Rapport de projet

Implémentation de méthodes heuristiques pour le partitionnement de graphes

Paul-Emile Boutoille
Claire Pennarun
Tatiana Rocher

Bordeaux 1
Graphes et Recherche Opérationnelle

8 janvier 2013

Table des matières

Première partie

Présentation du programme

1 But

Ce programme permet d'utiliser différents algorithmes d'heuristique pour la résolution d'un problème de partitionnement d'un graphe en un nombre donné de classes de sommets. Ce programme permet ainsi de faire des tests de rapidité et de comparer la vitesse des algorithmes en fonction du nombre de sommets du graphe et du nombre de classes.

2 Mode d'emploi du programme

Le programme doit être utilisé sur des fichiers de la forme suivante :
nbSommets nbAretes
4 4
dmin dmax
1 3
source but valeur
1 2 1
1 3 1
2 3 1
3 4 1
sommet degre
1 2
2 2
3 3
4 1

La dernière partie du fichier (qui donne les degrés de chaque sommet) n'est pas nécessaire car elle n'est pas utilisée par notre programme.

Notre programme se lance avec les arguments suivants :
<fichier> <algorithme> <nbClasse> <option1> <option2>

Les deux derniers arguments ne sont nécessaires que dans certains cas, mais ne font pas planter le programme si ils sont présents.

- <fichier> représente le fichier sur lequel on veut lancer le programme, chaque fichier représentant un graphe.
- <algorithme> représente l'algorithme que l'on veut utiliser : "ex" pour l'algorithme d'exhaustion, "grad" pour l'algorithme de descente de gradient, "recuit" pour l'algorithme du recuit simulé, "tabou" pour l'algorithme tabou.

- <nbClasse> représente le nombre de classes de sommets voulu.
- <option1> représente le voisinage que l'on veut utiliser : "PnD" pour le voisinage Pick and Drop, "Swap" pour le voisinage swap. Cet argument n'est pas nécessaire pour lancer l'algorithme exhaustif mais obligatoire pour les autres.
- <option2> Est lié à l'algorithme :
  - pour l'algorithme du Recuit Simulé, <option2> est la température initiale (c'est un double)
  - pour l'algorithme Tabou, cet argument est la taille du tableau des mouvements tabous (int supérieur à 0).
  - pour les autres algorithmes, cet argument est inutile.

Deuxième partie

Choix généraux

3 Choix des algorithmes implémentés

Nous avons implémenté quatre algorithmes :
- Algorithme exhaustif
- Algorithme avec descente de gradient
- Algorithme de recuit simulé
- Algorithme Tabou

Les trois premiers algorithmes étaient obligatoires, et nous avons choisi comme quatrième algorithme l'algorithme Tabou, car la gestion d'un tableau de mouvements interdits nous paraissait intéressante. Les trois derniers algorithmes sont lancés plusieurs fois sur plusieurs solutions initiales. Nous détaillons plus loin cette fonctionnalité.

4 Langage de programmation

Nous avons choisi de programmer en Java, car c'est un langage que nous connaissons et qu'un langage objet nous paraissait pertinent pour gérer des objets comme des solutions ou des graphes. De plus, Java nous permet d'utiliser les fonctions liées à ses bibliothèques qui sont déjà optimisées. Nous savions donc que si un algorithme s'exécutait dans un temps supérieur à la normale, ce serait de la faute de l'algorithme et non des fonctions Java.

5 Structures de données

Pour stocker le graphe en mémoire, nous avons choisi d'utiliser une liste d'adjacence qui est remplie à la lecture du fichier par le programme. Cette liste est ensuite traitée par le programme pour créer un premier partitionnement (stocké dans l'objet GraphePartition), qui est la base de notre programme et sur lequel toutes les méthodes seront appliquées. Cet objet GraphePartition représente un graphe partitionné et contient notamment le numéro de classe attribué à chaque sommet, le nombre de classes prévues et l'évaluation de la solution courante.

6 Voisinages implémentés

Nous avons décidé d'implémenter deux voisinages, le Pick'n'Drop et le Swap. Ainsi l'utilisateur pourra choisir entre eux. Nous n'avons pas implémenté le voisinage par

Sweep car il revient à effectuer plusieurs Swap d'affilée.

7 Solution initiale

8 Evaluation

Troisième partie

Algorithme exhaustif - Enumération explicite

9 Description

Cet algorithme va examiner toutes les solutions possibles. Nous avons choisi d'effectuer un parcours en profondeur.

10 Paramètres

Il n'est donc pas nécessaire de choisir un voisinage. La solution initiale est une solution avec tous les sommets dans la première classe (0).

11 Tests effectués

11.1 Graphe à 5 sommets

nb de classes	2	3	4
temps (ms)	17	28	101

11.2 Graphe à 10 sommets

nb de classes	2	3	4	5
temps (s)	0.113	1	10	75

11.3 Graphe à 15 sommets

nb de classes	2	3
temps (s)	0.152	60

Graphe à 20 sommets / 2 classes : 17s
Graphe à 22 sommets / 2 classes : 46s

12 Résultats

C'est un algorithme qui devient très rapidement long, mais qui donne toujours la meilleure solution pour les petits graphes.

