Agrégation externe de mathématiques, session 2006
Épreuve de modélisation, option informatique

(*D01*)

Résumé : On s'intéresse à la résolution et à la création de problèmes de Su Doku.

Mots clefs : Parcours d'arborescences, coloration de graphes, calculs de complexité.

► *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.*

Le Su Doku est un jeu de réflexion ayant fait son apparition dans les journaux de grande diffusion français durant l'été 2005. Une grille de Su Doku est composée de neuf carrés de neuf cases, soit quatre-vingt-une cases. Le but du jeu est de remplir chacune de ces cases avec un chiffre compris entre 1 et 9, de telle sorte que dans chaque ligne, dans chaque colonne et dans chaque carré apparaisse une fois et une seule chaque chiffre. En début de jeu certaines cases sont remplies, en principe de telle façon qu'il soit possible de compléter la grille d'une et d'une seule manière en respectant les règles précédentes.

Voici une grille (facile) à compléter, ainsi que sa complétion :

1	2			6		9		
	4	9				2		
5			2	9		4	3	
7					2		1	8
	3		7	1	6		2	
9	1		5					6
	7	8		5	4			3
		3				6	7	
		1		2			5	4

1	2	7	4	6	3	9	8	5
3	4	9	8	7	5	2	6	1
5	8	6	2	9	1	4	3	7
7	6	5	9	4	2	3	1	8
8	3	4	7	1	6	5	2	9
9	1	2	5	3	8	7	4	6
2	7	8	6	5	4	1	9	3
4	5	3	1	8	9	6	7	2
6	9	1	3	2	7	8	5	4

Une grille dont toutes les cases sont remplies sera dite *complète*, une grille dont certaines sont vides sera dite *incomplète* ; si elle peut être complétée en une ou plusieurs grilles vérifiant les règles elle sera dite *admissible*.

Ce jeu crée deux problèmes : celui du joueur qui doit compléter la grille et celui du concepteur qui doit créer une grille incomplète, mais qui ne puisse être complétée que d'une et d'une seule façon.

On s'intéresse maintenant aux grilles de taille $n^2 \times n^2$, où n est un entier ($n = 3$ pour le Su Doku) vérifiant les même règles à l'ordre n au lieu de l'ordre 3.

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'exécution permettant d'en vérifier le bon fonctionnement. La clarté et la concision du programme seront des éléments importants d'appréciation pour le jury.

Écrire une fonction prenant pour paramètres un entier, p et un tableau carré de côté p (donc de taille p^2) d'entiers, T et renvoyant un booléen disant si ce tableau est un carré latin, c'est-à-dire contenant dans chaque ligne et chaque colonne une et une seule fois chaque entier de 1 à p.

En prenant $p = n^2$ on obtient une partie des contraintes d'admissibilité d'une grille complète de Su Doku, mais il reste encore à vérifier la contrainte sur les petits carrés.

1. Dénombrer les grilles possibles

1.1. *Dénombrement par énumération exhaustive*

On peut interpréter une grille comme un vecteur de taille n^4 en concaténant les lignes. On peut donc introduire sur les grilles admissibles un ordre qui est l'ordre lexicographique pour les vecteurs associés, les cases vides étant supposées contenir la valeur 10.

Il est possible, en partant de la grille totalement vide et en parcourant les grilles admissibles dans l'ordre lexicographique, d'obtenir la grille complète minimale pour l'ordre lexicographique. L'ordre de grandeur de la complexité maximale d'une telle opération est a priori assez élevé. En poursuivant l'exploration dans l'ordre lexicographique, on pourrait dénombrer toutes les grilles complètes admissibles. L'estimation de la complexité de cette opération, même pour $n = 3$ peut conduire au découragement.

1.2. *Nombre chromatique d'un graphe*

Un graphe non-orienté sans boucle est un couple $G = (S, A)$ où S est un ensemble fini dont les éléments s'appellent les *sommets* de G, et A une famille de parties à deux éléments de S dont les éléments s'appellent les *arêtes* de G. Une même arête peut donc être présente plusieurs fois dans A.

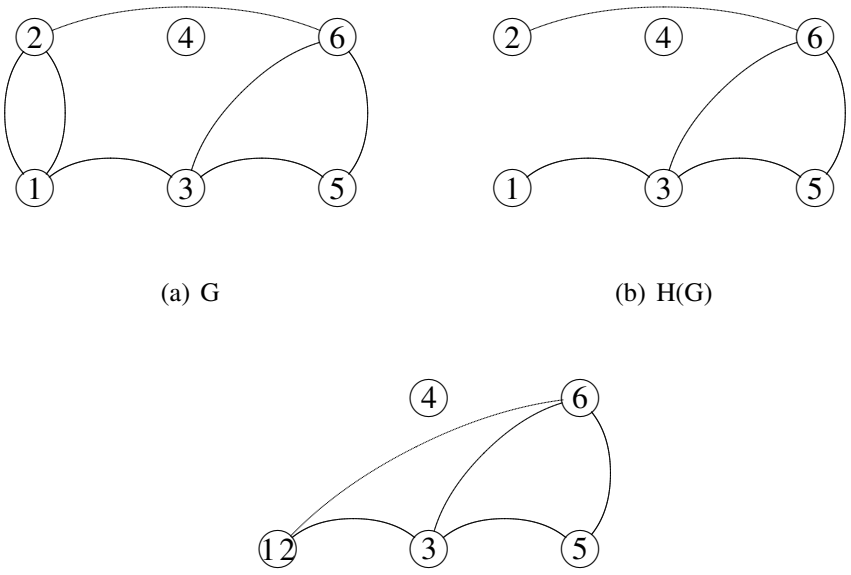
Un coloriage à p couleurs de G est une application c de S vers l'intervalle $[1, p]$ des entiers naturels, telle que si $\{s, s'\}$ est une arête de G alors $c(s) \neq c(s')$. On note $N(G, p)$ le nombre de coloriages du graphe G avec p couleurs.

