PARCOURIR L'INFINI AVEC DES ROBOTS

D'intéressants problèmes algorithmiques se posent quand on cherche à piloter des robots pour explorer une grille infinie.

LES AUTEURS

JEAN-PAUL DELAHAYE
professeur émérite à l'université de Lille et chercheur au laboratoire Cristal (Centre de recherche en informatique, signal et automatique de Lille)

PHILIPPE MATHIEU
directeur de l'équipe Systèmes multi-agents et comportements (Smac) du Centre de recherche en informatique, signal et automatique de Lille (Cristal)

L'informatique enrichit aujourd'hui les mathématiques de nouvelles familles de problèmes : théorie du calcul et de l'information, graphes, réseaux, etc. Parmi ces problèmes : la programmation de robots. Comment gérer leurs déplacements et les coordonner pour atteindre des buts précis ? Certains de ces problèmes sont impossibles à résoudre – il faut alors le démontrer –, d'autres ne peuvent être résolus qu'à l'aide d'un nombre minimum de robots, qu'on cherchera à connaître. Tout un nouvel ensemble de questions est né de ces dispositifs particuliers. Pour en donner une idée, nous allons considérer un problème simple en apparence, et indiquer quelques résultats positifs – des algorithmes – et négatifs – des affirmations d'impossibilité.

Nous nous concentrerons sur le problème du parcours complet d'une grille carrée infinie, sur laquelle un ou plusieurs robots se déplacent, parfois se rencontrent, et tentent de s'organiser. Chaque case de la grille a pour coordonnées un couple d'entiers positifs ou négatifs, par exemple $(0,0)$, $(3,-1)$, $(-4,10)$. Le problème consiste donc à donner des instructions aux robots afin que, collectivement, ils visitent toutes les cases de la grille sans en oublier une seule.

QUELS ROBOTS ?

Les robots que nous envisageons se déplacent d'une case à la fois. Quand un robot se déplace, il passe à l'une des quatre cases directement voisines. On suppose que les robots n'utilisent pas le hasard pour déterminer leurs mouvements. On suppose aussi que le temps est discret – il y a un instant 0, un instant 1, etc. – et que les robots sont synchronisés, c'est-à-dire qu'ils ont accès à une même horloge. Enfin, on suppose que chaque robot possède une orientation qui définit sa direction de déplacement, qu'il peut changer en pivotant quand c'est nécessaire, et que chaque robot sait ce que veut dire « à droite » et « à gauche ».

Nous ferons diverses hypothèses sur les outils dont disposent les robots pour se localiser, et notamment sur leur mémoire et les moyens dont ils disposent pour interagir ou communiquer quand ils sont plusieurs. En attribuant beaucoup de moyens aux robots, le problème est facile à résoudre. C'est en envisageant des robots simples, aussi démunis que possible, que cela devient intéressant.

Imaginons d'abord un unique robot muni d'une sorte de GPS, qui lui indique à chaque instant les coordonnées de la case de la grille où il se trouve. Il est facile, en utilisant les informations du GPS, de programmer les déplacements du robot pour qu'il parcoure la grille infinie en passant par chaque case au moins une fois.

Cette méthode admet un grand nombre de variantes. On numérote toutes les cases du plan sans répétition : case n° 0, case n° 1, case n° 2, etc. Par exemple, cela peut être la numérotation en spirale carrée de l'encadré 1. On programmera alors le robot pour lui commander d'aller à la case n° 0, puis – éventuellement en plusieurs

Jean-Paul Delahaye a récemment publié : **Au-delà du Bitcoin** (Dunod, 2022).

étapes – de rejoindre le plus directement possible la case n° 1, etc. Grâce à son GPS, le robot ira sans mal d'une case à l'autre, et, connaissant la numérotation, on est certain qu'il n'oubliera aucune case. Trouver le moyen le plus efficace d'aller de la case numéro n à la case numéro $n+1$ est facile.

Si la numérotation des cases est telle que la case numéro $n+1$ est toujours à côté de la case numéro n, le robot ne repasse jamais deux fois par la même case, et son parcours est alors aussi économique que possible. C'est ce qui se passe avec la numérotation en spirale carrée présentée dans l'encadré 1. Si ce n'est pas le cas, ce n'est pas grave : le robot passera plusieurs fois par les mêmes cases, mais son algorithme de déplacement lui assurera quand même de parcourir toute la grille. Bien sûr, l'exécution du plan de visites – case n° 0, puis case n° 1, puis case n° 2, etc. – peut exiger des calculs plus ou moins complexes, mais on comprend bien qu'il n'existe aucun obstacle fondamental à sa réussite.