Il est utilisable quelque soit le nombre de classes jusqu'à 10 sommets et pour 2 classes jusqu'à 22 sommets.

12.1 Comparaison avec le temps théorique

Quatrième partie

Descente de gradient

13 Description

Cet algorithme part d'une solution initiale aléatoire et tourne tant qu'une solution meilleure a été trouvée dans son voisinage.

14 Paramètres

Cet algorithme dépend du type de voisinage de la solution. On cherche à chaque tour de boucle la meilleure solution voisine. Si celle-ci est meilleure que la solution courante, elle devient solution courante. Sinon, l'algorithme s'arrête.

Meilleure solution voisine : Afin de trouver la meilleure solution voisine nous examinons toutes les solutions voisines de la solution courante et retenons la meilleure.

15 Tests

16 Résultats

Cinquième partie

Recuit simulé

17 Description

Cet algorithme est basé sur la physique. On utilise la température comme paramètre. Pour chaque baisse de température, on cherche une solution courante optimale. A la température minimale, la solution trouvée doit être optimale.

18 Paramètres

Nous laissons à l'utilisateur le choix de la température initiale, c'est un double qui doit être supérieure à la température minimale.

19 Choix d'implémentation

Taille du problème Dans cet algorithme nous utilisons N, la taille du problème. Nous avons décidé de prendre le nombre d'arètes comme taille A COMPLETER (pk)

Solution initiale Comme pour les autres algorithmes, la solution initiale est aléatoire.

Température initiale Elle est donnée au choix à l'utilisateur. Celle-ci est un double qui doit être supérieure à la température minimale, 0.1° . Sachant que la baisse de température est assez rapide (voir suite) la température doit être assez conséquente, supérieure à 20° pour que l'algorithme tourne suffisamment longtemps.

Choix condition 1 Afin de sortir de la boucle, l'algorithme doit répondre à l'une des deux conditions suivantes :

- soit on a atteint la température minimale, fixée à 0.1°
- soit la température a changé 5 fois et la solution courante n'a pas changé.

Nous pensons que ces deux conditions sont importantes car ... A COMPLETER

Choix condition 2 Afin de sortir de cette deuxième boucle, l'algorithme doit répondre à l'une des trois conditions suivantes :

- soit on a fait N*N tours de boucle
- soit on a tiré 10*N solutions dont l'évaluation est plus grande que la solution courante

– soit on a accepté 10*N solutions dont l'évaluation est plus grande que la solution courante

A COMPLETER

Choix de la solution aléatoire POLO

Choix de la fonction g La fonction g est la fonction de chute de la température. Celle-ci est simple : on multiplie la température actuelle par la raison r. Au début de l'algorithme, r est choisie au hasard entre 0.7 et 0.9. A COMPLETER (dire pk)

20 Tests effectués

21 Résultats

Sixième partie

Algorithme Tabou

22 Description

Cet algorithme est une heuristique qui utilise un tableau de mouvements interdits (ou tabous) pour pouvoir sortir des minima locaux. On stocke les mouvements interdits plutôt que les solutions interdites par souci d'économie de la mémoire.

23 Paramètres

Le tableau de mouvements interdits (ou tableau tabou) est un tableau d'objets de type Mouvement. Sa taille est fixée par l'utilisateur à l'appel de l'algorithme.

Un mouvement est un objet qui contient plusieurs champs : deux champs représentant des sommets, un champ représentant une classe et un champ représentant le timestamp associé à ce mouvement. Cette représentation permet de gérer des mouvements de type Pick'n'Drop (en instanciant seulement un des deux sommets et le numéro de la classe) et des mouvements de type swap (en instanciant les champs correspondant aux deux sommets à échanger). Le timestamp de chaque mouvement, qui correspond à la durée de présence du mouvement dans le tableau des mouvements tabous, est incrémenté à chaque tour de boucle et le mouvement est retiré du tableau quand son timestamp atteint la valeur de la taille du tableau.

24 Tests effectués

Tests avec un tableau tabou de taille 5 et un partitionnement en deux classes :
20 sommets : 24 ms
50 sommets : 62 ms
100 sommets : 384 ms
500 sommets : 31 s
1000 sommets : 191 s

25 Résultats

Septième partie
Méthode de travail

26 Utilisation de git

Pour travailler sur ce projet, nous avons utilisé le logiciel de gestion de versions Git et sa plateforme d'hébergement web GitHub. Nous avons travaillé en mettant en place des branches de travail, qui étaient ensuite intégrées au code principal. Nous avons également utilisé le système des "issues" de GitHub, qui nous ont permis de lister les tâches à effectuer et donc d'avoir un suivi global du projet.

27 Séparation des t$\hat{a}$ches

Huitième partie
Conclusion