Méthodes algorithmiques avancées

Jerzy KORCZAK, Piotr LIPINSKI, Arnaud QUIRIN
email : <jjk,lipinski, quirin>@dpt-info.u-strasbg.fr

Introduction

Informatique : …une manière de pensée et de communication
hacker (ang) *… a person who enjoys programming rather than just theorizing about programming …*

Le point de vue dans ce cours :
En informatique, … théorie et pratique sont très proches
… la solution unique à un problème n'existe pas
… on doit construire beaucoup d'objets
… on doit programmer en utilisant des standards
… il y a une certaine notion de l'exactitude
…

Plan

1. Introduction : qqs problèmes représentatifs, 2h
2. Arbres de recherche : plus courts chemins, 2h
3. Recherche de motifs, 3h
   Algorithmes de Rabin-Karp, de Knutt-Morris-Pratt, de Boyer-Moore
4. Algorithmique avancé en calcul evolutif, 3h
   ECGA, Estimation of Distrubution Algorithms
5. Informatique géométrique, 3h
   Algorithmique de courbes et de surfaces
6. Clustering, 2h
7. Algorithmes approximatifs, 3h
   Algorithmes approximatifs et aléatoires
   Problème de voyageur de commerce

Références

Cormen T.H, Leiserson C., Rivest R., *Introduction à l'algorithmique,* (trad.), Dunod, 1994.
Knuth D., *The Art of Computer Programming*, Addison-Wesley
Seminumerical Algorithms, vol. 2, 1969
Sorting and Searching, vol 3., 1973.

Cori R., Levy J., *Algorithmes et programmation*, http://www.enseignement.polytechnique.fr/profs/informatique/Jean-Jacques.Levy/poly/
Vivien F., *Algorithmique avancée*, Doc. Interne, 2005.

1. Introduction : qqs problèmes représentatifs

1. Stable Matching Problem [Gale, Shapley]
   National Resident Matching Program
   Scottish Foundation Allocation Scheme …
1. Interval Scheduling
2. Bipartite Matching
3. Independent Set
4. Competitive Facility Location

Wikipedia: http://en.wikipedia.org/wiki/Stable_marriage_problem

Stable Matching Problem

Could one design a job recruiting process, that was self-enforcing ?

Exemple : Les étudiants en informatique postulent à des stages. Chaque étudiant formule par ordre de priorité ses préférences par rapport à des entreprises, et chaque entreprise formule ses préférences par rapport aux candidats.

Existe-t-il des mécanismes pour assurer la convergence ?

Stable Matching Problem: auto-renforcement

The origin of SMP: Suppose that Raj has just accepted a summer job at CluNet. A few days later, the small start-up company WebExodus calls up Raj and offers him a job as well. Now, Raj prefers WebExodus to CluNet and it may cause him to retrack his acceptance of the CluNet offer and go to WebExodus instead.

CluNet offers to one of its wait-listed applicants, who promptly retracts his previous acceptance of an offer from software giant Babelsoft …and the situation begins to spiral out of control.

Suppose that Raj's friend Chelsea, destinated to go to Babelsoft but having just heard Raj's story, call up the people at WebExodus and says «*you know, I'd really rather spend the summer with you guys than at Babelsoft*» . They realize that they would have rather hired her than other student …

Qu'est ce qui provoque l'instabilité du système ?
Comment le résoudre?

Stable Matching Problem: auto-renforcement

More stable situation: consider another student, who has arranged to spend the summer at CluNet but calls up WebExodus and reveals that he, too, would rather work for them. But in this case, based on the offers already accepted, they reply «*No, it turns out that we prefer each of the students we've accepted to you, so we're afraid there's nothing we can do.*» Each of its top applicants confirms : « *I'm happy where I am* ».