TROIS MÉMOIRES CONTRE UN GPS

L'utilisation du GPS est en réalité inutile, et ce sera la deuxième version du problème. S'il fait bien attention à ses déplacements, un robot ne disposant pas de GPS peut de lui-même se repérer dans le plan, grâce à sa position et son orientation initiales. En effet, puisque le robot possède une orientation et connaît sa droite et sa gauche, des instructions comme « avancer », « pivoter vers la droite », « pivoter vers la gauche » ou « faire demi-tour » ont un sens pour lui, relativement à son orientation instantanée. On s'en servira pour écrire des programmes.

On aimerait minimiser les données que le robot doit mémoriser pour s'assurer qu'il visite toutes les cases de la grille. Le second problème consistera à programmer un robot dépourvu de GPS et ne disposant que d'une mémoire de trois nombres entiers pour qu'il parcoure la grille dans sa totalité.

La solution se fonde sur l'observation attentive de la spirale carrée. Sur le dessin, on voit que les étapes du parcours sont :

- avancer, pivoter vers la droite, avancer, pivoter vers la droite ;

puis :

- avancer 2 fois, pivoter vers la droite, avancer 2 fois, pivoter vers la droite ;

puis :

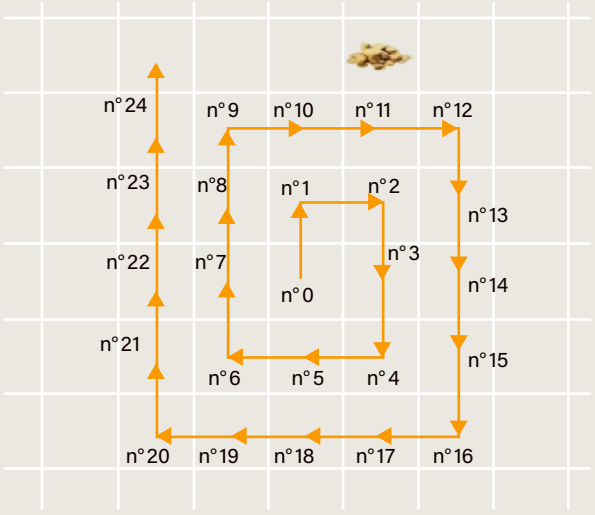
- avancer 3 fois, pivoter vers la droite, avancer 3 fois, pivoter vers la droite ;

etc.

Le parcours est donc composé de deux segments de longueur 1, puis deux segments de longueur 2, puis deux segments de longueur 3, etc., chacun suivi d'une rotation d'un

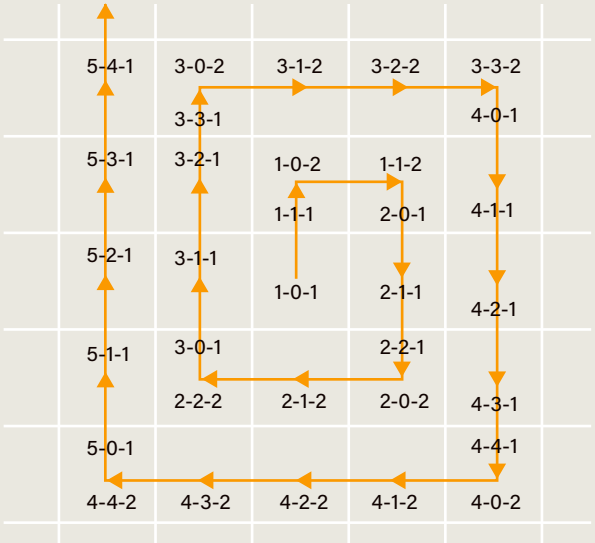
1 INÉVITABLES SPIRALES CARRÉES

Un trésor est caché dans l'une des cases d'une grille infinie. Pour trouver le trésor, on cherche à faire parcourir entièrement la grille à un robot mobile sachant se repérer grâce à un GPS. Si on dispose d'une numérotation des cases de la grille infinie, on demande au robot d'aller d'abord à la case n° 0, puis à la case n° 1, etc. Lorsque les cases de numéros successifs sont toujours côte à côte, le robot ne passera qu'une seule fois dans chaque case. La numérotation en spirale carrée s'impose.



