Routage dans un réseau de robots

Pirro Bracka, Serge Midonnet, Gilles Roussel

Routage dans un réseau de robots

Pirro Bracka, Serge Midonnet et Gilles Roussel

IGM – Université de Marne-la-Vallée – 5 bd Descartes – Champs-sur-Marne – 77454 Marne-la-Vallée

Cet article traite de la communications dans un réseau de robots. Plus précisément, nous nous intéressons à la communication en mode *ad hoc* dans laquelle tous les robots participent à l'acheminement des messages. La particularité de ce réseau est que ses éléments sont souvent déconnectés mais également que les déplacements des robots peuvent être contraints. Dans ce cadre, nous proposons un algorithme pour le routage des messages basé sur l'ordonnancement des déplacements des robots qui assure que n'importe quelle communication entre deux robots du réseau se fera en temps borné.

Keywords: routage, réseau ad hoc, communication asynchrone

1 Introduction

Un réseau *ad hoc* sans fil est constitué d'un ensemble d'éléments mobiles formant un réseau provisoire sans infrastructure fixe ni gestion centralisée. Dans un tel réseau, comme les interfaces ont une portée limitée, il se peut qu'un élément mobile utilise d'autres éléments pour acheminer un message vers sa destination. Ainsi, chaque élément mobile se comporte comme un routeur.

Dans cet article, nous traitons le problème de la communication dans un réseau *ad hoc* de robots. Chaque robot est autonome. Il se déplace dans une zone limitée pour effectuer diverses tâches et il communique avec les autres robots *via* des interfaces sans fil. Dans ce cadre, nous nous sommes intéressés à la communication entre les robots sans l'aide de moyen de communication global. Cette particularité peut être dûe à l'insuffisance des infrastructures fixes ou imposée par des objectifs de confidentialité. Chaque robot est ainsi contraint, s'il ne veut pas retourner systématiquement à une base, de communiquer *via* les autres robots, quand ils sont à sa portée, pour transmettre les données qu'il a collectées ou pour obtenir de nouveaux objectifs.

Les protocoles de communication existants pour les réseaux *ad hoc* [BMJ+98, RT99, JM96, PC97, Per01] ne sont pas adaptés à ce type de réseau. Ils ont été conçus pour le cas où la déconnexion est une "exception" tandis que pour nous, elle est plutôt la règle. Les liens dans notre réseau ont une durée de vie très limitée : le temps pendant lequel les robots sont à la portée les uns des autres. D'autre part, nous avons l'avantage de pouvoir contrôler les déplacements de nos robots. Ceci nous permet de réduire fortement le degré de liberté de notre système.

Il n'existe pas, à notre connaissance, d'autre travail sur ce sujet. Le travail le plus proche dont nous avons connaissance [LR00] s'intéresse à la modification de la trajectoire des éléments mobiles pour maintenir la connectivité dans un réseau *ad hoc*.

Dans ce contexte, nous essayons de trouver un ordonnancement des déplacements des robots et un schéma de routage s'assurant que tous les robots pourront communiquer les uns avec les autres en temps borné. Pour résoudre ce problème, nous commençons dans la section 2 par limiter les degrés de liberté du système. La contrainte la plus forte que nous imposons est que chaque robot attend ses voisins sur les *points de rendez-vous* avant de continuer son déplacement. Dans la section 3, nous proposons un algorithme très simple pour établir les déplacements pour chaque robot. Les points de rendez-vous sont arbitrairement numérotés et chaque robot les parcourt «dans l'ordre». Dans la section 4, nous présentons les propriétés qui sont respectées par l'ordonnancement produit. Celles-ci nous assurent qu'un message émis par un robot atteindra sa destination dans un temps borné. En conclusion, dans la section 5, nous présentons le calcul des plus courts chemins dans notre réseau utilisé pour effectuer le routage.

2 Les contraintes

Si aucune contrainte n'est placée sur les déplacements des robots, il n'est pas possible d'assurer que la communication aura lieu en temps borné. En effet, il est toujours possible de trouver un déplacement dans lequel les robots ne se rencontrent jamais. C'est pourquoi, nous avons restreint notre problème en contraignant les déplacements et les communications.

