Algorithmes du monde réel
Reductions de problèmes NP-Complets

Descombe Hervé Desenne Nicolas Sellier Xavier

12 novembre 2008

Table des matières

1 Pré-requis

Tout d'abord nous devons avoir notre amr.cpp dans le même répertoire que l'exécutable Minisat2.
Pour compiler notre programme il faut taper le commande suivant :

```
g++ amr.cpp -o amr
```

2 Notes sur le fonctionnement du programme

Pour connaître les arguments du programme, on peut trouver la liste des problèmes traités en lançant le programme sans argument. Il est à noter que le graphe passé en paramètre doit être de la même forme qu'un graphe généré par gengraph. Le deuxième argument correspond à un problème NP particulier, et le troisième correspond à la taille de la solution que l'on cherche, pour un problème particulier (cet argument est inutile pour un problème comme celui du circuit Hamiltonien).

Pour commencer, nous stockons les arêtes dans une matrice de booléens(en plusieurs phases, comme détaillé plus bas). Une valeur à la ligne i et la colonne j est vraie s'il existe une arête entre i et j. Le problème avec les graphes non orientés est que l'on a une matrice symétrique (donc pas forcément efficace en terme de stockage), mais c'est beaucoup plus aproprié pour l'étude de graphes orientés.

Puis nous travaillons avec la matrice obtenue sur le problème désiré. Notre programme lance des procédures différentes en fonction des arguments passés en paramètre. Typiquement, pour chacun d'entre eux, on applique un algorithme de réduction, dans lequel on va remplir un fichier destiné à être interprété par minisat. Ce fichier va contenir des entiers qui vont coder les représentations de nos graphes en valeurs booléennes. Une fois que l'on obtient ce fichier, on lance minisat sur ce fichier, et l'on récupère les valeurs retournées par minisat.

Enfin, on effectue une analyse des valeurs retournées. Pour certains problèmes, des opérations supplémentaires sont nécessaires.

3 Détails du parseur implémenté

Pour charger un graphe, nous avons implémenté un analyseur syntaxique qui va lire le fichier passé en premier argument du programme, le découper en arêtes, ces arêtes sont insérées dans un vecteur temporaire avant d'être stockés dans une matrice.

La syntaxe à utiliser pour entrer un graphe est assez simple. On peut soit entrer des listes d'arêtes séparées par un espace ou un retour à la ligne. On peut aussi entrer des chemins séparés aussi par un espace ou un retour à la ligne. Mais surtout pas d'espace en fin de ligne.

Voici un exemple de fichier de graphe :
0-1-2-3-4-5-6-7-8-9
9-10-11-12-13 13-14

Notre programme va lui même lancer minisat, analyser le résultat et afficher un résultat à l'écran. Il crée deux fichiers. Le fichier minisat.txt est le fichier contenant la réduction, c'est le fichier d'entrée de minisat. Le deuxième fichier, result.txt, est le fichier de sortie de minisat. Le programme n'efface pas ces fichiers car, si

l'utilisateur le souhaite, il est possible de rajouter l'inverse de la solution trouvée dans minisat.txt et de relancer minisat pour voir si il n'y a pas d'autres solutions.

4 K-coloriage

Le problème de la k-coloriabilité permet de déterminer si un graphe peut être colorié avec k couleurs sans que deux sommets voisins aient la même couleur.

Pour réduire ce problème à SAT, il faut autant de variables que de sommets multipliés par le nombre de couleurs.

Il faut que pour chaque sommet i et j tels que i et j soient voisins, $A_i \rightarrow \neg A_j$, A_i signifiant que le sommet i a la couleur A. L'implication peut être remplacé par un ou ce qui donne $\neg A_i$ ou $\neg A_j$.

Il faut aussi qu'un sommet n'ait qu'une seule couleur donc pour chaque sommet i, $Ai \rightarrow \neg Bi$ et $Ai \rightarrow \neg Ci$ ... Comme pour l'implication précédente pour être testée, on remplace l'implication par $\neg Ai$ ou Bi.

Il faut aussi qu'un sommet ait une couleur donc on a pour chaque sommet i A_i ou B_i ou C_i ...

5 Chemin et circuit Hamiltoniens

Pour réduire ces problèmes, nous avons fait le choix de suivre les conseils indiqués sur la feuille de TD et de considérer des variables représentant la position i d'un sommet j dans un chemin (circuit) Hamiltonien. Nous allons considérer tout d'abord le problème du chemin Hamiltonien, puis dans un second temps, nous nous pencherons sur celui du circuit, qui diffère très peu au niveau de la réduction.