2 LE ROBOT À TROIS VARIABLES

Si le robot ne possède pas de moyen lui permettant de savoir où il est, parcourir toutes les cases de la spirale carrée est moins facile. Il faut que le robot utilise un moyen pour se repérer uniquement en mémorisant ce qu'il fait. Pour cela l'utilisation de trois variables *A*, *B* et *C* est le moyen le plus simple. La variable *A* mémorise la longueur de la branche de la spirale où se situe le robot. La variable *B* mémorise où le robot en est du segment de spirale de longueur *A* qu'il parcourt. La variable *C* mémorise si le robot est sur le premier segment de longueur *A* qu'il doit parcourir (*C* = 1) ou sur le second (*C* = 2). Voir dans le texte principal le programme qui gère les trois variables et qui indique au robot quand il doit avancer et quand il doit pivoter. L'inconvénient de cette méthode est qu'elle exige du robot qu'il possède une mémoire infinie pour stocker les deux variables *A* et *B*, qui prendront des valeurs de plus en plus grandes.

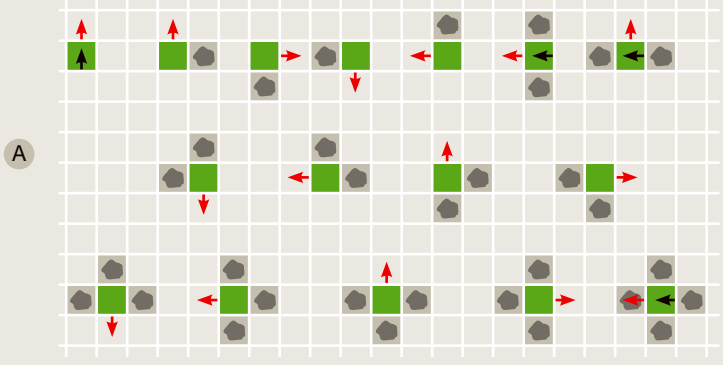


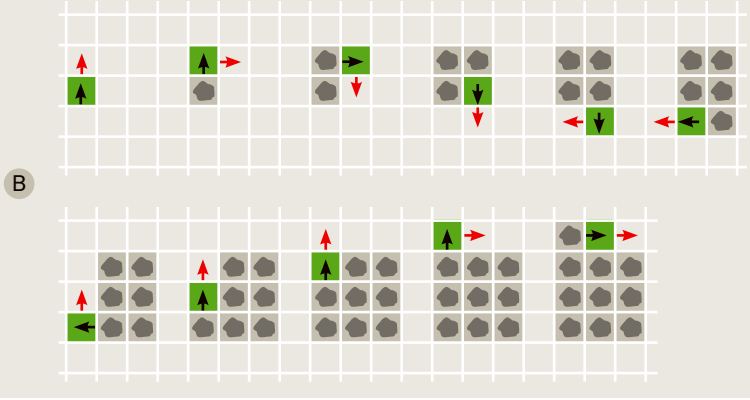
quart de tour vers la droite. On en déduit un programme qui indique au robot ce qu'il doit faire, et qui n'utilise que trois variables entières, qu'on notera A, B et C.

La variable A sert à mémoriser où le robot en est des longueurs de segments à parcourir au fur et à mesure que le robot s'éloigne du centre vers l'infini (longueur 1, longueur 2, longueur 3, etc.). La variable B sert à mémoriser où en est le robot des A déplacements qu'il doit faire pour aller d'un bout à l'autre des segments de longueur A qu'il parcourt. La variable C prend alternativement les valeurs 1 et 2, pour mémoriser si le robot est dans le premier ou le second segment de longueur A.

3 LE ROBOT PETIT POUCET

Un robot qui peut laisser des traces derrière lui – par exemple, qui peut déposer un caillou dans chaque case visitée –, même s'il ne dispose que d'une mémoire bornée, peut parcourir la spirale carrée. En constatant la présence ou l'absence de cailloux dans les quatre cases qui l'entourent, le robot décide où aller. Les règles de déplacement qu'il doit utiliser sont indiquées sur les dessins de la figure *(A)*. La flèche rouge indique dans quelle case le robot choisit d'aller. Souvent cette case est indépendante de l'orientation du robot. Quand ce n'est pas le cas une flèche noire indique l'orientation du robot.
Dans les dessins de la figure *(B)*, on a représenté ce que fait un robot placé sur une grille vide : il emprunte la spirale carrée, ce qui lui permet de parcourir l'ensemble de la grille. La flèche rouge indique où il choisit d'aller, la flèche bleue son orientation. Précisons qu'à chaque fois, le robot pivote sur lui-même si c'est nécessaire avant de se déplacer, et qu'à chaque fois qu'il quitte une case qui ne contient pas de trace, il en laisse une.





