

Master S&T Informatique, Univ. Bordeaux 1

Algorithmes du monde réel — Projet

6 octobre 2008

1 Modalités

Le projet comporte 2 parties, à réaliser par équipes de 3 étudiants. Les sources du projet, ainsi qu'un court rapport (en pdf, typiquement 5 pages) détaillant les choix effectués et les résultats obtenus sont à transmettre par mail à l'ensemble des enseignants selon le calendrier :

- 1^{re} partie à remettre le 12/11/2008.
- 2^e partie à remettre le 11/12/2008.

Chaque étudiant d'un groupe présentera en soutenance l'une des 3 parties et sera responsable de la rédaction de la partie correspondante du rapport. Le programme devra être écrit en langage C ou C++, et devra pouvoir être compilé par `gcc` ou `g++`. Il devra pouvoir être lancé par

```
./commande <fichier_graphe>
```

2 Objectif

L'objectif du projet est de réaliser des programmes résolvant des problèmes NP-complets sur des graphes. Pour tester le projet, on utilisera le générateur de graphes <http://dept-info.labri.fr/~gavoille/gengraph.c>. Le lancement de `./gengraph` (sans paramètre) fournit une aide en ligne.

Chacune des 3 parties aborde le problème sous un angle différent. Dans la première partie, on utilise des réductions pour se ramener au problème SAT que l'on résout en utilisant un *SAT-solver*.

3 Partie 1 : réductions

Dans cette partie, on pourra, pour fixer les idées, s'intéresser uniquement aux graphes non orientés. Les problèmes NP-complets que l'on souhaite résoudre sont les suivants :

1. k -colorabilité pour $k = 3$ et éventuellement d'autres valeurs.
2. Circuit Hamiltonien.
3. Couverture de sommets.
4. Clique.
5. Ensemble indépendant.

La liste n'est pas exhaustive. On pourra traiter d'autres problèmes, sur les graphes ou numériques.

Pour obtenir des algorithmes pour chacun de ces problèmes, on utilisera dans cette partie un *SAT-solver* (logiciel basé sur des heuristiques pour résoudre SAT) . Le principe est d'écrire, pour chacun de ces problèmes, un algorithme transformant une entrée E du problème considéré en formule Booléenne φ_E , en forme CNF, telle que la réponse au problème sur l'entrée E est OUI si et seulement si φ_E est satisfaisable. Autrement dit, la formule φ_E doit traduire l'existence d'un k -coloriage (pour le 1^{er} problème), d'un circuit Hamiltonien (pour le second), etc. Il s'agit donc de trouver et de programmer des réductions *inverses* de celles vues en cours : des problèmes que l'on veut résoudre *vers* SAT.

Pour déterminer si φ_E est satisfaisable (et si oui, une affectation des variables rendant φ vraie), on utilisera le SAT-solver `minisat` : <http://minisat.se>. Pour l'utiliser en ligne de commande, une documentation est disponible sur <http://www.dwheeler.com/essays/minisat-user-guide.html>.

On demande enfin, dans le cas positif, de retraduire l'affectation des variables rendant la formule φ_E vraie (`minisat` fournit une telle affectation), pour obtenir effectivement une solution au problème considéré : un k -coloriage, un circuit Hamiltonien, etc.

La ligne de commande

```
./commande <fichier_graphe> <numéro du problème> [paramètre]
```

donnera le résultat pour le problème dont le numéro est donné ci-dessus.

Par exemple,

```
./commande graphe.txt 3 4
```

pour tester si le graphe a une couverture de sommets de taille 4.

Indications pour les réductions

Soit $1, \dots, n$ les sommets du graphe d'entrée.

Pour tester l'existence d'une clique de taille K , on pourra utiliser $n.K$ variables $p_{i,j}$, $1 \leq i \leq n$ et $1 \leq j \leq K$. La variable $p_{i,j}$ sera vraie si le sommet i est le j^{e} sommet d'une clique.

Pour circuit Hamiltonien, on pourra utiliser de même des variables $p_{i,j}$ avec $1 \leq i, j \leq n$. La variable $p_{i,j}$ sera vraie si le sommet i est le j^{e} sommet d'un circuit Hamiltonien.

4 Partie 2 : approximations

Dans cette partie, on va s'intéresser au problème de minimisation *Vertex Cover* (VC, couverture par sommets), dont l'énoncé est rappelé ci-dessous :

Entrée : Graphe non-orienté $G = (V, E)$.

Sortie : La taille minimale de $U \subseteq V$ avec la propriété que chaque arête $uv \in E$ possède au moins une extrémité dans U , c'est-à-dire $\{u, v\} \cap U \neq \emptyset$. On appelle U une couverture.

4.1 VC et arbres

Dans cette partie on cherche d'abord un algorithme polynomial (même *linéaire*) qui résout le problème de minimisation de couverture sur les *arbres*¹.

1. On a mentioné en cours que VC peut être résolu en temps polynomial sur les graphes bipartis, en utilisant les algorithmes de couplage. Les arbres sont un cas particulier de graphes bipartis...

1. Montrez la propriété suivante :

Soit F une forêt (c'est à dire, une union disjointe d'arbres) et uv une arête telle que le degré de v est 1. Il existe alors une couverture de F de taille minimale qui contient le sommet u , mais pas v .

Indication : considérez une couverture U qui ne contient pas u , et construisez à partir de U une couverture U' de même taille qui contient u .

2. Proposez et implémentez un algorithme récursif qui calcule une couverture de taille minimale pour un arbre T . Le temps de calcul de votre algorithme doit être *linéaire*. En particulier, chaque sommet et chaque arête ne doit être considéré qu'un nombre constant de fois (c'est-à-dire indépendant de la taille de T).

Dans un deuxième temps, on propose l'algorithme *approché* suivant pour VC sur des graphes quelconques (connexes) G :

- a) Calculer un arbre T correspondant à une recherche en profondeur de G .
- b) Retourner l'ensemble S des sommets non-feuille de T . (On voit l'arbre T comme arbre orienté, et les sommets non-feuille sont les sommets qui ont au moins un successeur dans l'arbre.)

1. Justifiez que l'algorithme approché retourne une couverture.
2. On voudra comparer le temps de calcul et la taille de la solution retournée par 1) l'algorithme vu en cours et basé sur un couplage maximal et 2) l'algorithme ci-dessus. Implémentez les 2 algorithmes et testez-les sur quelques exemples.

Rappel : l'algorithme vu en cours calcule un couplage maximal, c'est à dire, un ensemble M d'arêtes tel que deux arêtes différentes n'ont pas d'extrémité en commun ; de plus, on ne peut pas rajouter une autre arête à M sans violer la propriété de couplage. L'algorithme retourne l'ensemble des extrémités des arêtes de M .

3. Trouvez un graphe sur lequel l'algorithme ci-dessus retourne une solution dont la taille est 2 fois la taille optimale d'un VC.
4. Justifiez que l'algorithme ci-dessus est 2-approché.

Indication : Montrez que G possède un couplage de taille $|S|/2$ et utilisez l'argument vu en cours.