TP d'Algo/Complexité/Calculabilité

Année 2010-11

Version 1.0

Université de Montpellier
Place Eugène Bataillon
34095 Montpellier Cedex 5

Rodolphe Giroudeau
161, rue Ada
34392 MONTPELLIER Cedex 5
TEL : 04-67-41-85-40
Mail : rgirou@lirmm.fr

Université de Montpellier II — 2010/2011
Algo/complexité/calculabilité — Durée: 6h

TP
TD – Séance n°

1 Descriptif des tâches

Vos résultats seront présentés en procédant à la rédaction d'un mémoire dont la qualité influencera la note finale. Ce manuscrit sera rendu le jour de la soutenance. La soutenance consiste à la présentation des résultats pratiques (choix du langage, choix des structures de données, résultats obtenus, tests sur un grand jeu de données, analyse de ceux-ci ...). Vous aurez 15 minutes au maximum questions comprises. Vous avez la possibilité d'utiliser des transparents.

2 Partie théorique

2.1 Partie algorithmique

Exercice 1 – Modélisation et résolution d'un problème d'ordonnancement par un problème de flot maximum : ordonnancement avec coûts dépendants des dates de début

L'ensemble des n jobs $\{J_1, \ldots, J_n\}$ de durées $p_1, \ldots, p_n$, reliés par des contraintes de précédence données par un graphe $G = (\{J_1, \ldots, J_n\}, E)$ doivent être ordonnancés dans la fenêtre de tempss $[0, T]$. Toutes les données du problèmes sont entières et nous ne considérons ici que les ordonnancements dans lesquels les jobs ne commencent qu'à des dates entières. Pour chaque instant t et chaque job J_i, nous disposons du coût ω_{it} associé au démarrage de J_i à t.

Dasn le cas général nous allons montrer que notre problème se réduit à la recherche d'une coupe minimale dasn un graphe G^* construit comme suit :

- Sommets. A chaque job J_i, et à chaque date de début possible (plus une unité) $t \in [0, T-p_i+1]$, est associé un sommet v_{it}. Soient de plus deux autrs sommets s et p qui nous servirons de source et de puits.
- Arcs d'affectation. A chaque job J_i est associée une chaîne d'arcs d'affectation

$$(v_{i0}, v_{i1}, v_{i2}, \ldots, v_{iT-p_i}, v_{iT-p_i+1})$$

.
- Arcs associés aux précédences. A chaque contrainte de précédence $(J_i, J_j) \in E$, on associe un ensemble d'arcs de précédence dans G^*, $(v_{it}, v_{jt'})$ pour $t + p_i = t'$.
- Arcs auxiliaires. La source s est connectée aux sommets v_{i0} et les sommets v_{iT-p_i+1} sont connactés au puits.
- capacités. La capacité d'un arc d'affectation $(v_{it}, v_{it+1}), \forall t \leq T - p_i$ correspond au coût de démarrage ω_{it} du job i à t. La capacité de tous les autres arcs est infinie.

1. Construire le graphe G^* pour $n = 3$, $T = 5$, $p_1, 1, p_2 = 2, P_3 = 1, E = \{(1,2); (1,3); (3,2)\}$ et les coûts suivants

i, t	0	1	2	3	4
1	0	2	5	0	1
2	1	1	2	4	–
3	1	10	2	3	3

2. Montrer qu'il existe une coupe dans G^* de capacité minimale de laquelle sort un et un seul arc d'affectation par job.
3. Montrer que l'on peut associer un ordonnancement réalisable (qui respectent toutes les contraintes) à toute cuope de capacité finie minimale dasn le graphe. Quel est le coût de cet ordonnancement ?
4. Montrer qu'à tout ordonnancement réalisable correspond une coupe dont la capacité est égale au coût de l'ordonnancement.

Exercice 2 – Coupes et chemins arcs-disjoints

Considérons le réseau donné par la figure 1. Tous les arcs admettent une capacité unitaire.

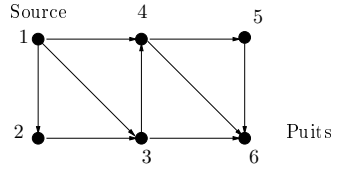

FIG. 1 – Réseau

1. Calculer le nombre maximum des chemins d'arcs-disjoints à partir de la source jusqu'au puits dans le réseau donné par la figure 1.
2. Enumérer tous les $s-t$ coupes dans le réseau donné par la figure 1. Pour chaque $s-t$ coupes $[S, \bar{S}]$, lister les sommets, les arcs avants et les arcs arrières.
3. Vérifier que le nombre maximum de chemins d'arcs-disjoints à partir du sommet source jusqu'au puits est égal au nombre minimum d'arcs avant dans une $s-t$ coupe.