D'abord nous avons fixé quelques contraintes sur la topologie du réseau :

1. *l'espace est divisé en zones qui ne se recouvrent pas ;*
2. *chaque zone contient un seul robot ;*
3. *chaque robot ne peut communiquer qu'avec les robots qui se déplacent dans les zones adjacentes ;*
4. *chaque robot communique avec un seul robot à la fois ;*
5. *la position des points de communication entre robots est fixe, nous les appelons points de rendez-vous ;*
6. *il existe au plus un point de rendez-vous entre deux robots ;*
7. *le réseau est connexe, c'est-à-dire qu'entre deux robots il existe toujours une suite de points de rendez-vous les reliant.*

Nous supposons ici que les points de rendez-vous sont fixés une fois pour toute pour chaque robot. Dans la pratique, ces points de rendez-vous sont plutôt des zones de rencontre dans lesquelles deux robots peuvent communiquer.

Le réseau de la figure 1 respecte ces contraintes. Parmi d'autres informations, il indique que quatre robots sont attachés aux zones A, B, C et D, que les robots de la zone A et B ont un point rendez-vous en commun et que les robots de la zone A et C peuvent seulement communiquer en envoyant des messages *via* les robots des zones B ou D.

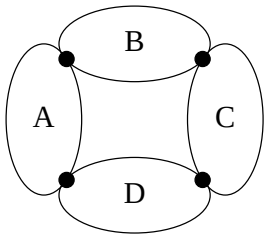

FIG. 1: Exemple de réseau

Malgré ces contraintes, il est encore possible que les robots parcourent tous leurs points rendez-vous sans se rencontrer. Il est ainsi nécessaire de limiter encore les déplacements des robots. Compte tenu des ressources limitées des robots en terme de processeur, de mémoire et d'énergie, nous avons essayé de trouver une contrainte qui puisse être respectée avec un coût minimum. Une solution simple consiste à forcer les robots à s'attendre mutuellement à leur point de rendez-vous.

Plus précisément nous avons imposé la contrainte suivante sur l'ordonnancement :

> *le premier robot qui arrive dans le point de rendez-vous attend son homologue avant de continuer son déplacement.*

Cette contrainte sur les déplacements est très forte et il est très facile de trouver un ordonnancement dans lequel certains robots ne se déplacent jamais. Par exemple, si le robot A attend B, B attend C, C attend D et D attend A, les robot seront bloqués (*dead-lock*).

C'est pourquoi, nous proposons un algorithme permettant de trouver un ordonnancement pour les déplacements des robots qui assure que deux robots qui ont un point rendez-vous en commun se rencontreront au bout d'un temps borné. Cette condition garantit qu'un message suivant une suite de points de rendez-vous atteindra sa destination en temps borné.

3 Trouver un ordonnancement

Afin de fournir un ordonnancement pour les déplacements des robots, nous construisons le graphe des points de rendez-vous $\mathcal{G}_{rdv} = (R, E)$. L'ensemble des sommets R est l'ensemble des robots (éléments mobiles), et il existe un lien uv dans E si et seulement si les robots u et v ont un point de rendez-vous en commun. Pour établir l'ordonnancement des déplacements des robots, nous choisissons une numérotation pour les points de rendez-vous, c'est-à-dire pour les liens du graphe. Pour notre exemple, nous choisissons la numérotation présentée dans la figure 2 qui produit un comportement intéressant. Néanmoins toute autre numérotation peut être choisie arbitrairement.

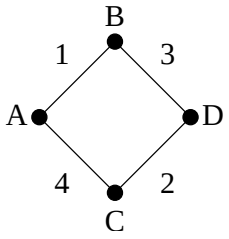

FIG. 2: Exemple de numérotation des points de rendez-vous

Étant donné un graphe $\mathcal{G}_{rdv} = (R, E)$ et une numérotation sur E, nous utilisons un *algorithme d'ordonnancement* pour construire, pour chaque robot r_k, un automate local qui représente le parcours de ses points rendez-vous. Chaque automate, $\mathcal{A}_k$, a un ensemble d'états Q_k (points de rendez-vous), de transitions T_k (mouvement entre points de rendez-vous) et un état initial $i_k \in Q_k$ (position initiale). Les étiquettes des états sont les numéros correspondant aux arcs du graphe $\mathcal{G}_{rdv}$.

L'algorithme d'ordonnancement pour un robot r_k est le suivant :

- *il existe, dans l'automate $\mathcal{A}_k$, un état q_i pour chaque arc numéroté q_i du robot r_k ;*
- *il existe une transition de q_i à q_j*
  - *si $q_i < q_j$ et s'il n'existe pas d'état $q \in Q_k$ tel que $q_i < q < q_j$;*
  - *si $q_i = max(Q_k)$ et $q_j = min(Q_k)$;*
- *l'état initial de l'automate est $min(Q_k)$.*

La figure 3 représente les automates $\mathcal{A}_k$ obtenus à partir de la numérotation de la figure 4.

