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
B/b01.stp -	PCC	82			0.00451
	CM	83			0.00087
	BP-E	82.0	0.0	0.0	0.14385
	BP-G	82.0	0.0	0.0	0.53851
	MP-E	82.0	0.0	0.0	1.34391
	MP-G	82.0	0.0	0.0	1.25432
	RAN	82			0.28891
	LOC	82			0.31425
B/b02.stp -	PCC	90			0.00602
	CM	96			0.00102
	BP-E	89.0	0.0	0.0	0.12763
	BP-G	88.9	0.09	0.3	2.05193
	MP-E	88.4	0.64	0.8	1.04937
	MP-G	88.2	3.76	1.93907	1.54754
	RAN	83			0.95667
	LOC	83			0.49511
B/b03.stp -	PCC	140			0.0236
	CM	144			0.00074
	BP-E	138.8	0.16	0.4	1.00194
	BP-G	138.9	0.09	0.3	0.57525
	MP-E	138.0	0.0	0.0	1.4193
	MP-G	138.0	0.0	0.0	1.49279
	RAN	138			0.34568
	LOC	138			0.42526
B/b04.stp -	PCC	62			0.00575
	CM	63			0.00164
	BP-E	61.7	0.81	0.9	0.79119
	BP-G	62.0	0.0	0.0	0.16545
	MP-E	59.1	0.09	0.3	1.39856
	MP-G	59.6	0.84	0.91652	1.51993
	RAN	59			0.47931
	LOC	59			0.4395
B/b05.stp -	PCC	64			0.00806
	CM	68			0.00116
	BP-E	61.0	0.0	0.0	1.28889
	BP-G	61.0	0.0	0.0	1.2023
	MP-E	61.0	0.0	0.0	1.863
	MP-G	61.0	0.0	0.0	1.84786
	RAN	61			0.73086
	LOC	61			0.5936
B/b06.stp -	PCC	128			0.03117
	CM	133			0.00091
	BP-E	124.0	0.0	0.0	0.38745
	BP-G	124.0	0.0	0.0	0.2602
	MP-E	124.0	0.0	0.0	1.60832
	MP-G	124.0	0.0	0.0	1.43613
	RAN	124			0.67447
	LOC	124			0.72533

instance-OPT	Algorithme	Résultat	Variance	Écart-type	Temps d'exécution
B/b07.stp -	PCC	111			0.00889
	CM	127			0.00168
	BP-E	111.0	0.0	0.0	0.20408
	BP-G	111.0	0.0	0.0	4.94428
	MP-E	111.0	0.0	0.0	2.28961
	MP-G	111.0	0.0	0.0	2.33191
	RAN	111			1.05543
	LOC	111			0.80671
B/b08.stp -	PCC	104			0.01663
	CM	111			0.00139
	BP-E	104.0	0.0	0.0	4.2071
	BP-G	104.0	0.0	0.0	0.32796
	MP-E	104.0	0.0	0.0	2.08557
	MP-G	104.0	0.0	0.0	1.9711
	RAN	104			0.75536
	LOC	104			0.65521
B/b09.stp -	PCC	221			0.07283
	CM	226			0.00125
	BP-E	220.0	0.0	0.0	0.47687
	BP-G	220.0	0.0	0.0	0.34666
	MP-E	220.5	0.25	0.5	3.69861
	MP-G	220.3	0.21	0.45826	4.44612
	RAN	220			1.8433
	LOC	220			1.34599
B/b10.stp -	PCC	98			0.01338
	CM	104			0.00154
	BP-E	93.2	2.76	1.66132	6.96276
	BP-G	92.0	7.4	2.72029	40.21622
	MP-E	90.1	3.29	1.81384	6.39866
	MP-G	90.2	3.16	1.77764	5.86142
	RAN	86			1.78942
	LOC	86			4.18135
B/b11.stp -	PCC	93			0.02507
	CM	123			0.00183
	BP-E	91.6	0.64	0.8	10.33494
	BP-G	91.4	0.84	0.91652	24.65561
	MP-E	90.7	0.81	0.9	4.79364
	MP-G	90.5	0.65	0.80623	5.32104
	RAN	90			4.21122
	LOC	90			1.78671
B/b12.stp -	PCC	174			0.09496
	CM	181			0.0014
	BP-E	174.0	0.0	0.0	0.7846
	BP-G	174.0	0.0	0.0	4.98017
	MP-E	174.0	0.0	0.0	6.91646
	MP-G	174.0	0.0	0.0	8.1265
	RAN	174			5.16738
	LOC	174			4.10185

