Mariages parfaits randomisés

Introduction

On se donne un ensemble fini $U = \{u_1, \ldots, u_n\}$ de n "serveurs" et un ensemble fini $V = \{v_1, \ldots, v_n\}$ de n "clients". Les services sont fournis à travers des cablages; pour certains couples $(u_i, v_j) \in U \times V$, ces cablages ont effectivement été réalisés. L'ensemble $E \subset U \times V$ de ces couples est une donnée imposée du problème.

On désire relier chaque client à un serveur de manière bijective ("un client - un serveur"), et ce, à travers les cablages existants. Voici deux exemples:

Il est clair que le premier réseau de cablages permet d'attribuer des serveurs aux clients en respectant les règles énoncées, alors que le second réseau de cablages ne le permet pas, bien qu'il contienne plus de cablages. Le problème est de décider, en général, si un réseau donné permet une telle attribution et, le cas échéant d'en construire une. Nous ne ferons qu'effleurer ce dernier aspect.

1 Formalisme et notations

Définition 1
Un *graphe biparti* $G = (U, V, E)$ est formé de deux ensembles finis disjoints de *sommets* U et V et d'un ensemble $E \subset U \times V$ d'*arêtes*. Les *extremités* d'une arête $(u, v) \in E$ sont les sommets u et v. Nous ne considérerons que le cas où les ensembles U et V ont même nombre $n \geq 1$ d'éléments: $|U| = |V| = n \in \mathbf{N}^*$.

Définition 2
Un *mariage parfait* sur le graphe biparti $G = (U, V, E)$ est un sous-ensemble $E' \subset E$ d'arêtes tel que chaque sommet est extremité d'une et une seule arête de E'.

Énumérons les sommets: $U = \{u_1, \ldots, u_n\}$ et $V = \{v_1, \ldots, v_n\}$. Il revient évidemment au même de se donner une permutation $\sigma \in \mathcal{S}_n$ telle que $\forall i \in [[1, n]]$, $(u_i, v_{\sigma(i)}) \in E$.

Naturellement, un algorithme d'exploration systématique de toutes les possibilités permettrait de vérifier s'il existe un mariage parfait et d'en exhiber un, mais le coût serait prohibitif (au moins d'ordre exponentiel). En revanche, tester si un mariage donné est parfait (et compatible avec le graphe) est un problème facile, conformément à une situation très répandue dans les problèmes combinatoires: trouver une solution coûte cher mais la tester est possible en temps polynomial.

Définition 3
On associe au graphe biparti $G = (U, V, E)$ (supposé tel que $|U| = |V| = n \in \mathbf{N}^*$) une matrice A_G sur l'anneau $\mathbf{Z}[\underline{X}] := \mathbf{Z}[X_{1,1}, \ldots, X_{n,n}]$ des polynômes à coefficients entiers en les indéterminées $X_{i,j}$, où $1 \leq i, j \leq n$. Les coefficients $a_{i,j}$ de A_G sont donnés par la règle suivante:

$$\forall (i,j) \in [[1,n]]^2 \ , \ a_{i,j} = \begin{cases} X_{i,j} \text{ si } (u_i, v_j) \in E \\ 0 \text{ si } (u_i, v_j) \notin E \end{cases}$$

Cette matrice est donc à distinguer de l'habituelle matrice des connexions du graphe G (que l'on retrouverait en remplaçant partout $X_{i,j}$ par 1 dans A_G).

Théorème 1
Il existe un mariage parfait si, et seulement si, le polynôme $P_G := \det A_G$ de $\mathbf{Z}[\underline{X}]$ n'est pas nul.

Par exemple, les deux graphes dessinés plus haut ont pour matrices:

$$\begin{pmatrix} X_{1,1} & X_{1,2} & 0 \\ 0 & 0 & X_{2,3} \\ 0 & X_{3,2} & 0 \end{pmatrix} \quad \text{et} \quad \begin{pmatrix} X_{1,1} & X_{1,2} & X_{1,3} \\ X_{2,1} & 0 & 0 \\ X_{3,1} & 0 & 0 \end{pmatrix},$$

dont les déterminants sont respectivement $-X_{1,1}X_{2,3}X_{3,2}$ et 0. On peut même "extraire" du premier polynôme l'information que le seul mariage parfait est $\{(u_1, v_1), (u_2, v_3), (u_3, v_2)\}$.