2.2 Partie complexité

Exercice 3 – Sur quelques réductions

1. On vous demande de rappeler la réduction de SAT à 3-SAT.
   (a) ennoncer SAT et $3-SAT$
   (b) définir la réduction.
   (c) justifier alors que $3-SAT$ est $\mathcal{NP}$-complet (sachant que SAT est $\mathcal{NP}$-complet).

(d) Application : si un ensemble de clauses contient n_v variables, n_1 clauses à un littéral, n_2 clauses à 2 littéraux, n_3 clauses à 3 littéraux, n_4 clauses à 4 littéraux et n_5 clauses à 5 littéraux (et pas d'autre clause) combien le système obtenu par votre réduction contient-il de variables et de clauses ? vous devrez bien sûr justifier votre réponse.

2. Pourquoi le principe de la réduction ne permet-il pas de réduire $3-SAT$ à $2-SAT$ et de prouver ainsi que $2-SAT$ est $\mathcal{NP}$-complet ? (il ne s'agit pas d'expliquer pourquoi $2-SAT$ n'est pas $\mathcal{NP}$-complet, mais pourquoi cette réduction ne marche pas).
3. Il s'agit de prouver que $2-SAT$ est un problème polynomial. Vous avez un article en français expliquant cette preuve à
   http ://philippe.gambette.free.fr/SCOL/Graphes/
   (a) Vous commencerez par fabriquer trois ensembles de 2-clauses, le premier valide, le deuxième insatisfiable et le troisième contingent, et pour chacun de ces ensembles de clauses vous construirez le graphe correspondant. Vous expliquerez comment apparaît sur chacun des trois graphes la validité de l'ensemble de clauses correspondant.
   (b) Vous expliciterez ensuite l'algorithme de transformation et vous évaluerez sa complexité.
   (c) Vous expliciterez ensuite l'algorithme d'exploration du graphe et vous évaluerez sa complexité *en fonction de la taille de l'ensemble de clauses initial*.
   (d) enfin vous justifierez l'équivalence de la réponse au problème $2-SAT$ et au problème qui est posé dans le graphe.

2.3 Partie Calculabilité

Exercice 4 – Sur le problème de codage

1. Comment enumérer les couples d'entiers ?
2. Donner les fonctions de codage et de décodage $f_1(z) \to x$ et $f_2(z) \to y$.
3. Montrer que l'on peut coder les triplets. Généraliser aux k-uplets. Puis aux listes de longueurs quelconques.
4. Peut-on énumérer les fonctions C syntaxiquement correctes ? Et les fonctions C qui ne bouclent jamais ? Justifier vos réponses le plus clairement et le plus synthétiquement possible.

3 Partie pratique sur les algorithmes de flots

Exercice 5 – La méthode de Edmonds-Karp et celle de Dinic

1. Programmer une procédure qui construit à partir d'un graphe orienté valué (les valuations représentent les capacités)et deux sommets s et p du graphe , le graphe d'écart associé (correspondant à un flot nul sur chaque arc).
2. Programmer un procédure qui à partir d'un graphe orienté et deux sommets s et p donne un plus court chemin en nombre d'arcs de s à p ou qui signale si il n'y en a pas.
3. Étant donnés un graphe G orienté et valué et un chemin de G, écrire une fonction qui calcule l'arc de plus petite valuation sur le chemin.

4. Étant donnés un graphe d'écart, un chemin et un entier k, donner une procédure qui met à jour le graphe d'écart si on augmente le flot de k le long de la chaîne augmentante correspondant au chemin.
5. Écrire une procédure qui à partir du graphe initial et du graphe d'écart final (plus de chemins entre s et p donne la valeur du flot maximum ainsi que la valeur du flot sur chaque arc lorsque le flot maximum est atteint).
6. En utilisant les procédures et les fonctions précédentes, programmer l'algorithme de Edmonds-Karp.
7. Écrire une procédure qui prend en compte l'ensemble des plus courts chemins en nombres d'arcs.
8. Écrire une procédure qui calcule la plus petite valeur du flot dans le graphe de couche.
9. Écrire une procédure qui met à jour le flot dans le graphe G.
10. En déduire l'algorithme de Dinic.
11. Comparer les résultats (temps d'exécution, taux d'occupation mémoire) entre les deux méthodes. Vous apporterez un soin tout particulier à la génération des vos résultats et à leur présentation.

4 Barême

- $7pts$ partie théorique
- $8pts$ mémoire sur la partie 3,
- $5pts$ présentation.