On peut associer un graphe G à une grille de Su Doku. Ses sommets sont les cases de la grille. $\{s, s'\}$ est une arête si et seulement si s et s' sont dans la même ligne, la même colonne ou le même carré. Le nombre $N(G, n^2)$ est alors le nombre de grilles admissibles.

La difficulté est de calculer $N(G, p)$ pour un graphe non orienté quelconque.

On propose l'algorithme suivant, pour un graphe de m sommets numérotés de 1 à m :
- si le graphe ne possède aucune arête $N(G,p)$ est facilement calculable ;
- si le graphe G possède au moins une arête de sommets s_1 et s_2.On construit le graphe $H(G)$ obtenu en supprimant dans G toutes les arêtes joignant s_1 et s_2 ;
- on appelle $K(G)$ le graphe obtenu à partir de $H(G)$ en confondant les sommets s_1 et s_2 et en renumérotant ses sommets de 1 à $m-1$; et
- on a alors $N(G,p) = N(H(G),p) - N(K(G),p)$.

En supposant l'algorithme valide, on peut prouver que $N(G,p)$ est un polynôme en p de degré Card$S$.



(a) G (b) H(G)

(c) K(G)

FIG. 1. Construction des graphes H(G) et K(G)

2. Le point de vue du joueur : la reconstruction

La solution étant définie par un ensemble de contraintes, la programmation d'un algorithme de résolution est aisée dans un langage adapté à la logique. Mais l'efficacité d'un tel programme peut être médiocre.

2.1. *Les grilles faciles*

Une grille incomplète est *facile* si à chaque étape de sa complétion, il existe au moins une case vide qui ne peut être remplie que par une unique valeur, du fait des contraintes directes

imposées par la numérotation d'autres cases de sa ligne, de sa colonne ou de son carré. La résolution du problème est alors aisée et de complexité polynomiale.

2.2. *Les grilles de difficulté moyenne*

Si cette condition n'est pas vérifiée, une deuxième condition peut simplifier le problème. Il suffit que, lors d'une étape de la complétion où la première condition n'est pas vérifiée, il existe un nombre et un ensemble de base (ligne, colonne ou carré) dans lequel il ne reste qu'une seule case dans laquelle on puisse placer ce nombre, du fait des contraintes directes à cet étape. La complexité de la résolution reste polynomiale.

2.3. *Les grilles difficiles*

Pour une grille ne possédant pas la propriété des grilles moyennes,on pourra explorer l'arborescence des grilles admissibles. On dispose alors de plusieurs stratégies usuelles pour cela (par exemple parcours en largeur d'abord, parcours en profondeur d'abord). Il faudra alors définir des structures de données adaptées à chacune de ces stratégies.

On a fait la remarque que l'exploration exhaustive des grilles admissibles, même dans l'ordre lexicographique, pouvait être peu efficace. On aura donc intérêt à développer des heuristiques d'exploration, basées par exemple sur les méthodes utilisées pour les grilles faciles et de difficulté moyenne.

3. Le point de vue du créateur : l'effacement

3.1. *Les grilles faciles et de difficulté moyenne*

La création d'une telle grille n'est pas conceptuellement différente de celle de sa résolution. Sa complexité est aussi polynomiale. Il suffit par exemple de partir d'une grille admissible complète et d'effacer des cases une à une en s'assurant que la grille obtenue conserve à chaque étape la ou les propriétés souhaitées.

3.2. *Les grilles difficiles*

Le problème de la construction de grilles difficiles est qu'il faut s'assurer de l'existence et de l'unicité de la solution. Pour l'existence, cela est clair si on est parti d'une grille admissible complète dans laquelle on a effacé des cases, mais cela l'est moins si on a placé des nombres dans une grille vide, en respectant l'admissibilité de la grille. Il faut donc un algorithme permettant d'obtenir toutes les solutions d'un problème donné, ou tout au moins capable d'affirmer qu'une solution trouvée est unique (et dans ce cas l'algorithme nous donne bien toutes les solutions puisqu'il n'y en a qu'une).

Le problème de savoir si une grille dont on sait qu'elle est admissible peut être complétée d'une autre façon est un problème *NP*-complet (inutile néanmoins de s'intéresser ici à la justification de ce résultat).

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.*

- Donner une explication aux ordres de grandeur de complexité donnés. Donner des estimations par défaut ou par excès des complexités non exprimées.
- Proposer plusieurs algorithmes de parcours de l'arborescence des grilles admissibles, dans l'optique de la recherche d'une solution, en discutant de leurs avantages et de leurs inconvénients, suivant la valeur de n.
- Proposer des structures de données adaptées aux algorithmes, en lien par exemple avec la proposition précédente.
- Donner une idée de ce que pourrait être un programme logique permettant de compléter une grille.
- Justifier la validité de l'algorithme permettant de calculer $N(G,p)$, et le fait que $N(G,p)$ soit bien un polynôme en p .