Agrégation externe de mathématiques, session 2008
Épreuve de modélisation, option informatique

(*public 2008*)

Résumé : Étude de diverses méthodes pour gérer une équipe de robots qui doivent coopérer

Mots clefs : automates, graphes

► *Il est rappelé que le jury n'exige pas une compréhension exhaustive du texte. Vous êtes laissé(e) libre d'organiser votre discussion comme vous l'entendez. Ce texte comporte un exercice de programmation et des suggestions de développement largement indépendantes les unes des autres. Vous n'êtes pas tenu(e) de suivre ces suggestions. Ce n'est pas une épreuve centrée sur la programmation. Il vous est conseillé de mettre en lumière vos connaissances à partir du fil conducteur constitué par le texte.*

De nombreuses activités exigent l'intervention de robots au lieu d'humains, que ce soit en milieu hostile, sur d'autres planètes ... Lorsque de plus leur travail est complexe et varié, il est plus efficace de construire une flotte de quelques robots spécialisés chacun dans un domaine plutôt qu'un unique robot multifonctions ; cela assure une plus grande fiabilité au système face aux pannes, et permet que les différents éléments agissent en parallèle. Mais il faut alors également prévoir que les robots puissent se rencontrer régulièrement et échanger leurs informations.
Le but de ce texte est de présenter quelques-unes des méthodes d'approche de ce problème. Le candidat n'est pas tenu de toutes les aborder. Il peut choisir de les comparer, d'en implanter certaines ...

1. Présentation du problème

Le système est constitué de n robots R_1, ..., R_n.
Il existe un total de m rendez-vous L_1, ..., L_m, mais chaque robot n'en utilise qu'une partie. En effet, pour éviter de surcharger les notations, L_i indique à la fois le lieu géographique du rendez-vous et les numéros des robots qui doivent s'y rencontrer.
Chaque robot R_i, pour $i \in [|1,n|]$, a donc en mémoire le tableau T_i de ses rendez-vous successifs. Lorsqu'il a parcouru tout le tableau, il recommence au début.
Ainsi, pour $n = 3$ et $m = 3$ par exemple, les tableaux :

$$T_1 = [1,3],\ T_2 = [2,1],\ T_3 = [3,2]$$

signifient que R_1 doit se présenter en L_1 puis en L_3, retourner en L_1 ... ; R_2 fait des aller- retour entre L_2 et L_1 et R_3 entre L_3 et L_2. Ici, il faut bien sûr comprendre que L_1 est un rendez-vous en un certain lieu entre R_1 et R_2, L_2 entre R_2 et R_3, L_3 entre R_1 et R_3.

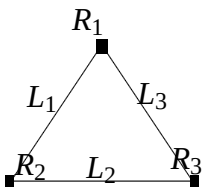
2. Méthode élémentaire

L'algorithme le plus simple consiste à ordonner à chaque robot de se rendre à son premier point de rendez-vous, d'y attendre que son collègue le rejoigne, puis de passer au point de rendez-vous suivant et ainsi de suite.
Comme on s'en convaincra avec l'exemple précédent, cet algorithme peut échouer : R_1 attend R_2 qui attend R_3 qui lui-même attend R_1, à des endroits *a priori* différents !
Or, le problème que l'on veut résoudre est en fait que chaque couple de robots se rencontre régulièrement, et il suffirait de modifier l'ordre des rencontres, par exemple en prenant les tableaux

$$T_1 = [1,3],\ T_2 = [1,2],\ T_3 = [2,3]$$

pour que le système fonctionne parfaitement.
On peut modéliser ceci par un graphe. L'ensemble des sommets est l'ensemble des robots, et il existe un lien entre deux sommets si et seulement si les robots correspondants ont un rendez-vous commun.