Le robot est posé sur une case quelconque de la grille avec les trois données $A=1$, $B=0$, $C=1$, qui constituent l'initialisation de son programme. Ensuite à chaque étape le programme du robot exécute l'une des trois lignes de commande suivantes, selon les valeurs de A, B et C:

- si $B<A$, alors avancer et ajouter 1 à B;
- si $B=A$ et $C=1$, alors pivoter vers la droite, attribuer la valeur 0 à B , attribuer la valeur 2 à C;
- si $B=A$ et $C=2$, alors pivoter vers la droite, ajouter 1 à A, attribuer la valeur 0 à B , attribuer la valeur 1 à C .

Voilà ci-dessous ce que donne le début de l'exécution du programme. On indique à chaque étape les valeurs de A, B et C sous la forme A-B-C. On écrit «pivoter» à la place de «pivoter vers la droite».

1-0-1, avancer, 1-1-1, pivoter, 1-0-2, avancer, 1-1-2, pivoter, 2-0-1, avancer, 2-1-1, avancer, 2-2-1, pivoter, 2-0-2, avancer, 2-1-2, avancer, 2-2-2, pivoter, 3-0-1, avancer, 3-1-1, avancer 3-2-1, avancer, 3-3-1, pivoter, 3-0-2, avancer, 3-1-2, avancer, 3-2-2, avancer, 3-3-2, pivoter, 4-0-1, etc. (*voir l'encadré 2*).

UNE SEULE MÉMOIRE ?

Comme troisième version du problème, on voudrait que le robot n'utilise la mémoire que d'un seul nombre entier. C'est assez facile, car on peut mémoriser trois entiers en un seul : on remplace A, B, et C par $D=2^A3^B5^C$. La décomposition d'un nombre entier en facteurs premiers étant unique, connaître D permet de connaître A, B, et C.

Une fois que l'on a ramené le problème à un seul nombre D, les opérations qu'on faisait sur A, B et C se traduisent en opérations sur D. Par exemple l'opération «ajouter 1 à A» devient «multiplier D par 2». La fonction décrite plus haut, qui fait évoluer les trois variables et détermine le mouvement du robot, est maintenant plus compliquée, mais se traduit en une fonction qui fait évoluer D. C'est une fonction arithmétique parfaitement bien définie, et si le robot dispose d'un processeur qui la calcule, il pourra parcourir la spirale carrée sans rien mémoriser d'autre, entre deux mouvements successifs, que le nombre D.

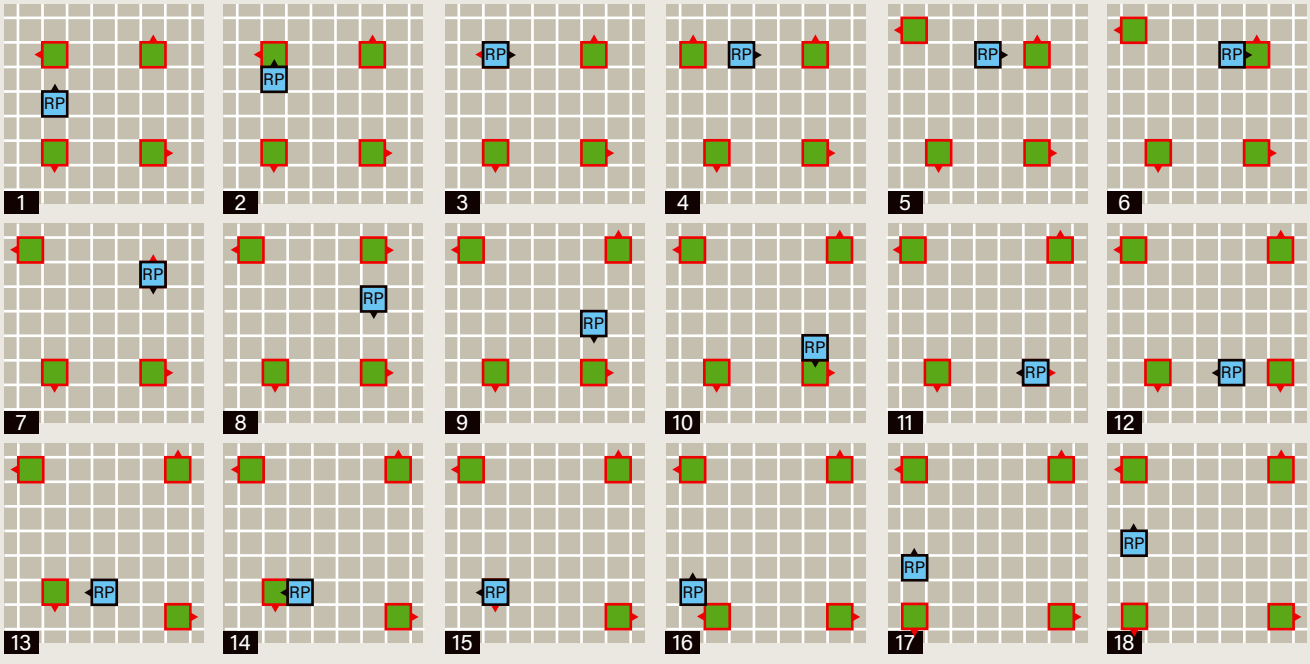
Nous ne donnons pas le détail de l'opération faisant passer d'une valeur de D à la suivante car, quoi qu'il en soit, qu'on parle de trois mémoires pouvant contenir des nombres entiers ou d'une seule, ce n'est pas technologiquement réaliste : stocker des nombres entiers de longueur illimitée demanderait une mémoire informatique potentiellement infinie… ce qui n'existe pas ! Au passage, notons qu'un GPS tel qu'on l'a envisagé est aussi technologiquement impossible, car il lui faudrait également une capacité infinie de mémoire.