Le théorème suggère donc de calculer le déterminant P_G. Le calcul évident (formule complète, ou développements successifs le long d'une ligne ou d'une colonne) conduit à une explosion des calculs *et de la taille des "coefficients"*. Les méthodes de type "pivot" permettent en principe un calcul en nombre d'étapes polynomial, mais l'explosion de taille des coefficients maintient un temps de calcul prohibitif, comme on peut le vérifier sur l'exemple d'un graphe biparti complet (cas où $E = U \times V$).

2 Un algorithme probabiliste

On propose donc un algorithme faillible, dont le principe est le suivant. On *évalue* le polynôme P_G en un point à coordonnées entières $(a_{1,1}, \ldots, a_{n,n})$, autrement dit, on calcule le déterminant obtenu en remplaçant dans A_G les indéterminées $X_{i,j}$ par des valeurs $a_{i,j}$. En principe, des algorithmes efficaces (de type pivot de Gauss) permettent de le faire en temps polynomial, et l'on ne s'attend pas *a priori* à une explosion des données (mais ce point sera discuté plus bas). Si l'on trouve une valeur non nulle, il est certain que P_G n'est pas nul, et l'on sait qu'il existe un mariage parfait. Si l'on trouve une valeur nulle on peut conjecturer que P_G est nul et qu'il n'existe pas de mariage parfait, mais au risque de se tromper.

Nous allons temporairement nous intéresser à un polynôme à coefficients entiers en $s \geq 1$ indéterminées $P(T_1, \ldots, T_s) \in \mathbf{Z}[T_1, \ldots, T_s]$, qui est soit nul soit de degré $d \geq 1$, et pour lequel on dispose d'une méthode efficace (mais non précisée) d'évaluation en tout point de $\mathbf{Z}^s$. Pour tout entier naturel $m \geq 1$, nous noterons $[m] = [[1, m]]$. On suppose donné un programme de tirages aléatoires indépendants dans $[m]$.

```
Algorithme Nul(P)
tirer aleatoirement a1,...,as dans [m];
si P(a1,...,as) = 0
  alors rendre(vrai)
  sinon rendre(faux);;
```

Bien sûr, on peut remplacer cet algorithme par un algorithme qui ne se trompe pas: il suffit de choisir m assez grand et d'évaluer P en tous les points de $[m]^s$. À nouveau, cela représente un coût prohibitif. Nous allons plutôt évaluer la probabilité que l'algorithme se trompe. Cela ne peut se produire que si $P(a_1, \ldots, a_n) = 0$ alors que P est non nul.

Théorème 2
Soit $P \in \mathbf{R}[T_1, \ldots, T_s]$ un polynôme non nul de degré d. Alors la proportion des points de $[m]^s$ en lesquels P s'annule est majorée par $\frac{d}{m}$:

$$\frac{\text{card}\{(a_1, \ldots, a_n) \in [m]^s \ / \ P(a_1, \ldots, a_n) = 0\}}{m^s} \leq \frac{d}{m}.$$

On remarque que ce majorant ne dépend pas du nombre s d'indéterminées. L'idée est donc de choisir m assez grand par rapport au degré. Ensuite, la probabité p d'erreur étant faible, une exécution répétée de l'algorithme de base devrait finir par rendre une réponse correcte; mais combien d'essais semblent raisonnables avant de conclure ? Il s'agit d'un problème du type "temps moyen d'attente du premier pile" dans un jeu de pile ou face avec pièce truquée.

Corollaire 3
Si P n'est pas nul, le temps moyen d'attente avant que l'algorithme le dise est majoré par $\frac{m}{m-d}$.

```
Algorithme NulObstine(P,k)
i := 0;
repete (i := i + 1 ; reponse := Nul(P))
jusque ((i = k) ou (reponse = faux));
rendre(reponse);;
```

Reste à calibrer k d'après le corollaire et à appliquer cette méthode au problème de départ.

3 Calcul exact d'un déterminant

Nous allons nous intéresser à une autre méthode pour maitriser le coût du calcul d'un déterminant.

Soit $A \in M_n(\mathbf{Z})$ une matrice carrée à coefficients entiers et soit $D = \det A$. La méthode "efficace" habituelle de calcul de D est la méthode du pivot: on obtient par exemple une décomposition LU de la matrice A et il reste à effectuer $(n-1)$ multiplications. Le nombre total d'opérations élémentaires est majoré par un polynôme explicite, mais *il n'est pas raisonnable d'espérer que ces opérations se fassent en temps constant.* En effet, ces calculs se font dans $\mathbf{Q}$ et les numérateurs et les dénominateurs des fractions manipulées peuvent devenir très grands.