Soit un ensemble de préférences venant des entreprises et des candidats, peut-on assigner chaque candidats à chaque entreprise, de telle sorte que pour chaque entreprise E et chaque candidat C qui n'est pas destiné à travailler dans l'entreprise E :

1) E préfère tous les candidats acceptés, plutôt que C ; ou
2) C préfère sa situation actuelle, plutôt que de travailler pour l'entreprise E.

Si cette situation est respectée, le système est stable.

Formulation du *Stable Matching Problem*

Soit un ensemble $M=\{m_1, \ldots, m_n\}$ de n hommes et un ensemble $W=\{w_1,\ldots w_n\}$ de n femmes. $M\times W$ dénote l'ensemble ordonné de toutes les paires possibles (m, w) où $m\in M$ et $w\in W$.

Une ***correspondance*** (***matching***) S est un ensemble ordonné de paires, chacune prise dans $M\times W$, de telle sorte que chaque membre de M et chaque membre de W apparaît dans au plus une paire de S.

Une ***correspondance parfaite*** (***perfect matching***) S' est une correspondance de telle sorte que chaque membre de M et chaque membre de W apparaît exactement une fois dans S'.

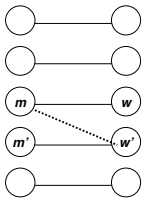
Une correspondance parfaite dans notre exemple : chaque homme est marié à une femme et personne n'est marié à plus d'une personne, et personne n'est célibataire ou polygame.

SMP – notion de préférences

Chaque $m\in M$ ordonne toutes les femmes ;
m préfère w à w' si m classe w avant w'
et chaque femme procède de la même façon.

Soit S une correspondance parfaite, qu'est ce qui peut rendre instable le système ?

Instabilité:
Il y a deux paires *(m,w)* et *(m',w')* ayant les deux propriétés suivantes: *m* préfère *w'* à w, et *w'* préfère m à *m'. Une paire (m,w') est instable par rapport à S : (m,w')* n'appartient pas à *S.*



But: trouver un ensemble de mariage sans instabilité !

Une correspondance est *stable* si (*i*) elle est parfaite et (*ii*) il n'y a pas d'instabilité par rapport à S.

SMP – Une correspondance stable

Etant donné des listes de préférences, peut-on construire efficacement une correspondance stable, si celle-ci existe ?

Exemple:
Supposons que nous avons un ensemble de deux hommes *{m,m'},* et un ensemble de deux femmes *{w,w'}.* Les listes de préférences sont les suivantes :

m préfère w à w'
m' préfère w à w'
w préfère m à m'
w' préfère m à m'

Il existe une correspondance stable unique : *(m,w)* et *(m',w').*

Question : (m',w) et (m,w') stable ?

SMP – Une correspondance stable

Exemple:

Les listes de préférences sont les suivantes:

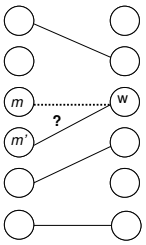
m préfère w à w'
m' préfère w' à w
w préfère m' à m
w' préfère m à m'

Il existe deux correspondances stables :

(m,w) et *(m',w') : les deux hommes sont satisfaits !*
et *(m',w)* et *(m,w') : les deux femmes sont satisfaites* !

SMP – Conception de l'algorithme

Fondements :
- Initialement, personne n'est marié
- *Fiançailles* – un état intermédiaire
- Un état dans lequel qq hommes et qq femmes sont célibataires et d'autres sont fiancés
- L'algorithme se termine quand plus personne n'est célibataire



13

SMP – Gale-Shapley algorithm

Initially all $m \in M$ and $w \in W$ are free
While there is a man m who is free and hasn't proposed to every woman
Choose such a man m
Let w be the highest-ranked woman in m's preference list to whom m has not yet proposed
If w is free then
(m,w) become engaged
else w is currently engaged to m'
If w prefers m' to m then m remains free
else w prefers m to m'
(m,w) become engaged
m' becomes free
Endif
Endif
Endwhile
Return the set S of engaged pairs