4 CINQ ROBOTS À MÉMOIRE BORNÉE

Pour la quatrième version du problème, considérons que le robot ne dispose que d'une mémoire bornée, mais qu'il peut laisser une trace là où il passe, comme le Petit Poucet semant des cailloux derrière lui. Notre robot ne peut laisser qu'une seule trace par case, mais bien sûr il peut voir les traces qu'il a laissées. On peut alors fixer des règles n'exigeant du robot Petit Poucet qu'une mémoire bornée, et l'amenant à parcourir la spirale carrée jusqu'à l'infini.

Pour savoir où aller, le robot visite les quatre cases autour de lui et regarde celles qui contiennent une trace qu'il a laissée. Il revient alors à sa place, dans son orientation initiale. C'est pour cette phase qu'il a besoin d'un peu de mémoire, qu'il pourra effacer dès qu'il saura quel mouvement faire. Une hypothèse équivalente serait que le robot voit les 4 cases autour de lui et saurait donc, sans se déplacer, si elles contiennent des traces laissées par ses soins. En fonction de ce qu'il a trouvé ou vu, il décide alors de se déplacer dans une des quatre cases voisines de sa position. Pour faire le mouvement choisi, il commence par pivoter pour s'orienter vers la case de destination, puis se déplace en laissant une marque dans la case qu'il quitte si elle n'en

Ici, l'idée est encore de passer par tous les points de la grille en parcourant la spirale carrée. Le robot principal, RP, effectuera ce travail. Il utilisera une mémoire bornée, et s'appuiera sur quatre robots dont les positions lui serviront de balises pour délimiter le carré qu'il va parcourir et qui, petit à petit, s'agrandira. Plusieurs robots peuvent se trouver sur une même case. Quand c'est le cas, ils le savent et agissent donc d'une façon particulière – c'est même leur seule possibilité de communiquer entre eux. Selon leur position au nord-ouest, nord-est, sud-est, sud-ouest, les robots-balises se nomment R-NO, R-NE, R-SE et R-SO. Les robots-balises sont le plus souvent immobiles. Quand un robot-balise rencontre RP, cela déclenche un déplacement du robot-balise. Par exemple dans le cas du robot R-NO, un double mouvement le fait aller vers la case en diagonale au nord-ouest de la case où la rencontre s'est faite (*voir les images 4 et 5*). Le robot R-NE, lui, ira occuper la case en diagonale au nord-est, etc. À chaque fois qu'il rencontre un robot-balise, le robot RP pivote vers la droite. Sinon à chaque étape, RP avance d'une case. Il a un comportement légèrement différent quand il rencontre R-SO : au lieu de simplement pivoter vers la droite, il avance d'une case, et seulement après ce pas supplémentaire vers l'ouest il pivote (*voir les images 15, 16 et 17*). C'est ce comportement particulier qui permet d'agrandir le carré parcouru, et donc le cheminement général en spirale carrée jusqu'à l'infini. Pour effectuer cette chorégraphie, chacun des cinq robots n'utilise qu'un calcul simple, n'exigeant que peu de mémoire. Par exemple quand le robot R-NO doit bouger, il a besoin de savoir où est le nord-ouest, ce qu'il peut savoir en se basant sur son orientation. Si on veut qu'après chaque rencontre, il se réoriente vers l'ouest, il doit faire quatre mouvements élémentaires, qu'il peut grouper deux par deux : avancer, pivoter vers la droite, avancer, pivoter vers la gauche. Il lui faut donc un peu de mémoire pour savoir où il en est de ces mouvements. Dans les schémas, nous faisons l'hypothèse que les robots se déplacent puis se repositionnent en pivotant si nécessaire. Les dessins représentent les robots après leur repositionnement. La position de départ consiste à placer les quatre robots-balises les uns contre les autres en carré, avec RP dans la même case que R-NO. Le robot RP n'a, lui, besoin d'aucune mémoire, sauf quand il rencontre R-SO. Au total, nous avons un ensemble de cinq robots ayant chacun un programme de fonctionnement simple et n'exigeant presque pas de mémoire, qui fait en sorte que RP parcoure toutes les cases de la grille infinie.