5.1 Chemin Hamiltonien

5.1.1 Réduction vers SAT

Dans un graphe, un chemin est Hamiltonien si et seulement s'il passe une seule fois par tous les sommets de ce graphe. Ainsi, soit n le nombre de sommets d'un graphe G. Si ce chemin passe une seule fois par tous les sommets, il va donc être consitué de n sommets ; il y aura donc n positions possibles. On considère x_{ij} la variable boolénne indiquant si le sommet j est situé à la position i d'un tel chemin. Sur cette base, on peut définir un ensemble de règles permettant de spécifier à partir de variables booléennes si un chemin est Hamiltonien :

CH 1) $\forall j \in V(G)\,(x_{1j} \vee x_{2j} \vee \ldots \vee x_{nj})$

Cette première règle indique que chaque sommet est associé à au moins une position dans le chemin. C'est une condition nécessaire puisque le chemin Hamiltonien est censé parcourir tous les sommets.

CH 2) $\forall i$ avec $1 \Leftarrow i \Leftarrow n, (x_{i1} \vee x_{i2} \vee \ldots \vee x_{in})$

A l'inverse de la première règle, celle-ci indique que toute position dans un chemin Hamiltonien doit être associée à au moins un sommet.

CH 3) $\forall i, 1 \Leftarrow i \Leftarrow n, \forall \{j,k\} \in V(G), j \neq k, (\neg x_{ij} \vee \neg x_{ik})$

Cette règle spécifie que deux sommets ne peuvent pas se trouver à la même position dans un chemin Hamiltonien.

CH 4) $\forall i, 1 \Leftarrow i < n, \exists \{j,k\} \in V(G), j \neq k, j,k \in E(G), (x_{i,j} \wedge x_{i+1,k})$

Cette règle détermine que pour toute position i d'un chemin Hamiltonien, il existe un sommet adjacent à celui courant qui se trouve à la position *i+1* du chemin. C'est cette dernière règle qui permet d'eliminer beaucoup de cas dans lesquels on ne peut pas atteindre un sommet sans passer par ceux déjà visités (qui implique qu'ils sont à une position inférieure à la position courante). On peut remarquer que l'on n'a pas parmi ces quatre règles de restrictions explicite sur le fait qu'un même sommet peut se trouver à deux positions différentes. En fait, les règles 1), 2) et 4) impliquent cette restriction : on a n sommets et n positions ; si on attribue deux positions différentes à un même sommet, on va se retrouver avec un chemin à n positions mais *n-1* sommets. Or, la règle 4) précise que pour chaque position k de 1 à n, il existe un successeur différent. Pour que cette condition soit respectée, il faut donc qu'il y ait n sommets distincts dans le chemin Hamiltonien. Le fait de ne pas expliciter cette règle va en plus être utile pour le cas du circuit Hamiltonien.

Pour en revenir à cette quatrième règle, cette formulation ne convient pas pour une traduction triviale vers le format *minisat* (qui prend des conjonctions de disjonctions). On va donc modifier la formule afin de pouvoir la traduire sans trop de difficultés en disjonction de variables booléennes :

CH 4') $\forall i, 1 \Leftarrow i \Leftarrow n, \forall \{j,k\} \in V(G), j,k \notin E(G), (\neg x_{i,j} \vee \neg x_{i+1,k})$

Maintenant, on spécifie que si deux sommets ne sont pas adjacents, ils ne peuvent pas se trouver à deux positions consécutives dans le chemin.

5.1.2 Algorithme du programme

La fonction *hamiltonian_path* applique les règles explicitées ci-dessus sur notre matrice de booléens représentant les arêtes entre les sommets. Il n'y a en fait que la quatrième règle qui explore la matrice, puisqu'il faut vérifier s'il existe une arête entre deux sommets. Nous préférons ici expliquer la manière dont nous avons codé les variables booléennes pour les inclure dans le fichier destiné à être utilisé par *minisat*.