Ces automates produisent pour chaque robot un parcours itératif de leurs deux points rendez-vous. Au départ, les robots des zones A et B sont au point de rendez-vous numéroté 1 et les robots des zones C et D au point de rendez-vous 2. Ces déplacements sont représentés dans la figure 4. L'état initial des robots est indiqué par les flèches en gras.

4 Propriétés de l'ordonnancement

Maintenant nous devons prouver que, quel que soit le réseau de départ, les déplacements produits par l'algorithme d'ordonnancement respectent strictement les propriétés que nous attendons. Dans ce but nous construisons, à partir des automates locaux, leur automate *produit* pour représenter l'état global du réseau à tout instant. Cet automate est utilisé pour prouver les propriétés intéressantes de notre ordonnancement.

Étant donné l'ensemble des automates locaux $\mathcal{A}_k$, l'automate produit est construit comme suit :

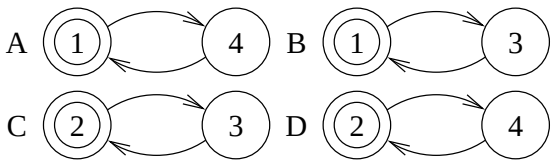

FIG. 3: Les automates locaux des déplacements des robots

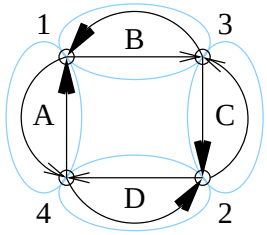

FIG. 4: Exemple des déplacements des robots

- *chaque état est constitué d'un n-uplet d'états* $(q_1, ..., q_n) \in Q_1 * ... * Q_n$ *;*
- *l'état initial de l'automate produit est l'état constitué de tous les états initiaux des robots* $(i_1, ..., i_n)$ *;*
- *il existe une transition entre deux états* $Q = (q_1, \ldots, q_n)$ *et* $Q' = (q'_1, \ldots, q'_n)$ *de l'automate produit si et seulement si l'état Q et l'état Q' ne diffèrent que sur les états de deux automates locaux* $\mathcal{A}_i$ *et* $\mathcal{A}_j$*, que dans Q,* $q_i \equiv q_j$[†] *et que dans les automates* $\mathcal{A}_i$ *et* $\mathcal{A}_j$ *il existe, respectivement, une transition de l'état* q_i *à* q'_i *et une transition entre* q_j *et* q'_j*. On dit que la transition se fait sur les automates* $\mathcal{A}_i$ *et* $\mathcal{A}_j$*.*

Afin de réduire au minimum le nombre de transitions, nous avons choisi de ne pas représenter les transitions dans lesquelles plus de deux automates locaux changent d'état. Pourtant, dans la réalité, il est possible que deux couples de robots soient en même temps à leur point de rendez-vous et qu'ils changent simultanément d'état. Toutefois, l'automate produit étant une abstraction du système, il n'y a pas de notion de temps absolu associée aux transitions et il est donc toujours possible de considérer que deux transitions sur des points de rendez-vous distincts sont ordonnées.

L'automate produit construit à partir des automates locaux de la figure 3 est décrit par la figure 5. Dans cet automate nous avons omis de représenter les états qui ne sont pas accessibles à partir de l'état initial. En effet, notre automate produit devrait contenir $2^4 = 16$ états alors qu'il n'en contient que 6.

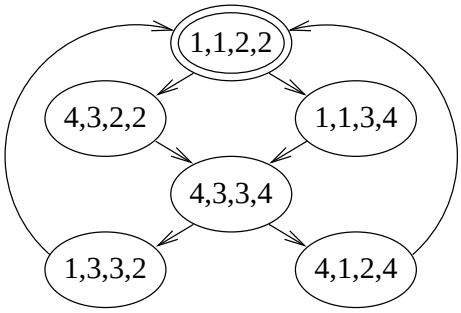

FIG. 5: Exemple de l'automate produit

Dans cet automate tout état a un successeur, ce qui prouve que le système n'est jamais bloqué s'il n'y a pas de robot qui tombe en panne. D'autre part, quel que soit le chemin suivi dans l'automate produit, les robots parcourent tous leurs points de rendez-vous après un nombre borné de transitions. Si ces propriétés sont évidentes sur cet exemple, elles ne le sont pas dans le cas général. En particulier, dans la figure 5, nous montrons seulement les états de l'automate produit accessibles à partir de l'état initial. Dans les états non accessibles ces propriétés ne sont pas vérifiées.