contient pas déjà une. Le choix de la case de destination se fait selon la règle suivante (*voir l'encadré 3*) :

- si aucune trace n'est trouvée dans les cases voisines, il avance dans la direction définie par son orientation;
- si une seule trace est trouvée, il va dans la case voisine de la sienne, qui le fait tourner dans le sens des aiguilles d'une montre autour de la case où il a vu la marque;
- si deux cases contiennent une trace et qu'elles sont diamétralement opposées, il va devant lui si la case devant lui n'est pas marquée, et sinon il va à droite de la direction définie par son orientation;
- si deux cases contiennent des traces et qu'elles ne sont pas diamétralement opposées, il va dans celle des deux cases voisines et non marquées qui le fera le plus tourner dans le sens des aiguilles d'une montre. Autrement dit : entre « devant » et « droite », il choisit « droite »; entre « droite » et « derrière », il choisit « derrière »; entre « derrière » et « gauche », il choisit « gauche »; entre « gauche » et « devant », il choisit « devant »;
- si trois cases contiennent des traces, il va dans celle qui n'en contient pas – éventuellement après avoir pivoté;
- si les quatre voisines contiennent des traces, il avance dans la direction définie par son orientation.

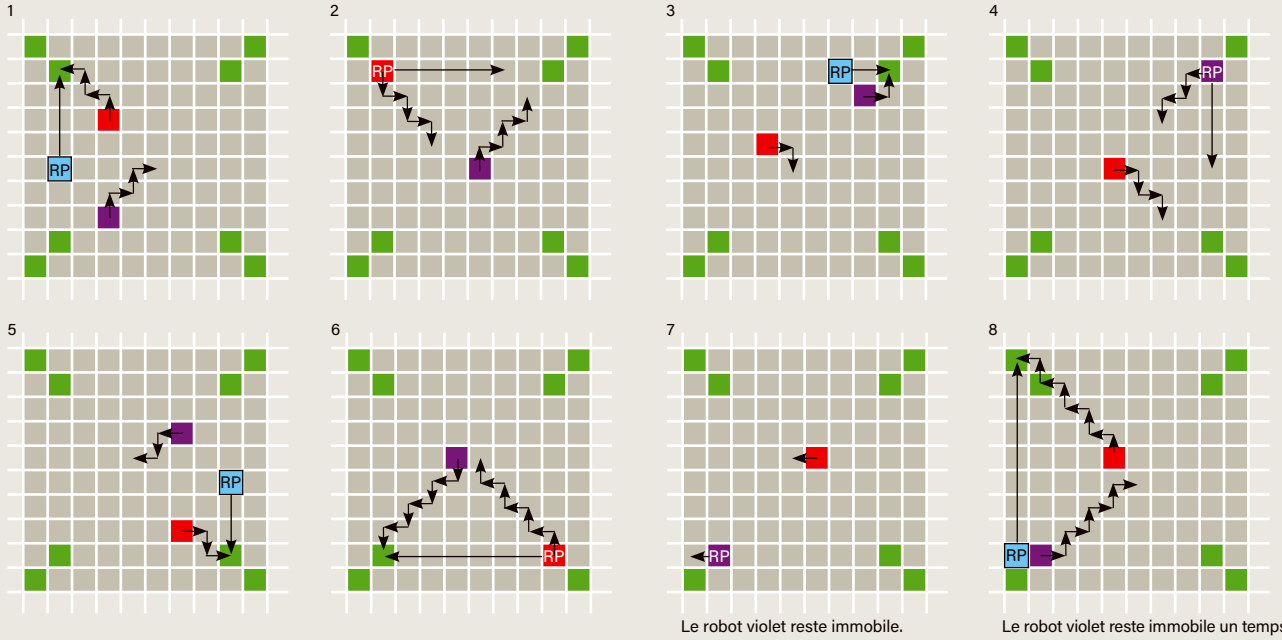
Chose amusante : si on pose le robot sur une grille où se trouve déjà un nombre fini de traces, il ne va pas passer par toutes les cases de la grille, mais après une phase assez désordonnée de déplacements il va emprunter une sorte de spirale jusqu'à l'infini ne laissant derrière lui qu'un nombre fini de cases non marquées.

