Recherche Opérationnelle

IUP-MIAGe Master 1

janvier 2004

Tous documents autorisés

1. **6 points** Quel est le calcul qui permet de déterminer la distribution stationnaire d'une chaîne de Markov qui a une seule classe finale si cette classe est apériodique?

   réponse On résoût un système d'équations où il y a une équation pour chaque colonne i de la matrice P des probabilités de transition (sauf une colonne). Cette équation exprime le fait que si on est dans la distribution stationnaire, la probabilité d'arriver dans l'état i est égale à la probabilité (p_i) de cet état dans la distribution:
   $p_i \ = \ \sum_{j=1}^{j=n} p_j P_{j,i}$
   Les n équations ne sont pas linéairement indépendantes; pour cette raison, on en supprime une (choisie arbitrairement) et on la remplace par l'équation:
   $\sum_{i=1}^{n} p_i \ = \ 1$
   Expliquer pourquoi ce calcul n'est pas valable dans les autres cas.

   réponse Dans le cas de plusieurs classes finales, il y a plusieurs distributions stationnaires (une pour chaque classe finale plus leurs moyennes pondérées) et le système d'équations donné ne peut pas être résolu.
   Dans le cas d'une classe finale périodique, il y a une distribution stationnaire mais elle n'est pas forcément la limite de la chaîne

   Décrire un algorithme pour trouver les classes d'une chaîne et de décider si elles sont finales et/ou périodiques.

   réponse Construire la matrice booléenne $A[n, n]$ qui a des valeurs vraies où P a des valeurs strictement positives;

   Construire $B[n, n]$ qui est comme A sauf que tous les éléments $B[i, i]$ sont vrais.

   Calculer la puissance $C \ = \ B^{n-1}$. De C on peut facilement trouver les états de la même classe qu'un état donné i; ils sont les états j tels que $C[i, j] = C[j, i] = vrai$. Trouver la classe de l'état 1; et les supprimer; ensuite trouver la classe du premier état qui reste et ainsi de suite.

   Une classe est finale si aucun état i de la classe a $P[i, j] > 0$ pour un état j qui n'est pas dans la classe.

Pour décider si une classe est périodique, on calcule une grande puissance $D = A^m$ ($m = n^2$ suffit). La classe est périodique si on a $D[i,j] = faux$ pour deux états i et j de la classe.

2. **6 points** Décrire comment un programme linéaire exprimé par des contraintes $\leq$ peut être transformé dans la forme utilisée par l'algorithme du simplexe (avec des contraintes d'égalité et les contraintes que toutes les variables soient ≥ 0).

   réponse Pour une contrainte de la forme $l \leq c$ (l une expression linéaire et c une constante), on ajoute une nouvelle variable (d'écart) y et on remplace la contrainte par $l + y = c$ (et $y \geq 0$).

   S'il y a des variables (exemple x) qui ne sont pas contraintes d'être ≥ 0, on remplace x par deux nouvelles variables, disons x_1 et x_2, on remplace chaque occurrence de x par $x_1 - x_2$ et on ajoute $x_1 \geq 0$ et $x_2 \geq 0$.

   Expliquer le processus du choix d'un pivot dans l'algorithme du simplexe et pourquoi ce choix du pivot garantit que la valeur de la fonction objective ne peut pas être réduite.

   réponse Il faut noter que les éléments du vecteur b (les constantes des contraintes) sont forcément ≥ 0 parce que la solution actuelle est réalisable.

   On choisit une colonne j avec $c_j > 0$; on ne considère que les lignes i avec $A[i,j] > 0$; parmi ces lignes on choisit une i qui minimise $b_i/A[i,j]$. Le pivot est $A[i,j]$.

   Pourquoi une opération de pivotage avec ce choix du pivot est-elle garantie de préserver les conditions utilisées par votre réponse à la question précédente?

   réponse

   L'opération de pivotage va maintenant incrémenter x_j de 0 à $b_i/A[i,j]$ et les autres variables qui étaient hors base restent nulles. Donc la fonction objective est incrémentée par $c_j \times b_i/A[i,j]$ qui est ≥ 0 parce que $c_j > 0$, $A[i,j] > 0$ et $b_i \geq 0$.

   Et chaque élément b_k reste ≥ 0 parce que le calcul est $b[k] - = b[i] * A[k,j]/A[i,j]$; on a $b[i] \geq 0$ et on a choisi $A[i,j] > 0$; soit $A[k,j] \leq 0$ et $b[k]$ ne diminue pas, soit $A[k,j] \geq 0$ et le choix de i donne que $b_i/A[i,j] <= b_k/A[k,j]$ ce qui garantit que b_k reste ≥ 0.

3. **4 points** Par un exemple d'un programme en 2 variables, démontrer que c'est possible qu'un Programme Linéaire en Nombres Entiers est faisable mais la solution optimale de son programme relaxé ne peut pas être transformée en solution réalisable entière en arondissant chaque variable soit vers le haut soit vers le bas.

   réponse Le programme: maximiser x avec les contraintes $5x - 6y \geq 0$, $5x - 2y \leq 5$, $y \geq 0$.

   Les seules solutions entières réalisables sont $(0,0)$ et $(1,0)$ mais la solution optimale du programme relaxé est $(1.5, 1.25)$.

4. **4 points** Expliquer comment la programmation dynamique peut aider à résoudre le problème suivant:

   réponse

   Je veux calculer le meilleur chemin pour conduire du point Sud-ouest au point Nord-est d'une ville en voyageant toujours vers le nord ou vers l'est.

   La ville est dans la forme d'un carré de 1 km par 1 km est il y a des routes du sud au nord et de l'ouest à l'est tous les 100 mètres; à chaque intersection il y a des feux tricolores et j'ai un tableau donnant le temps moyen d'attente à chaque intersection dans les quatre cas possibles (arrivée du sud ou de l'ouest, départ vers le nord ou l'est). Je conduis toujours à une vitesse de 50 km/h. Je veux trouver le chemin qui minimise le temps total moyen.

   réponse Il faut deux tableaux donnant pour chaque intersection le temps moyen pour arriver à NE dans les deux cas où on y arrive du S ou de l'O. On met V et H (vertical/horizontal) pour ces deux et HH etc. pour les données sur les temps d'attente et C pour le temps de parcourir 100 mètres à 50 km/h. Si on numérote les rues de 0 à 10 dans les deux directions avec le $(0, 0)$ au point de départ, il y a trois cas de calcul selon si l'intersection est sur l'extremité N, ou sur l'extremité E de la ville ou à l'intérieur:
   le calcul se fait de la façon suivante:

```
H[10,10]=V[10,10]=0;
for (i=9;i>=0;i--)
   {
   H[10,i]=H[10,i+1]+HH[10,i]+C;
   V[i,10]=V[i+1,10]+VV[i,10]+C;
   }
for (i=9;i>=0;i--)
   for (j=9;j>=0;j--)
      {
      H[i,j]=min(HV[i,j]+V[i+1,j],HH[i,j]+H[i,j+1])+C;
      V[i,j]=min(VV[i,j]+V[i+1,j],VH[i,j]+H[i,j+1])+C;
      }
```

   et le résultat se trouve en `H[0,0]` ou `V[0,0]` selon la direction d'entrée dans la ville.

   Le temps de calcul est, donc, $O(n^2)$ pour une ville $n \times n$.