Comme il y a n positions et n sommets, on va avoir $n \times n$ configurations possibles, soit $n \times n$ variables booléennes dans notre fichier de sortie. On avait plusieurs possibilités de coder un sommet j à une position i. Nous avons choisi la manière suivante :

- les n premiers entiers codent les n positions pour le sommet 1 ; par exemple, si l'entier *2* est positif (soit que la variable qu'il représente est vraie), alors le sommet 1 se trouve à la deuxième position dans le chemin ; si l'entier est négatif, alors il ne se trouve pas à cette position ;
- les entiers de *n+1* à *2n* codent les positions pour le sommet 2 ;
- • • •
- pour un sommet k, ses positions sont représentées par les entiers situés dans l'intervalle [*(k-1)*n+1* ; *k*n*].

Typiquement, pour la première règle, on aura une première ligne dans le fichier résultat qui va signifier que le sommet 1 peut prendre au moins une des n positions dans le chemin. Pour un graphe de taille *4* :

```
1 2 3 4 0
```

L'algorithme de la fonction est découpé en quatre parties distinctes correpondant à l'application des quatres règles. Nous ne trouvons pas utile d'expliciter les autres détails de l'implémentation.

5.2 Tests et résultats

Nous avons pu tester la validité de nos règles et de notre algorithme sur des graphes de petites tailles.

Nous avons fait des tests de charge. Pour un graphe de 100 sommets, le programme met moins d'une seconde à traduire le fichier d'entrée en un nouveau fichier au format SAT. Pour un graphe de 500 sommets, par contre, il nécessite environ 1500 secondes d'exécution.

5.3 Circuit Hamiltonien

Le problème du circuit Hamiltonien est très proche de son prédécesseur. Dans ce cas, on ne cherche plus un chemin, mais un cycle qui passe une seule fois par chaque sommet du graphe. Toutefois, on peut considérer ce problème comme un cas particulier du chemin Hamiltonien à ceci près que les sommets situés en première et dernière positions doivent être adjacents. Ainsi, il suffit de rajouter cette règle à l'ensemble déjà défini. n correspond toujours au nombre de sommets dans le graphe, soit au nombre de positions possibles.

CH 5) $\exists k, j \in V(G), \{k, j\} \in E(G) et\, (x_{1j} \wedge x_{nk})$

Encore une fois, cette formulation ne s'adapte pas à la transformation vers le format SAT. On reformule la règle comme suit :

CH 5') $\forall j, k \in V(G), \{j, k\} \notin E(G), (\neg x_{1j} \vee \neg x_{nk})$

On remarque cependant que cette règle est un cas particulier de la règle 4) du chemin Hamiltonien.

Dans l'algorithme utilisé pour la réduction, cela se traduit par un petit ajout à effectuer dans le traitement de la quatrième règle. La fonction *hamiltonian_circuit* code cet algorithme. C'est la même que celle du chemin Hamiltonien exception faite de l'ajout à effectuer, qui se traduit par deux lignes à rajouter dans le fichier SAT à chaque fois que deux sommets ne sont pas adjacents.

Au niveau des tests, on a là-aussi testé sur des graphes de petites tailles, et en particulier sur des graphes qui possédaient un ou plusieurs chemins Hamiltoniens mais pas de circuit. Nous n'avons pas jugé utile de procéder à des tests de charge puisque la différence avec le problème précédent provient de quelques opérations supplémentaires s'exécutant en temps constant.

6 Clique

Dans un graphe, une clique est un ensemble de sommets tous reliés entre eux. Là-aussi, on va procéder à une réduction vers SAT.

6.1 Reduction vers SAT

Un peu à la manière du circuit Hamiltonien, on va utiliser des variables booléennes x_{ij} où i est le $j^{ième}$ sommet appartenant à la clique. Nous avons décidé d'utiliser cette représentation pour plusieurs raisons : d'abord, elle ressemble beaucoup à celle utilisée pour le circuit Hamiltonien, et même si les deux problèmes ne se ressemblent à priori pas, nous allons voir que les deux réductions sont assez similaires dans la forme. D'autre part, si dans le circuit Hamiltonien il nous fallait les positions des sommets, le deuxième indice dans ce cas permet surtout de faire ressortir le paramètre variable de la taille de la clique désirée.

En effet, dans ce problème, comme dans quelques uns qui suivent, il y a un paramètre supplémentaire à prendre en compte, qui est la taille du problème pour lequel on cherche une solution. Pour le circuit Hamiltonien, ce paramètre était en quelque sorte implicite, puisqu'on cherchait un chemin sur tous les sommets du graphe. Dans le cas présent, la taille est comprise entre 1 et n. Le problème de la clique revient à chercher un chemin entre plusieurs sommets tous reliés entre eux, et dont la taille est celle passée en paramètre. De ce fait, les règles de réduction ressemblent beaucoup à celles employées pour le chemin Hamiltonien. On va utiliser $n \times K$ variables.