UN OU DEUX ROBOTS À MÉMOIRE BORNÉE

Déposer des marques dans les cases exige de disposer d'une infinité de petits cailloux. Ce n'est pas encore parfaitement réaliste. Supposons donc, à présent, que le ou les

5 TROIS ROBOTS À MÉMOIRE BORNÉE

Par rapport à la solution utilisant cinq robots à mémoire bornée, une astuce permet encore d'économiser deux robots, et donc de ramener la solution à trois robots. L'idée est de rendre les robots-balises dynamiques. Il n'y a plus que deux robots-balises, et chacun tient un double rôle.
Le schéma représente quelques étapes de l'interaction des trois robots. Nombre d'entre elles, intermédiaires, ne sont pas représentées. Les cases vertes sont des repères pour suivre l'explication des mouvements des robots, mais les robots n'utilisent pas ces repères.
Le robot-balise R-NO-SE *(en rouge)*, quand il rencontre le robot principal RP au coin nord-ouest, le fait pivoter vers la droite. Il ne reste pas sur place, mais s'en va alors en diagonale vers le coin sud-est, pour y arriver en même temps que RP et le faire pivoter. Après sa rencontre avec RP au coin sud-est, le robot-balise R-NO-SE repart vers le coin nord-ouest. De même, le robot-balise R-NE-SE *(en violet)* fera sans cesse des allers et retours entre les coins nord-est et sud-ouest du carré parcouru par le robot RP. Comme précédemment, ce carré s'agrandira à chaque tour, car, arrivé au coin sud-ouest, RP avance d'une case supplémentaire vers l'ouest, comme dans la solution à cinq robots.
On remarquera que le robot-balise R-NE-SO, quand il rencontre RP au coin sud-ouest, doit rester immobile pendant deux unités de temps, de sorte que la rencontre avec RP se fasse au bon endroit quand il remonte. Il est facile de vérifier que le comportement des trois robots n'exige, pour chacun d'eux, qu'une mémoire bornée.



Le robot violet reste immobile.

Le robot violet reste immobile un temps.

robots ne peuvent plus laisser de trace dans les cases qu'ils visitent.

La cinquième version du problème consiste donc à faire parcourir toute la grille à un robot qui n'a qu'une mémoire bornée, et ne peut pas marquer les cases par lesquelles il passe. On a l'intuition qu'avec un unique robot à mémoire bornée, parcourir toute la grille est impossible. C'est effectivement le cas. Voici le raisonnement qui démontre cette impossibilité. Il nous a été expliqué par Quentin Bramas, chercheur à l'université de Strasbourg, de même que la généralisation avec deux robots.

Supposons le contraire, c'est-à-dire que le robot parcoure tout. Il existerait deux instants différents, t et $t+t_0$, où sa mémoire et son orientation seraient identiques – car il y a une infinité d'instants, mais un nombre fini de combinaisons mémoire + orientation. Les déplacements du robot entre t et $t+t_0$ seraient donc identiques à ses déplacements entre $t+t_0$ et $t+2t_0$. Les cases visitées entre $t+t_0$ et $t+2t_0$ seraient donc celles visitées entre t et $t+t_0$ auxquelles on applique une translation d'un vecteur PP', où P et P' désignent respectivement les cases occupées aux instants t et $t+t_0$. Il en irait de même entre $t+2t_0$ et $t+3t_0$, entre $t+3t_0$ et $t+4t_0$, etc. L'ensemble des cases par lesquelles le robot passerait serait donc constitué, à l'infini, d'un ensemble fini de cases E (celles visitées entre t et $t+t_0$), et des versions translatées de E par le vecteur PP', par 2 fois ce vecteur, par 3 fois ce vecteur, etc. Un tel ensemble de cases a une largeur bornée – c'est une sorte de bande – et ne peut donc pas correspondre à la totalité de la grille.