Pour maintenir des coûts constants (ou bornés) d'opérations arithmétiques, on choisit de faire les calculs modulo un nombre premier p. En effet, l'image $\overline{D}$ de D dans $\mathbf{Z}/p\mathbf{Z}$ est le déterminant de l'image $\overline{A}$ de A dans $M_n(\mathbf{Z}/p\mathbf{Z})$.

```
Algorithme det-mod-p(A)
calculer la decomposition LU de A mod p;
calculer D mod p;
relever en D;;
```

Il suffit de choisir p assez grand (et premier) pour que la connaissance de $\overline{D}$ permette de déterminer D. On choisit donc un majorant *a priori* de $|D|$ facile à calculer. C'est l'inégalité de

Hadamard qui nous le donne:

Théorème 4
Notons $C_1, \ldots, C_n$ les colonnes de A et $||C_1||, \ldots, ||C_n||$ leurs normes euclidiennes. Alors $|\det A| \leq ||C_1|| \cdots ||C_n||$.

Une variante de cette méthode consiste à réduire modulo plusieurs nombres premiers $p_1, \ldots, p_k$ dont le produit est assez grand, puis d'invoquer le lemme chinois, dont il faut alors connaitre une variante effective.

```
Algorithme det-mod-p1pk(A)
calculer la decomposition LU de A mod p1,...,pk;
calculer D mod p1,...,pk;
calculer D mod p1...pk;
relever en D;;
```

Pour en revenir à notre problème de départ, il nous suffit de reconnaitre qu'un polynôme n'est pas nul. Comme les coefficients non nuls de P_G valent ± 1, on voit que $P_G \pmod p$ est nul si, et seulement si, P_G est nul. On peut modifier ainsi l'algorithme. On choisit un nombre premier $p > m$. Le théorème 2 est alors valable tel quel dans $\mathbf{Z}/p\mathbf{Z}$. On évalue P_G en un point aléatoire, en itérant si nécessaire.

4 Suggestions de développements

- Dernier alinéa de la page 1: quel espace explore-t-on ? Il y a d'ailleurs deux réponses possibles: $\mathcal{P}(E)$ ou $\mathcal{S}_n$. On peut, dans chaque cas, proposer un algorithme, voire le programmer, et tenter de justifier les assertions sur les coûts de recherche et de vérification.
- Démontrer le théorème 1 et expliquer en quoi il permet en principe d'énumérer tous les mariages parfaits. On peut évoquer les structures de données pertinentes.
- Préciser et justifier les estimations pessimistes qui suivent le théorème 1, y compris en ce qui concerne la taille des coefficients dans la méthode du pivot. On peut traiter à la main, voire programmer, le cas d'un graphe complet.
- Préciser et justifier les estimations pessimistes qui suivent l'algorithme `Nul(P)`: comment choisir m ? Quel coût trouve-t-on pour le problème de départ ?
- Démontrer le théorème 2, et l'appliquer à l'exemple de P_G.
- Démontrer le corollaire 3, et l'appliquer à l'exemple de P_G: comment choisit-on k ? Quel coût obtient-on ? Quelle garantie de succés ? On pourra faire appel au théorème de Bienaymé-Tchebycheff. On pourra, si l'on préfère, programmer l'algorithme pour tester son comportement.
- Estimer le nombre d'opérations élémentaires lors du calcul du déterminant par décomposition LU, et vérifier sur des exemples que les tailles des fractions peuvent croitre sensiblement.
- Démontrer l'inégalité de Hadamard et préciser l'algorithme `det-mod-p(A)`. Préciser de même l'algorithme `det-mod-p1pk(A)`, y compris en ce qui concerne la forme effective du lemme chinois. Comparer les coûts de `det-mod-p(A)` et de `det-mod-p1pk(A)`.
- Vérifier que le théorème 2 s'étend au corps $\mathbf{Z}/p\mathbf{Z}$ si $p > m$, et détailler l'algorithme correspondant (description et coût). Exécuter cet algorithme (et les précédents) sur des exemples.
- Question plus ouverte: comment utiliser ces algorithmes pour *construire* un mariage parfait?