14

SMP – Analysis of Gale-Shapley algorithm

Remarques:
- **w est fiancée à partir du moment où elle reçoit sa première proposition; sa situation s'améliore de plus en plus (par rapport à sa liste de préférences).**
- **La séquence de femmes courtisées par m devient de moins en moins intéressante pour m.**
- **Propriétés :**
  - **L'ensemble S retourné à fin de l'algorithme est un correspondance parfaite.**
  - **L'ensemble S est une correspondance stable.**
- **L'algorithme G-S se termine après au pire n^2 itérations de la boucle *while*.**

15

SMP – Extensions

Correspondances stables multiples

Les listes de préférences sont les suivantes :
m préfère w à w'
m' préfère w' à w
w préfère m' à m
w' préfère m à m'

L'algorithme G-S algorithm se terminera pour la première correspondance stable (m,w), (m',w')!

***Injustement (unfairness)* l'algorithme G-S favorise les hommes :**

***« quelqu'un serait-il destiné à être malheureux »* ?**

Toutes les exécutions de l'algorithme G-S donnent-elles toujours les mêmes correspondances ?

16

Processus de conception de l'algorithme

Enoncer le problème avec une précision mathématique suffisante

Concevoir de cet algorithme pour le problème

Prouver que l'algorithme est correct

Analyser l'efficacité de l'algorithme

17

Algorithmes gloutons

Les algorithmes qui résolvent les problèmes d'optimisation parcourent en général une série d'étapes !

Un **algorithme glouton** (*ang. greedy*) fait toujours le choix qui semble le meilleur sur le moment dans l'espoir que ce choix mènera à la solution optimale globalement.

Choix : une règle

Exemple : location d'une voiture

but : satisfaire le max clients possible

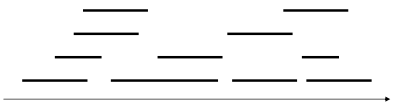
Demandes classées par dates de début croissantes

	e1	e2	e3
d	2	3	5
F	8	4	8

18

Cinq problèmes représentatifs

Ordonnancement de tâches



Une demande : puis-je réserver une ressource pour la période s, jusqu'à la date f ?

But: Maximiser le nombre de demandes acceptées

Ordonnancement de tâches pondérées

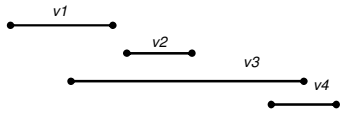
Chaque tâche (intervalle de temps) *i* est associée à *une valeur,* ou *un poids, v_i>0.*

But:
Trouver un sous-ensemble d'intervalles compatibles qui maximise la valeur totale.

Dans le cas $v_i=1$ $\Rightarrow$ algorithme glouton
Dans le cas de valeurs arbitraires $\Rightarrow$ *programmation dynamique*

Ordonnancement de tâches pondérées

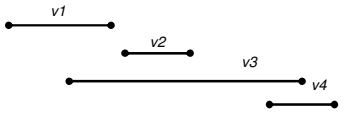
Structure de base et définition
- trier les tâches selon la date de fin
- définir une fonction *p* comme suit :
  - *p(1)* = 0
  - *p(i)* est le nombre de tâches que se terminent avant que la tâche *i* démarre



p(1) = 0
p(2) = 1
p(3) = 0
p(4) = 2

Ordonnancement de tâches pondérées

- Soit v_j le poids de tâche *j*
- Solution optimale pour l'ensemble de premières *j* tâches satisfait
  $OPT(j) = max \{v_j + OPT(p(j)), OPT(j-1)\}$
- Matrice supplémentaire *M[0…n]* initialisée à *0,p(1),…,p(n).*



p(1) = 0
p(2) = 1
p(3) = 0
p(4) = 2

Ordonnancement de tâches pondérées

Algorithme

For *j=1,…n* do
  Read *p(j) = M[j]*
  Set *M(j) = max {v_j + M(p(j)), M(j-1)}*

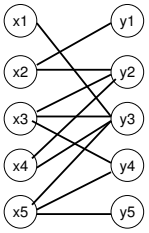
Complexité
- temps *: O(n log n)*
  - tri *: O(n log n)*
  - initialisation de *M[0..n] : O(n log n)*
  - algorithme : *n* opérations, *O(n)*
- mémoire *: O(n)* – matrice supplémentaire *M*