Le même type de raisonnement fonctionne aussi pour deux robots à mémoire bornée. En effet, pour qu'ils fassent quelque chose que ne fait pas un robot seul, il faut que les deux robots se rencontrent une infinité de fois – sans quoi ils ne parcourraient qu'un espace inclus dans une espèce de double bande. On trouvera donc une infinité d'instants où les deux robots seront au même endroit et dans le même état, conduisant comme précédemment à la conclusion que l'ensemble des cases qu'ils visitent est une sorte de bande de largeur finie.

Un seul robot à mémoire bornée ne peut pas parcourir toute la grille. Deux tels robots ne le peuvent pas non plus. Mais alors, combien en faut-il? Est-ce qu'un nombre fini de robots de ce type permet l'exploration de toute la grille? Et, le cas échéant, comment les programmer?

Une première réponse, notre cinquième solution, est donnée avec cinq robots à mémoire bornée, programmés pour que l'un d'entre eux décrive la spirale carrée. Quatre robots délimitent les coins d'un carré, dont le cinquième va parcourir les côtés. Leurs rencontres et interactions peuvent s'organiser d'une telle façon que les coins du carré s'écartent progressivement, permettant au robot qui parcourt les côtés du carré de décrire la fameuse spirale (*voir l'encadré 4*).

Des algorithmes d'une subtilité étonnante sont proposés pour le problème du parcours de la grille.

TROIS ROBOTS PEUVENT LE FAIRE

La solution avec cinq robots est assez astucieuse. Pourtant elle a été améliorée et il existe une solution, ce sera notre sixième, avec trois robots à mémoire bornée.

Le robot principal, que nous noterons RP, dessinera des carrés de plus en plus grands, comme dans toutes les solutions précédentes. Il sera aidé par deux robots qui, contrairement à la solution à cinq robots, ne seront pas immobiles la majorité du temps, mais iront d'un coin du carré à celui diamétralement opposé. Le robot R-NO-SE fera des allers et retours incessants en diagonale entre le coin nord-ouest et le coin sud-est. Le robot R-NE-SO fera des allers et retours en diagonale entre le coin nord-est et le coin sud-ouest.

La rencontre de RP avec R-NO-SE dans le coin nord-ouest provoque le changement d'orientation de RP, qui pivote de 90° à droite. Elle fait aussi changer R-NO-SE de sens de parcours de la diagonale : après la rencontre il part vers le coin sud-est, où il rencontrera plus tard RP.

Pour que les carrés parcourus s'agrandissent, la rencontre au coin sud-ouest de RP avec R-NE-SO provoque l'avancée de RP d'une case en plus vers l'ouest avant de pivoter. Il faut aussi l'immobilisation de R-NO-SE pendant deux unités de temps, pour préserver la bonne synchronisation des rencontres quand le carré s'est agrandi. L'encadré 5 montre quelques étapes de cette coordination des trois robots à mémoire bornée.

Nous n'avons donné que quelques exemples des problèmes que les chercheurs explorent avec soin. Ils envisagent des robots encore plus difficiles à manipuler que ceux présentés ici : des robots qui ne mémorisent pas leur orientation, qui ne partagent pas la même notion de droite et de gauche, qui ne communiquent entre eux que par des lumières qu'ils portent et les identifient ou dont ils peuvent changer la couleur, etc. Des algorithmes d'une subtilité étonnante sont proposés pour le problème du parcours de la grille, mais aussi pour le parcours d'un graphe ou pour le parcours répété infiniment d'une partie de grille.

Pour visualiser les solutions du robot disposant d'une mémoire d'un entier, du robot Petit Poucet, des cinq robots coordonnés et des trois robots de la dernière méthode, nous vous proposons d'aller voir sur https://github.com/cristal-smac/robot-spiral-path une animation et un programme qui leur sont consacrés. ■

BIBLIOGRAPHIE

A. Rauch *et al.*, **Optimal exclusive perpetual grid exploration by luminous myopic robots without common chirality,** *Netys 2021 Proceedings*, 2021.

Q. Bramas *et al.*, **Optimal, exclusive perpetual grid exploration by luminous myopic opaque robots with common chirality,** *ICDCN21 Proceedings*, 2021.

Q. Bramas *et al.*, **Infinite grid exploration by disoriented robots,** *Netys 2020 Proceedings*, 2021.

Y. Emek *et al.*, **How many ants does it take to find the food ?,** *Theoretical Computer Science*, 2015.