,d_2)\}$	$d_1 < d_2 < d_3$	$\curvearrowright$

Le tableau Stratégie présente les types de configurations définis par un ensemble de configurations délimité par une propriété φ, et la stratégie choisie dans ces types de configuration. Le tableau Algorithme distribué présente l'algorithme tel qu'exécuté par un robot r dont la vue fait partie de l'ensemble de vues $\{view\}$ délimitées par la propriété φ. Pour toutes vues différentes de celles présentées dans l'algorithme le mouvement est $\uparrow$.

Cet algorithme est correct par construction pour $n \in [3,15]$ et $n = 100$. Une preuve par induction montre facilement que l'algorithme est correct pour tout n [**?**].

6 Conclusions

Nous pensons que le cadre de notre encodage (typiquement, les configurations et les transitions entre les configurations) peut être réutilisé pour différents problèmes sur les réseaux en forme d'anneau, comme l'exploration avec arrêt ou l'exploration perpétuelle, et facilement étendu à d'autres topologies.

Dans les travaux futurs on voudrait s'intéresser à la synthèse de systèmes paramétrés. De plus, la taille du jeu augmente rapidement avec le nombre de robots ; il est prévu que des optimisations et/ou heuristiques soient découvertes afin d'aider à la recherche d'algorithmes paramétrés.

Nous nous sommes concentrés sur les modèles atomiques FSYNC et SSYNC. Dans un modèle asynchrone (ASYNC) où les robots ne peuvent pas maintenir une vision globale actuelle du système, un nouvel encodage de jeu à deux joueurs doit être envisagé (car les jeux distribués sont généralement indécidables).

Références

[BDP+12] François Bonnet, Xavier Défago, Franck Petit, Maria Potop-Butucaru, and Sébastien Tixeuil. Brief announcement : Discovering and assessing fine-grained metrics in robot networks protocols. In *Proc. of SSS'12*, volume 7596 of *LNCS*, pages 282–284. Springer, 2012.

[Maz01] René Mazala. Infinite games. In *Automata, Logics, and Infinite Games*, pages 23–42, 2001.

[MPBST14] L. Millet, M. Potop-Butucaru, N. Sznajder, and S. Tixeuil. On the synthesis of mobile robots algorithms : The case of ring gathering. In *Proc. of SSS'14*, volume 8756 of *LNCS*, pages 237–251. Springer, 2014.

[SY99] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots : Formation of geometric patterns. *SIAM Journal on Computing*, 28(4) :1347–1363, 1999.

‡. http ://people.cs.aau.dk/ adavid/tiga/