Correspondance bipartite

Un graphe *G(V,E)* est *bipartite* si son ensemble de noeuds V peut être partitionné en deux ensembles *X* et *Y* de telle sorte que chaque arête possède un sommet dans *X* et un autre dans *Y.*

Une correspondance dans un graphe est un ensemble d'arêtes $M \subseteq E$ possédant la propriété que chaque nœud soit le sommet d'au plus une arête de *M.*

Une correspondance parfaite

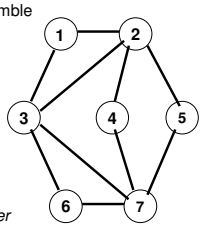


Exemples:
Hommes – Femmes
Professeurs – Cours
Emplois - Machines

Ensembles indépendants (EI)

Soit un graphe *G(V,E),* on dit qu'un ensemble de nœuds $S \subseteq V$ est *indépendant* si deux nœuds dans *S* sont reliés par une arête.

EI : Etant donné *G*, trouver l'ensemble indépendant le plus large possible.



Ensemble indépendant {1,4,5,6}
Largeur maximale : 4

Exemple: Un groupe d'amis invités à dîner

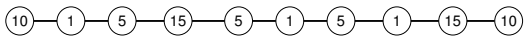
Actuellement, il n'existe pas d'algorithmes efficaces connus pour le problème EI !

25

Competitive Facility Location

Jeu: Deux larges sociétés – *JavaPlanet* et *Queequeg's Coffee* –mettent en place des franchises-café dans le pays et sont en compétition dans cette zone géographique.

Contraintes: deux franchises ne doivent pas être voisines



Qui va gagner ?

26

Competitive Facility Location

Soit un graphe $G=(V,E)$, où V est un ensemble de zones, et (i,j) est une arête dans E si les zones i et j *sont voisines*.

Les contraintes de zones: l'ensemble complet de franchises ouvertes doit former un ensemble indépendant dans G.

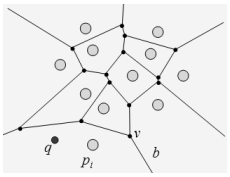
Existe t-il une stratégie pour P2 de telle sorte que, quelle que soit la stratégie de P1, P2 sera toujours capable de sélectionner un ensemble de nœuds dont la valeur cible vaut au moins B ?

Les problèmes PSPACE-complets sont plus difficiles que les problèmes NP-complets

27

The Voronoi Game

The Voronoi Game is a simple geometric model for the competitive facility location. In this case the Voronoi Diagram is represented by the bisector relative of the two sites on the plane. The bisector is the locus of points in the plane at equal distance from the two sites thereby it determinate two regions constituted by points that are at a closer distance to one of the sites than the other.



More formally defined a metric on the plane, a point q will be within the corresponding cell of the site pi if and only if the distance between q and pi is smaller than between q and any other site in the plane.

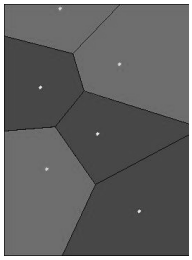
28

The Voronoi Game

The Voronoi game is played in its basic version by two players (Blue & Red) that insert in alternated mode a certain number n of points.

The scenario is divided by calculating the Voronoi Diagram relative to the points inserted and finally the winner is the player that has occupied the greatest area at the moment when all the points from both players are inserted.

The question is: Is there a strategy that allows one of the players to win?



29