D'abord, voici une variante de la règle CH 1) utlisée plus haut. n est la taille du graphe :

CL 1) $\forall k, 1 \Leftarrow k \Leftarrow n, (x_{1k} \vee x_{2k} \vee \ldots \vee x_{nk})$

Cette fois-ci, on vérifie que chaque position dans la clique est associée à au moins un sommet. Cette règle est plus souhaitable que la première car un sommet n'appartiendra pas forcément à une clique ; la première règle en serait donc faussée.

Nous reprenons ensuite la règle CH 3), qui spécifie que deux sommets ne peuvent pas être associés à une même position. Il faut ensuite définir une règle explicitant le fait qu'un sommet ne peut tenir qu'une seule et même position dans la clique.

CL 2) $\forall i \in V(G)$, $\forall k$, $l \Leftarrow K, k \neq l, (\neg x_{ik} \vee \neg x_{il})$

Ainsi, une position est obligatoirement associée à une seul sommet distinct, d'où on est sûr maintenant d'avoir comme solution un ensemble de taille K, si solution il y a. Il faut toutefois encore définir à quelle condition une telle association est possible. Pour cela, nous allons adapter la règle CH 4) qui stipule que s'il n'y a pas d'arête entre deux sommets, ils ne peuvent pas être consécutifs dans un chemin Hamiltonien. Dans le cas de la clique, comme il n'y a pas de notion d'ordre entre les sommets, nous allons simplement dire que s'il n'y a pas d'arêtes entre deux sommets, alors ils ne peuvent pas faire partie de la même clique.

CL 4) $\forall i, j \in V(G)$, si $\{i, j\} \notin E(G), \forall k, l \Leftarrow K$, $(\neg x_{ik} \vee \neg x_{jl})$

Un sommet est à la base une clique de taille 1 (lui-même). Par un raisonnement récursif, si un sommet est relié à tous les sommets d'une clique de taille k, alors il fait partie d'une clique de taille $k+1$.

Nous allons maintenant dire quelques mots sur le codage des sommets et des positions au sein de notre programme.

6.2 Algorithme du programme

Contrairement aux problèmes du k-coloriage et du circuit Hamiltonien, les n premiers entiers ne représentent pas les positions possibles pour le sommet 1, mais plutôt la position 1 pour tous les sommets. Ainsi, un entier x positif dans l'intervalle $\left[1,n\right]$ signifie que le sommet $x\%n$ se situe en première position dans la clique. Un entier y positif dans l'intervalle $\left[n+1, 2n\right]$ signifie que le sommet $y\%n$ est en deuxième position dans la clique.

C'est en fait le codage inverse à celui du circuit Hamiltonien. Nous avons préféré faire cette inversion pour des raisons d'ordre pratique. Tout comme pour le problème précédent, notre algorithme implémente les quatre règles définies ci-dessus.

6.3 Quelques tests

Données de tests	Résultats
0-3 3-1 1-2	-1 2 -3 -4 0
0-1 0-2 0-3 1-2 1-3 1-4 2-3 3-4	1 2 3 4 -5 0
0-1 0-3 1-2 1-3 2-4 2-5 2-6 3-4 4-5 4-6 5-6	-1 -2 3 -4 5 6 7 0

7 Ensemble indépendant

Un ensemble indépendant, dans un graphe, est un ensemble de sommets reliés entre eux par aucune arête, ce qui est l'exact contraire d'une clique, où les sommets y appartenant sont tous reliés deux à deux. De ce fait, puisqu'on a déjà étudié le problème de la clique, on peut réduire le problème de l'ensemble indépendant à celui de la clique, afin d'utiliser directement l'algorithme. Pour cela, il suffit de calculer le graphe complémentaire à celui sur lequel on travaille (soit que deux sommets reliés dans le graphe de départ ne le sont plus dans celui d'arrivée, et vice-versa). En effet, deux sommets reliés par une arête forment une clique à eux deux. Si l'on supprime cette arête, ils ne sont plus reliés et forment un ensemble indépendant. On peut généraliser cette remarque à un ensemble de n sommets. Si un ensemble de sommets sont tous reliés entre eux et forment un graphe complet (une clique), alors le graphe complémentaire sera un graphe sans arête (soit un ensemble indépendant de sommets).