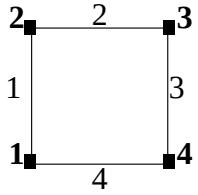
Ainsi, pour l'exemple précédent, le graphe est :

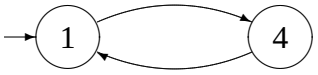


Il suffit alors de choisir un ordre de parcours des différents rendez-vous possibles, et on saura comment organiser le planning des robots pour que le système fonctionne. Cela revient, pour chaque robot, à décider dans quel ordre il doit rencontrer ses collègues. Prenons par exemple comme convention que chaque robot parcourt ses rendez-vous successifs par ordre de numéro croissant.
Nous modéliserons ceci de la façon suivante. Le problème nous est donné par un graphe $\mathscr{G}$ tel que le précédent, dans lequel les sommets sont les numéros des différents robots et les arêtes sont étiquetées par les numéros des rendez-vous (on n'indiquera plus que les numéros 1, 2 ... et non R_1, R_2 ... pour les nœuds du graphe et les numéros 1, 2, ... et non L_1, L_2 ... pour les étiquettes des arêtes du graphe).
Pour chaque robot R_k, on associe à ce graphe un automate $\mathscr{A}_k$ construit comme suit :

- les états sont les numéros des rendez-vous qui concernent le robot R_k (i.e. des arêtes dont une extrémité est k dans $\mathscr{G}$) et l'état initial i_k est l'état de numéro minimal parmi ceux-ci
- il existe dans $\mathscr{A}_k$ une transition de l'état i à l'état j si et seulement si :
  - soit $i < j$ et il n'existe pas dans $\mathscr{A}_k$ d'état p tel que $i < p < j$
  - soit i et j sont respectivement les numéros maximal et minimal des états de $\mathscr{A}_k$

Ainsi, si on a un système de quatre robots pour lequel le graphe $\mathscr{G}$ est :



l'automate $\mathscr{A}_1$ associé au robot R_1 est : 

Il ne reste qu'à construire un automate produit $\mathscr{A}$ des automates $\mathscr{A}_k$ pour savoir comment se comporte le système. On ne conserve évidemment dans le produit que les états accessibles.
Pour vérifier que le système ne se bloque pas, il suffit de voir que tout état de l'automate produit a un successeur. Est-ce bien le cas ?
Par contre, il pourrait arriver que l'un des robots soit bloqué quelque part, même si le système complet n'est pas bloqué. On montre que, si le système de robots est connexe, c'est-à-dire s'il existe entre deux robots quelconques une suite de points de rendez-vous les reliant, aucun robot ne peut rester bloqué.
Enfin, on peut s'intéresser au temps de parcours des différents rendez-vous et montrer que les cycles de l'automate $\mathscr{A}$ ont une longueur multiple du nombre de points de rendez-vous.

Exercice de programmation :

> *Il vous est demandé de rédiger un programme conforme aux spécifications ci-dessous dans l'un des langages C, Caml ou Java à votre choix. Ce programme devra être accompagné d'un exemple d'éxécution permettant d'en vérifier le bon fonctionnement. La clarté et la concision du programme seront des éléments importants d'appréciation pour le jury.*

On suppose donnés les tableaux T_i. Un état du système est représenté par un vecteur U de longueur n, dont la i-ème composante contient la position du robot R_i (sous forme du numéro j du lieu L_j où il se trouve).
Écrire une fonction/procédure/méthode `transition` qui transforme U en un état suivant du système (on admettra que les données sont telles qu'il existe un état suivant).
Simuler le système de robots pendant n unités de temps. On prendra comme état initial du robot R_i le premier élément du tableau T_i.

3. Méthodes plus évoluées

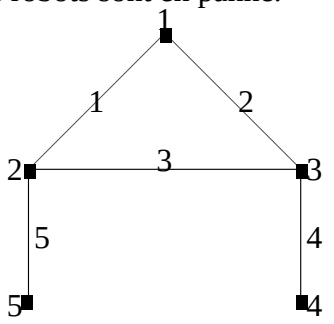
Il ne faut pas oublier que l'on veut aussi que le système soit robuste en cas de défaillance de certains robots. Ce ne serait pas le cas de la méthode précédente, dans laquelle un robot attendra indéfiniment son partenaire au rendez-vous fixé, même si celui-ci est en panne quelque part.

