Partitionnement de graphe

Rapport de projet

Kévin Barreau Guillaume Marques

17 mars 2015

Résumé

L'objectif du projet est d'implémenter différentes méthodes de partitionnement de graphe, vu précédemment en cours, afin des les comparer. Il s'agit de méthode exhaustive, avec l'énumération, d'algorithme glouton, avec la descente de gradient, et de métaheuristiques, avec le recuit simulé, la recherche Tabou et l'algorithme génétique. Le langage de programmation pour atteindre ce but est libre (Javascript a été choisi pour ce projet). Les résultats sont présentés de deux façons : une comparaison des résultats obtenus par les différentes méthodes sur les mêmes instances de graphe, ainsi qu'une visualisation en temps réel de la meilleure solution trouvée par les méthodes.

Sommaire

1 Présentation du projet **3**
1.1 Description du problème 3
1.2 Méthodes de résolution 3
1.3 Choix de mise en œuvre 3

1 Présentation du projet

1.1 Description du problème

À partir d'un graphe non orienté, il faut partitionner le graphe en K classes au moyen de plusieurs métaheuristiques, de telle sorte que la somme des poids entre sommets n'appartenant pas à la même classe soit minimale. De plus il faut s'assurer que les sommets du graphe soient répartis de manière (à peu prés) équitable. C'est un problème NP-complet.

Nous avons choisi d'utiliser comme notion d'équité une représentation par un seuil de tolérance dans la différence entre la taille du plus grand cluster et la taille du plus petit cluster. Par exemple, pour un graphe de 10 sommets, un partitionnement en 3 classes avec une tolérance de 2 autorise une solution de la forme $\langle \mathbf{2}, \mathbf{4}, \mathbf{4} \rangle$ ($4 - 2 = 2$, inférieur ou égal à la tolérance) mais n'autorise pas une solution de la forme $\langle \mathbf{2}, \mathbf{3}, \mathbf{5} \rangle$ ($5 - 2 = 3$, strictement supérieur à la tolérance).

Le nombre de classe et la tolérance sont paramétrables.

1.2 Méthodes de résolution

Pour résoudre ce problème, nous avons implémenté plusieurs algorithmes.

— Énumération

— Descente de gradient

— Recuit simulé

— Méthode Tabou

— Algorithme génétique

L'énumération est une méthode exhaustive, qui parcours toutes les solutions possibles pour garder le meilleur résultat. La solution finale est la solution optimale du problème.

La descente de gradient est un algorithme glouton, qui, à partir du voisinage d'une solution, se déplace vers le meilleur résultat améliorant. On ne revient donc jamais sur une solution déjà visitée. La solution finale n'est pas forcément la solution optimale du problème.

Le recuit simulé, la méthode Tabou et l'algorithme génétique sont des métaheuristiques, cherchant une solution en essayant de ne pas avoir le plus gros problème de la descente de gradient : rester bloquer dans un optimum local. La solution finale n'est pas forcément la solution optimale du problème.

1.3 Choix de mise en œuvre

Pour implémenter les différentes méthodes de partitionnement de graphe, nous avons choisi d'utiliser le langage de programmation Javascript. Ce dernier apporte de nombreux avantages comme :

— Rapidité d'écriture (langage interprété à typage dynamique)

— Accessibilité (un navigateur web suffit pour l'exécution)

— Visualisation (HTML, CSS, Canvas, WegGL...)

On peut cependant lui reprocher une lenteur relative, comparé à des langages comme C++ ou Java. Dans un problème d'optimisation comme celui du partitionnement de graphe, on cherche toujours les meilleures performances possibles. Cependant, nous avons voulu abordé ce projet comme une introduction aux différentes méthodes et à leur comparaison, et non comme une implémentation la plus optimisée qui soit.

Les structures de données utilisées sont ainsi orientées vers le besoin d'affichage des résultats, non vers le besoin de performance. Les performances de mémoire sont principalement impactées, car plusieurs représentation du graphe sont utilisées (liste de sommets et d'arêtes pour la visualisation, matrice d'adjacence pour les algorithmes).

Ce choix de langage nous permet aussi d'apporter une interface graphique riche à moindre coût. Toutes les méthodes sont ainsi paramétrables par le biais de cette interface, sans avoir besoin de modifier et de recompiler le code.