Proposition 1 (Blocage global) *Quelle que soit la numérotation choisie pour les points rendez-vous, si les déplacements des robots sont construits en respectant l'algorithme d'ordonnancement, le système n'est jamais bloqué.*

[†] Ils ont le même numéro.

Preuve Pour prouver qu'à tout instant le système n'est pas bloqué, il suffit de prouver que tout état de l'automate produit atteint à partir de l'état initial a un successeur.

Tout d'abord, l'état initial de l'automate produit a un successeur. En effet, par construction, dans cet état tous les automates locaux sont dans leur état initial. Par construction deux d'entre eux sont sur un état numéroté 1. Donc, il existe au moins un successeur pour l'état initial de l'automate produit qui consiste à effectuer la transition sur les deux automates locaux dont l'état est numéroté 1.

Maintenant nous devons prouver que, quelque soit l'état de l'automate produit atteint à partir de cet état initial, cet état a un successeur.

Pour cela nous prouvons que tout état atteint à partir de l'état initial est un état initial pour une autre numérotation produisant les «mêmes» automates (modulo les états initiaux et les labels). Cet état a donc au moins un successeur. Par récurrence on prouve que tout état atteint à partir de l'état initial est un état initial pour une autre numérotation et donc qu'il a au moins un successeur.

Dans l'état initial, supposons que la transition t choisie se fasse sur les automates locaux $\mathcal{A}_i$ et $\mathcal{A}_j$. Ces deux automates locaux sont donc, par définition, dans des états ayant un même numéro qui est le plus petit numéro p de leurs états. Soit m le nombre total de points de rendez-vous. Appliquons à nos automates la renumérotation suivante :

- le numéro du point de rendez-vous p devient m ;
- si l'état q a le numéro k, et k est strictement supérieur à p, alors le numéro du point de rendez-vous k devient $k-1$;
- dans tous les autres cas le numéro de l'état q ne change pas.

Tout d'abord, nous nous intéressons aux automates locaux différents de $\mathcal{A}_i$ et $\mathcal{A}_j$ obtenus après renumérotation.

Sachant que les automates locaux $\mathcal{A}_i$ et $\mathcal{A}_j$ étaient les seuls à avoir un état avec le numéro p (restriction 6) et que cette nouvelle numérotation préserve l'ordre des numéros des autres états, ces automates sont les mêmes que ceux qui auraient été obtenus par l'algorithme de constructions des automates locaux pour cette nouvelle numération.

Comparons maintenant les automates $\mathcal{A}_i$ et $\mathcal{A}_j$ après renumérotation et ceux qui auraient été obtenus par l'algorithme de construction des automates locaux (l'algorithme d'ordonnancement) pour cette numérotation. L'ordre de parcours des états est respecté de façon «circulaire». Seul l'état initial est distinct. En effet, l'état initial dans l'automate construit est le successeur de p puisqu'il a le plus petit numéro.

L'automate produit obtenu après renumérotation est donc le même que celui construit par l'algorithme d'ordonnancement pour cette nouvelle numérotation, excepté l'état initial.

L'état initial de ce dernier automate est l'état atteint depuis l'état initial après une transition par t. En effet, en transitant par t dans l'automate produit, tous les automates distincts de $\mathcal{A}_i$ et $\mathcal{A}_j$ restent dans leur état initial et, $\mathcal{A}_i$ et $\mathcal{A}_j$ transitent dans leurs états de plus petit numéro dans la nouvelle numérotation qui sont, par construction, leurs nouveaux états initiaux. Ainsi, après une transition par t tous les automates sont dans leurs états initiaux pour cet nouvelle numérotation. □

La propriété que nous venons de prouver est une propriété globale. Elle nous assure qu'il y a toujours un robot qui «bouge» dans le système. En revanche, elle ne prouve pas que tous les robots bougent au bout d'un temps borné. Pourtant, pour assurer la communication entre deux robots en un temps borné nous avons besoin de cette propriété.

Proposition 2 (Blocage local) *Quelle que soit la suite de transitions dans l'automate produit, tous les automates locaux changent d'état au bout d'un nombre borné de transitions de l'automate produit.*

Preuve Tout d'abord, l'automate produit étant fini, toute suite de transitions dans l'automate produit finit par retomber sur un état déjà visité au bout d'un nombre borné de transitions. Sur le cycle (non vide) ainsi formé, les états des automates locaux étant égaux au début et à la fin du cycle, soit les automates locaux sont restés dans le même état, soit tous leurs états ont été visités au moins une fois.