3.1. *Solution avec temps d'attente*

Pour gérer ce problème, on peut rajouter dans la programmation de chaque robot un minuteur, et lui imposer un délai maximum d'attente après lequel il abandonne son rendez-vous et passe au suivant.
Par convention, une unité de temps est la durée du voyage d'un point à un autre, et on continue à ne pas tenir compte des durées des phases de travail.

On voudrait savoir s'il est possible, en fixant un temps d'attente maximal, d'assurer la correction de la méthode lorsqu'un (resp. deux, un nombre quelconque) de robots sont en panne.

Prenons l'exemple à cinq robots modélisé le graphe $\mathscr{G}$ ci-contre.



On applique la méthode précédente, en donnant un temps d'attente maximal N à tous les robots, et on suppose que le robot numéro 4 est en panne. Que se passe-t-il ?
Comme précédemment, on se demandera d'une part si le système complet peut rester bloqué, et d'autre part s'il fonctionne « correctement » compte tenu de l'absence de R_4.

3.2. *Modélisation par une chaîne de Markov*

Pour éviter que le système ne boucle trop vite, on peut rajouter une composante aléatoire au fonctionnement du robot : lorsqu'il a épuisé son temps d'attente, le robot décide aléatoirement, soit d'attendre encore 1 unité de temps, soit de passer au rendez-vous suivant (ou de revenir au premier rendez-vous, au choix du programmateur du système). On donnera par exemple une équiprobabilité aux deux possibilités.
L'état de chaque robot est désormais repéré par le numéro du rendez-vous auquel il se trouve et le temps d'attente qu'il lui reste. Le système est donc modélisé par une chaîne de Markov construite en prenant à chaque instant le produit des états de chacun des robots.
On admettra le lemme suivant.

***Lemme* 1.** *Soit S' un sous-ensemble des états d'une chaîne de Markov finie, tel que, pour tout état s, il existe un chemin de s à un état s' de S'. Alors, quel que soit l'état initial, la probabilité d'atteindre l'un des états de S' est* 1.

À l'aide de ce lemme, on se convaincra assez facilement que la méthode proposée ici est efficace.

4. Réseau de Petri

Le système de robots peut aussi être modélisé par un réseau de Petri. Il s'agit d'un quadruplet $(P, T, Pre, Post)$ où P est un ensemble fini de places, T un ensemble fini de transitions, $Pre, Post$: $P \times T \to \mathbb{N}$ sont des applications.
On associe à un réseau de Petri un graphe dont les sommets sont soit des places (notées avec un rond) soit des transitions (notées avec un carré). Le graphe contient un arc de la place p vers la transition t si et seulement si $Pre(p,t)$ est non nul, de la transition t vers la place p si et seulement si $Post(p,t)$ est non nul.

On « marque » un réseau de Petri en positionnant à chaque place un certain nombre de jetons. Un marquage M est donc un vecteur d'entiers naturels à π composantes, où π est le nombre de places, et $M(p)$ désignera le nombre de jetons à la place p.
On dit que la transition t est franchissable lorsque :

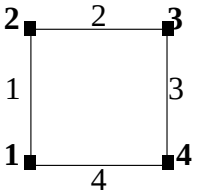
$$\forall p \in P, M(p) \geqslant Pre(p,t)$$

et on franchit dans ce cas la transition t en remplaçant le marquage M par le marquage M' défini par :