instance-OPT	Algorithme	Résultat	Variance	Écart-type	Temps d'exécution
B/b13.stp -	PCC	175			0.02811
	CM	189			0.00266
	BP-E	175.0	0.0	0.0	4.02466
	BP-G	174.4	0.84	0.91652	16.34604
	MP-E	173.5	1.85	1.36015	5.27839
	MP-G	173.3	2.01	1.41774	7.1027
	RAN	175			4.46725
	LOC	171			5.42496
B/b14.stp -	PCC	238			0.04599
	CM	250			0.00203
	BP-E	236.9	0.09	0.3	4.92103
	BP-G	237.0	0.0	0.0	0.64313
	MP-E	236.5	0.45	0.67082	6.04964
	MP-G	236.4	0.24	0.4899	7.23236
	RAN	236			7.36811
	LOC	235			4.66581
B/b15.stp -	PCC	325			0.19289
	CM	333			0.00174
	BP-E	319.0	0.0	0.0	6.53916
	BP-G	319.0	0.0	0.0	4.85837
	MP-E	318.6	0.24	0.4899	8.16159
	MP-G	318.4	0.24	0.4899	9.37917
	RAN	319			8.64159
	LOC	318			4.02937
B/b16.stp -	PCC	137			0.06532
	CM	169			0.00629
	BP-E	135.5	0.65	0.80623	154.63724
	BP-G	135.8	0.36	0.6	71.22357
	MP-E	135.8	0.16	0.4	10.71581
	MP-G	135.8	0.16	0.4	9.68463
	RAN	133			6.75747
	LOC	134			5.14052
B/b17.stp -	PCC	135			0.0631
	CM	146			0.00219
	BP-E	134.0	0.0	0.0	0.97911
	BP-G	133.4	0.84	0.91652	232.05891
	MP-E	133.0	0.8	0.89443	11.69888
	MP-G	133.0	1.0	1.0	10.24516
	RAN	133			12.37886
	LOC	131			8.81172
B/b18.stp -	PCC	222			0.24027
	CM	227			0.00179
	BP-E	219.8	0.36	0.6	16.5602
	BP-G	220.0	0.0	0.0	1.44029
	MP-E	218.3	0.41	0.64031	18.61842
	MP-G	218.9	0.69	0.83066	15.53312
	RAN	221			12.2912
	LOC	218			10.0428

instance-OPT	Algorithme	Résultat	Variance	Écart-type	Temps d'exécution
C/c01.stp -	PCC	88			0.00981
	CM	134			0.0178
	RAN	88			5.20974
	LOC	88			28.57226
C/c02.stp -	PCC	144			0.03966
	CM	194			0.01812
	RAN	144			2.6162
	LOC	144			39.4302
C/c03.stp -	PCC	779			2.35076
	CM	887			0.02155
	RAN	774			83.03549
	LOC	757			905.82514
C/c04.stp -	PCC	1105			5.78984
	CM	1163			0.01683
	RAN	1101			131.77565
	LOC	1089			902.8773
C/c05.stp -	PCC	1604			22.05328
	CM	1662			0.01204
	RAN	1587			384.18035
	LOC	1579			1012.31697
C/c06.stp -	PCC	60			0.01355
	CM	142			0.02329
	RAN	56			8.27837
	LOC	55			32.65182
C/c07.stp -	PCC	115			0.04561
	CM	186			0.02133
	RAN	103			19.24747
	LOC	115			69.87937
C/c08.stp -	PCC	534			3.55128
	CM	659			0.0196
	RAN	521			124.92103
	LOC	515			2845.07758
C/c09.stp -	PCC	727			7.88967
	CM	859			0.01777
	RAN	726			285.42667
	LOC	712			2744.46915
C/c10.stp -	PCC	1123			30.95651
	CM	1185			0.01521
	RAN	1106			532.90324
	LOC	1095			4611.81076
C/c11.stp -	PCC	35			0.03084
	CM	85			0.02522
	RAN	37			21.87121
	LOC	35			97.30895
C/c12.stp -	PCC	49			0.09542
	CM	79			0.02573
	RAN	48			30.71704
	LOC	47			304.05071
C/c13.stp -	PCC	275			6.24189
	CM	349			0.02818
	RAN	269			299.5505
	LOC	265			2495.9627

5 Conclusion