Montrons maintenant que si un automate local a plus d'un état alors il a changé d'état et donc qu'il a parcouru tous ses états. Supposons qu'il n'ait pas changé d'état, alors tous les autres automates avec lesquels il a des points de rendez-vous en commun n'ont pas bougés. En effet, sinon tous leurs états auraient été visités et en particulier celui-ci, ce qui est contradictoire. Comme par hypothèse le réseau est connexe

(restriction 7), par transitivité, aucun des robots n'a bougé, ce qui n'est pas possible sur un cycle non vide puisque au moins deux automates ont changé d'état. □

De cette propriété nous pouvons déduire que quelque soit l'état courant du système tout point de rendez-vous sera atteint au bout d'un nombre borné de transitions dans l'automate produit et donc que toute suite de points de rendez-vous sera parcourue en un temps borné.

Corollaire 1 *Tout message émis par un robot à destination d'un autre robot du réseau atteindra sa destination en un temps borné, s'il existe une suite de points de rendez-vous les reliant.*

De la démonstration précédente nous pouvons également déduire une propriété sur les cycles de l'automate produit.

Proposition 3 *Les cycles de l'automate produit ont une longueur multiple du nombre total de points de rendez-vous.*

Preuve Lors de la démonstration de la proposition 2 nous nous sommes appuyés sur le fait que si un robot ne bougeait pas alors tous les autres ne bougeaient pas non plus pour des raisons de transitivité. De même, si sur un cycle un robot parcourt tous ses points de rendez-vous n fois, alors les robots avec lesquels il a des points de rendez-vous doivent également parcourir n fois leurs points de rendez-vous. Grâce à la connexité du graphe on obtient que sur un cycle tous les robots parcourent tous leurs points de rendez-vous le même nombre de fois, ce qui prouve la propriété. □

Dans chaque réseau, quelque soit la numérotation, il existe au moins un cycle de longueur égale au nombre de points de rendez-vous, celui où les points des rendez-vous sont parcourus dans l'ordre.

D'autre part, nous avons trouvé des réseaux et des numérotations pour lesquelles il existe des cycles élémentaires de longueur supérieure au nombre de points de rendez-vous.

5 Routage dans le réseau

Maintenant que nous avons produit un ordonnancement des déplacements des robots qui nous assure que tout message peut être acheminé d'un robot à un autre en temps borné, nous nous intéressons au routage des messages dans ce réseau. Nous avons choisi de construire des tables de routage pour chacun des robots leur permettant d'acheminer les messages vers tous les autres robots du réseau.

La table de routage d'un robot doit contenir, pour être la plus précise possible, une entrée pour tous les couples point de rendez-vous du robot, destination. En effet, le chemin le plus plus court vers une destination peut dépendre de l'endroit où se trouve le robot au moment où il décide d'émettre le message. Une entrée de la table de routage donne le point de rendez-vous sur lequel le robot doit transmettre son message pour qu'ils atteignent le plus «rapidement» possible leur destination.

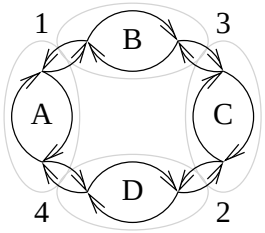

FIG. 6: Le graphe de routage

Les tables de routage des robots sont pré-calculées (comme l'ordonnancement) avant d'être placées sur chacun des robots. Pour effectuer ce calcul nous construisons un graphe à partir des ordonnancements. Dans ce graphe nous ajoutons des arcs pour prendre en compte le temps d'attente et/ou la communication au point de rendez-vous. Chacun des arcs est pondéré afin de rendre compte des déplacements et des tâches accomplies par chacun des robots.

Pour notre exemple, le graphe produit est celui présenté dans la figure 6.

Pour trouver le plus court chemin nous utilisons un algorithme classique de plus court chemin [Dij59]. Cet algorithme construit le plus court chemin entre deux points de rendez-vous de deux robots. Nous gardons ensuite, pour construire les tables de routage, le plus court chemin parmi les points de rendez-vous de chaque robot destination. En effet, nous nous intéressons au plus court chemin vers un robot, c'est-à-dire vers n'importe lequel de ses points de rendez-vous.

