Sorbonne Université
Faculté des Sciences et Ingénierie

Master d'informatique

4I201 - Résolution de problèmes

Rapport du projet : Métaheuristiques pour la résolution du problème de l'arbre de Steiner de poids minimum

B. Thanh Luong, 3504859
Hans Thirunavukarasu, 3605592

Encadrant :
M. Thibaut Lust

mai 2017

Table des matières

1 Introduction

Le problème de l'arbre de Steiner de poids minimum est NP-difficile. Étant donné un graphe connexe G = (V, E) et un ensemble T des sommets dits terminaux, nous allons chercher un ensemble des sommets non terminaux qui relient des sommets terminaux créant un arbre couvrant de poids minimum.
L'objectif de ce projet est d'utiliser 3 approches qui résolvent le problème en temps polynomial : un algorithme génétique, une heuristique de construction et une recherche locale.
L'algorithme génétique a été conçu de manière générale et aléatoire afin d'avoir une population variée des individus différents. Grâce à la variation de la population et aux différentes méthodes de génération d'une nouvelle population à chaque étape, la convergence vers un optimum local est rapide.
Nous allons utiliser deux heuristiques qui sont le plus court chemin et l'arbre couvrant minimum pour améliorer l'approche de problème.
Enfin, la recherche locale sera utilisé pour avoir une perspective globale de problème. Nous testerons nos algorithmes sur des instances des ensembles B, C, D, et E disponibles à http ://steinlib.zib.de/testset.php.

2 Algorithme génétique

2.1 Population initiale

N'importe quel individu (réalisable ou non réalisable) est codé par un vecteur binaire pour chaque sommet non-terminal, prenant 1 si le sommet est présent dans le graphe (ou la forêt dans le cas non réalisable) et 0 sinon.
La première phase consiste à construire une population aléatoire dont les individus ayant chaque bit une probabilité entre 0.2 et 0.5 d'être pris.
Ensuite, nous allons construire des générations suivantes à partir de la population initiale. Nous proposons 2 types de sélection des parents de la population courante ainsi que 2 stratégies de remplacement de population.
Sélection des parents : Parmi des individus, le moyen simple est de choisir celui qui a le meilleur score. Nous proposons une deuxième façon c'est pour chaque individu, nous le prenons comme parent si un tirage aléatoire est inférieur au ratio $\frac{best\ score}{son\ score}$.
Production des enfants : S'il y a un unique parent choisi, pour chaque bit, une probabilité entre 0.01 et 0.04 de le changer de 0 à 1 et inversement. Sinon nous faisons une opérateur de croisement, pour chaque pair de parents, nous utilisons le croisement au milieu pour reproduire deux enfants.
Remplacement de population : Le remplacement générationnel est d'écraser complètement la population courante par la nouvelle. Le remplacement élitiste est la sélection des meilleurs individus parmi les parents et les enfants avec une probabilité comme la sélection des parents.

Après avoir testé toutes les combinaisons des processus possibles, nous constatons que la convergence vers le minimum des croisements est très lente et aussi loin de l'optimum des problème. Nous allons rajouter dans la population initiale 2 individus spéciales qui sont les résultats des heuristiques du plus court chemin et de l'arbre couvrant minimum. Ces 2 individus sont pour but d'accélérer la convergence.

2.2 Heuristique du plus court chemin

Cette heuristique donne une des meilleurs approches dans la population initiale. Elle consiste à la construction un graphe complet des sommets terminaux. Si deux sommets terminaux ne sont pas connecter le coût de l'arête connectant deux sommets est égale au plus court chemin entre ces deux dans le graphe initiale. Une fois le graphe complet est fait, nous remplaçons toutes les arêtes "virtuelles" par le chemin réel, puis construire un arbre couvrant minimum tous les sommets du graphe et enfin enlever toutes les feuilles redondantes (i.e. les sommets non-terminaux de degré 1).
Cette heuristique est 2 - $\frac{2}{|T|}$ - OPT.

2.3 Heuristique de l'arbre couvrant minimum

Cette heuristique est moins efficace que celle du plus court chemin mais donne aussi une solution assez proche de l'optimum. Nous construisons l'arbre couvrant minimum de problème, puis la récursions est faite jusqu'à ce que l'on ne puisse plus enlever les feuilles redondantes.

2.4 Diversification la population initiale par randomiser des heuristiques de construction

La population aléatoire n'est pas efficace pour converger vers l'optimum. On converge très souvent vers celui du plus court chemin.
De ce fait, les individus aléatoires seront remplacés par ceux des deux heuristiques proposées en perturbant le poids des arêtes. C'est-à-dire, une nouvelle instance de l'arbre sera crée en changeant le poids associé aux arêtes de -0.2 à -0.05 ou de 0.05 à 0.2. Quand nous obtenons le résultat des heuristiques, nous reviendrons vers le problème initial des nouvelles instance.

3 Recherche locale

En partant de la population initiale générée par les heuristiques la randomisation des heuristiques des constructions, nous allons faire une recherche locale suivante :

Algorithme 1 : Recherche locale

Données : I : Meilleur individu de la population initiale
Résultat : Résultat de la recherche locale

```
faire
    Voisinage ← [ ]
    pour arête A ∈ non-terminaux faire
        si A prise dans I alors
            si degré(A) = 1 alors
                Nouveau voisin = I avec A non prise
                Voisinage = Voisinage ∪ { Nouveau voisin}
            sinon
                Nouveau voisin = I avec A non prise
                si Nouveau voisin connexe alors
                    Voisinage = Voisinage ∪ { Nouveau voisin}
                fin
            fin
        sinon
            Nouveau voisin = I avec A prise
            si Nouveau voisin connexe alors
                Voisinage = Voisinage ∪ { Nouveau voisin}
            fin
        fin
    fin
    si Meilleur candidat du voisinage a un meilleur score que I alors
        I ← Meilleur candidat du voisinage
        Non convergence
    sinon
        retourner I
    fin
tant que Non convergence;
```

4 Analyse des résultats

instance-OPT	Algorithme	Résultat	Variance	Écart-type	Temps d'exécution
b01 - **82**	PCC	82			0.01208
	CM	89			0.00199
	BP-E	82.0	0.0	0.0	0.14297
	BP-G	83.1	0.88999	0.94339	1.3402715
	MP-E	82.0	0.0	0.0	0.9216559
	MP-G	82.0	0.0	0.0	0.49829
	RAN	82			0.60108
	LOC	82			0.26112

5 Conclusion