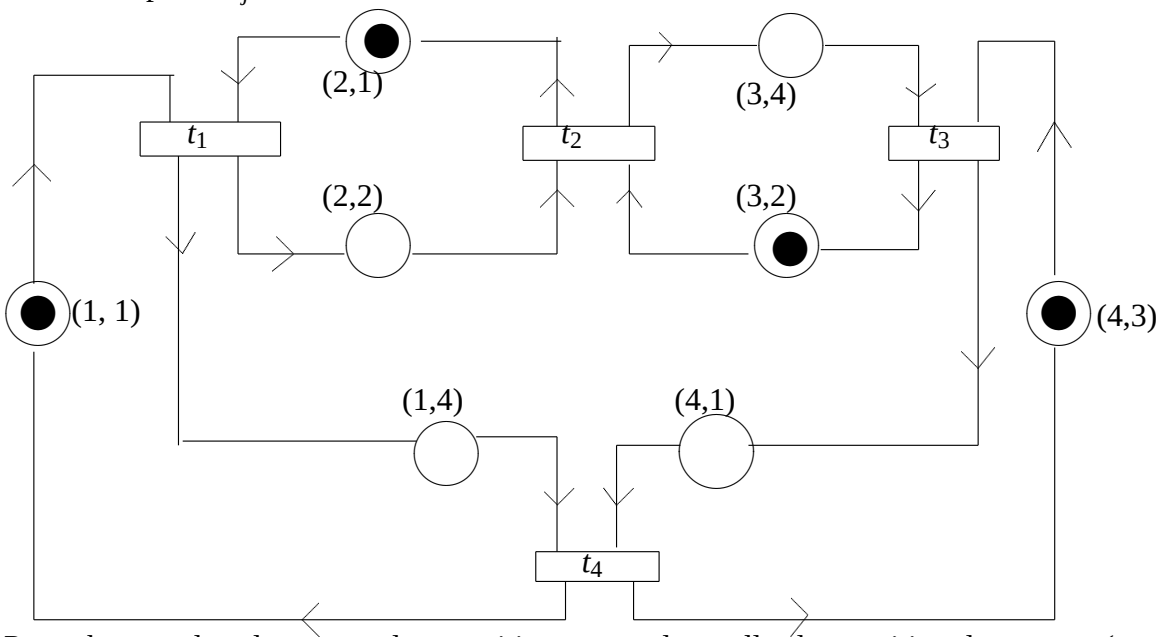
$$\forall p \in P, M'(p) = M(p) - Pre(p,t) + Post(p,t).$$

À partir d'un marquage initial M_0, le fonctionnement d'un réseau de Petri est ainsi décrit par la suite des marquages parcourue.

Appliquons ceci au système de robots décrit par le graphe $\mathscr{G}$:



L'ensemble P choisi est l'ensemble des couples (i, j) où i est le numéro d'un robot et j celui d'un point de rendez-vous. Le fait que la place (i, j) soit marquée signifiera que le robot R_i se trouve à la place L_j. Cela conduit au réseau de Petri :



Pour chacune des places p et des transitions t pour lesquelles la transition de p vers t (resp. de t vers p) existe :

$$Pre(p,t) = 1 \text{ (resp. } Post(p,t) = 1)$$

et $Pre(p,t) = 0$ (resp. $Post(p,t) = 0$) pour les autres couples.
Les gros points noirs indiquent la position initiale des jetons (i.e. le marquage M_0).
Initialement, la transition t_1 est franchissable. Une fois qu'elle est franchie, t_2 devient franchissable, puis t_3, puis t_4, et on retrouve ensuite la configuration initiale. Ce réseau décrit bien le fonctionnement du système.

De façon plus générale, on peut utiliser de tels réseaux de Petri pour étudier si le système de robots se comporte correctement.

5. Problème complémentaire

Une fois qu'on a assuré que les robots se voient régulièrement, il reste à savoir comment ils peuvent se transmettre des messages. C'est un problème de routage : assurer que deux robots R_i et R_j quelconques puissent communiquer, même s'ils n'ont pas de rendez-vous commun, en passant par l'intermédiaire de leurs collègues de travail.
Puisque le réseau a été supposé connexe (du moins tant qu'il n'y a pas de panne), ce problème admet des solutions, mais il resterait à voir si on peut trouver des méthodes efficaces.

Suggestions pour le développement

► *Soulignons qu'il s'agit d'un menu à la carte et que vous pouvez choisir d'étudier certains points, pas tous, pas nécessairement dans l'ordre, et de façon plus ou moins fouillée. Vous pouvez aussi vous poser d'autres questions que celles indiquées ci-dessous.*

- Justifier certaines des affirmations des paragraphes 2 et 3 concernant l'efficacité des méthodes proposées et le temps de parcours des robots
- Expliquer comment implémenter certaines des méthodes proposées et éventuellement en implémenter des parties
- Proposer d'autres façons de modéliser le système.