L'implémentation de cette réduction vers le problème de la clique est triviale. Il suffit d'inverser les valeurs booléennes de notre matrice symbolisant les arêtes entre les sommets du graphe (soit une boucle sur toutes les cases de la matrice) et de lancer l'algorithme de réduction de la clique vers SAT.

8 Couverture de sommets

Tout comme celui de l'ensemble indépendant, on peut réduire le problème de la couverture de sommets vers celui de la clique. En effet, supposons que l'on a un graphe G pourvu d'une couverture de sommets de taille au plus k. Pour deux sommets i et j de G, au moins un des deux sommets se trouvent dans cette couverture. Prenons maintenant l'ensemble des sommets n'appartenant pas à la couverture. Soit x et y deux sommets de cet ensemble. Si aucun des deux n'appartient à cette couverture, alors ils ne sont reliés par aucune arête. Cela implique qu'ils le sont dans le graphe complémentaire, soit qu'ils forment une clique à eux deux. En généralisant le raisonnement à tous les sommets n'appartenant pas à la couverture de sommets du graphe de départ, cela revient à rechercher les sommets appartenant à une clique dans le graphe complémentaire. Ainsi, les sommets qui n'appartiennent pas à une clique dans le graphe complémentaire font partie d'une couverture de sommets dans le graphe de départ.

A l'instar de l'ensemble indépendant, on commence donc par inverser les valeurs de notre matrice, puis on applique la réduction de la clique vers SAT. La seule différence est que la taille de la clique passée en paramètre (*wanted_size*) est égale à la taille du graphe à laquelle on ôte la taille de la couverture de sommets désirée (car on cherche les sommets complémentaires).

L'analyse des résultats donnés par SAT diffère aussi quelque peu. On répertorie d'abord les sommets retournés par SAT. Le résultat de la couverture est donc l'ensemble des sommets du graphe n'appartenant pas à la solution.

9 Conclusion

La principale remarque que l'on peut faire est que l'on peut remarquer des similitudes entre toutes ces réductions. Certains problèmes, comme celui du k-coloriage, du circuit Hamiltonien ou encore de la clique, peuvent être réduits en définissant des règles de réduction très proches. D'autres, justement, se réduisent à certains de ces problèmes (ex : le problème de l'ensemble indépendant et de la couverture de sommets se réduisent facilement au problème de la clique). Cela renforce le caractère ensembliste des problèmes NP-complets, et montre qu'une réduction directe vers SAT n'est pas forcément la solution la plus facile. Il y aurait aussi à traiter le cas des graphes orientés. Nous aurions voulu traiter quelques cas, d'où l'utilisation dès le départ d'une matrice de valeurs booléennes (quitte à se retrouver dès le début avec une matrice symétrique du fait de coder des arêtes), mais nous avons manqué de temps pour le faire.

10 Annexes

Nous avons testé la k-coloration avec 3 graphes assez proches. Le premier graph est 2-coloriable. Le second est 3-coloriable et le dernier 4-coloriable. Nous avons vérifié qu'un graph 3-coloriable est bien 4-coloriable et pas 2-coloriable.

Nous avons testé le chemin et le circuit hamiltonien. Nous avons créé deux graphes assez proche des précédents. Le premier a un chemin hamiltonien et le second un circuit hamiltonien.

Nous avons testé que le circuit et le chemin soient bien détecté, ainsi qu'il n'y a pas de chemin avec le graphe qui a seulement un chemin, et finalement qu'il n'y a pas de chemin dans le graphe 2-coloriable.

Pour tester la clique nous avons créé un graphe qui a deux cliques, une de taille 3 et une autre de taille 4.

Nous avons vérifié que quand on demande une clique de taille 3 le résultat est bien les sommets 4,5 et 6 et que quand on demande une clique de taille 4 on a bien 0,1,2 et 3.

Pour la couverture de sommet, nous avons créé un graphe et son inverse.

On vérifie donc que l'on a bien un couverture de sommet de taille 2 dans le graphe et une clique de taille 3 dans son inverse contenant les sommets qui ne sont pas dans la couverture du premier.

Pour tester l'ensemble indépendant, nous avons utilisé les deux précédents, en vérifiant qu'une clique sur l'un est bien un ensemble indépendant sur l'autre