Par exemple, si l'on considère que tous les arcs du graphe ont le même coût, la table de routage pour le robot de la zone A est le suivante :

Source	Dest.	RDV
A.1	B|C	A.1
A.1	D	A.4
A.4	B	A.1
A.4	C|D	A.4

Dans cette table de routage, on voit que si le robot de la zone A veut envoyer un message au robot de la zone C, il choisira de l'envoyer *via* la zone B s'il va au point de rendez-vous 1 et *via* la zone D s'il va au point de rendez-vous 4.

6 Implantation

Nous avons testé la faisabilité de notre algorithme d'ordonnancement sur de vrais robots. Pour cela, nous avons utilisé un réseau de quatre robots Lego MindStorm présentés dans la figure 7. Il est très difficile de contrôler précisément le déplacement des robots avec les capteurs existants. C'est pourquoi que nous avons utilisé une piste avec des lignes noires qui sont suivies par les robots à l'aide des capteurs de lumière. Les robots ont des moteurs et des capteurs pour effectuer leurs déplacements. Ils ont une interface infrarouge leur permettant de communiquer avec un autre robot ou avec une base. Pour effectuer les calculs, ils disposent d'un micro-controleur et de 32 Kb de RAM. Malgré ces ressources limitées nous n'avons pas eu de problèmes à implanter la communication et le routage simple dans le réseau présenté.

FIG. 7: Expérience avec des robots

7 Discussion et conclusion

Nous avons montré dans cet article qu'il était possible, dans un réseau de robots, de trouver un ordonnancement et une stratégie pour les déplacements des robots qui assure que les messages seront acheminés dans un temps borné. Toutefois, il nous reste encore beaucoup de points à explorer.

Tout d'abord, la pondération du graphe que nous utilisons pour trouver le plus court chemin est arbitraire. Elle peut être utilisée afin de favoriser certains chemins plutôt que d'autres. Par exemple, si l'on désire minimiser le nombre de communications pour favoriser la confidentialité, on placera un poids important sur les arcs du graphe correspondant à des points de rendez-vous. En revanche, si l'on désire minimiser l'énergie, au placera *a priori* un poids faible sur les arcs correspondant aux points de rendez-vous.

Le temps d'attente à un point de rendez-vous est lié à l'ordonnancement des déplacements et à leur durée. Si les déplacements des robots entre deux points de rendez-vous ont des durées connues, il doit être possible

de prédire, au moins pour un état stable, le temps d'attente de chaque robot au point de rendez-vous. En particulier, un robot arrivant toujours avant un autre sur le point de rendez-vous, un des temps d'attente doit toujours être nul. Ces informations pourrait-être utilisées pour effectuer la pondération du graphe afin d'obtenir des plus courts chemin en temps.

Le problème principal de notre algorithme est qu'il n'est pas tolérant aux pannes. Si un robot tombe en panne, tous les autres robots restent bloqués dans un point de rendez-vous. Pour résoudre ce problème, il nous faudra nécessairement introduire des délais d'attente maximale sur les points de rendez-vous.

Le choix de l'ordre de la numérotation des points de rendez-vous influence le comportement global du réseau de robots et l'acheminement de messages. Certains ordonnancements favorisent le parallélisme, c'est-à-dire la possibilité que plusieurs robots se déplacent en même temps. D'autres numérotations peuvent minimiser les déplacements dans le plan.

Un autre aspect que nous envisageons d'étudier est la possibilité de vérifier nos propriétés en utilisant des outils de preuve formelle. Nous nous sommes déjà penché sur la vérification de tout réseau donné au moyen du *model-checker* SPIN [Hol97].

Remerciements

Merci à Laurent Viennot, Marc Zipstein, Jean Berstel et Rémi Forax pour les discussions fructueuses que nous avons pu avoir avec chacun d'eux.

Références

[BMJ+98] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.

[Bra01] P. Bracka. Routage dans le réseau ad-hoc des robots. Memoire de dea, IGM - Université de Marne-la-Vallée, September 2001.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[Hol97] G. J. Holzmann. The model checker spin. *IEEE Transaction on Software Engineering*, 23(5) :279–295, May 1997.

[JM96] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[Joh94] D. Johnson. Routing in ad hoc networks of mobile hosts. In *Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, U.S., 1994.

[LR00] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, pages 44–55, Boston, August 2000. ACM Press.

[PC97] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks, 1997.

[Per01] C. Perkins. Ad hoc on-demand distance-vector routing, 2001.

[RT99] E. Royer and C. Toh. A review of current routing protocols for ad-hoc mobile wireless networks, 1999.