Métaheuristiques pour l'optimisation TP2

Deniz Sungurtekin

October 2020

1 Introduction

Le but de ce travail pratique est de nous familiariser avec "The Quadratic Assignment Problem" (QAP), un problème d'optimisation qui consiste à l'assignation d'un ensemble d'entité à un ensemble de position en connaissant la distance entre les positions et le flux existant entre les différentes entités. Ces données sont sous la forme de Matrice symétrique respectivement, D et W où D[r,s] désigne la distance entre la position r et s et W[i,j] désigne le flux entre l'entité i et j.

On veut donc positionner chacune de ces entités de manière optimal, pour cela on peut réduire le problème à la minimisation de la formule suivante:

$$\sum_{i,j=1}^{n} w_{ij} * d_{\psi(i),\psi(j)}$$

où ψ(i) donne la localisation de l'entité i.

Pour effectuer cela nous allons utiliser la liste tabu, elle sera de taille L et contiendra les configurations déjà visités afin d'éviter de repasser sur celles-ci durant L itération. Cependant nous avons introduit un facteur de diversification qui consistera à augmenter le coût d'une permutation si celle-ci est trop fréquente afin d'éviter de tomber toujours sur le même résultat sur le long terme.

Nous allons donc dans un premier temps, étudier la méthodologie utiliser pour résoudre ce problème et par la suite analyser l'impact que porte le facteur L et celui de la diversification sur nos résultats.

2 Méthodologie

Afin d'appliquer cette metaheuristique, il est nécessaire de définir exactement le voisinage d'une configuration. Dans notre cas, un voisin sera simplement cette même configuration à laquelle nous inter-changeons deux positions. Puisque cela correspond à la permutation de deux éléments dans un ensemble de taille n, on sait qu'une configuration possédera n*(n-1)/2 voisins.

Ensuite il est nécessaire de choisir le meilleur voisin, c'est à dire celui qui possède la fitness la plus basse, pour cela on applique la formule donnée dans le

point 1 sur tout les voisins, afin de calculer leur coût et choisir la moins élevé (En triant la liste de voisins de manière croissante selon leur coût). On peut donc par la suite choisir notre nouvelle configuration et stocker l'ancienne dans la liste tabu durant L itérations. Tant qu'une configuration se trouve dans la liste tabu, il est possible d'y retourner uniquement si cela mène à une nouvelle meilleur solution (Aspiration). J'ai pour cela définit une fonction simple qui renvoie True lorsqu'un voisin ne se trouve pas dans la liste tabu et une suite d'instruction qui permet le phénomène d'aspiration.

Pour introduire la notion de diversification dans mon implémentation j'ai choisis d'utiliser un dictionnaire qui calculera la fréquence d'apparition d'une configuration. Initialement à 1, elle s'incrémente à chaque fois qu'une même configuration est visitée. On ajoute ce nombre au coût d'un voisin lorsqu'on calcule sa fitness.

Finalement, il suffit de choisir une configuration initial aléatoire puis d'appliquer un certain nombre d'itération. Une fois le nombre d'itération maximal atteint la meilleur configuration ainsi que son coût est renvoyé à l'utilisateur.

3 Résultats obtenues

Voici la synthèse des 10 résultats obtenus lorsqu'il n y a pas l'implémentation de la diversification sur 20000 itérations avec chacune des valeurs de L, la taille de la liste tabu:

```
Pour L = 12: le meilleur solution est 578, on a une moyenne de 580.400000 et un std de 3.666061
Pour L = 6: le meilleur solution est 578, on a une moyenne de 583.000000 et un std de 6.767570
Pour L = 10: le meilleur solution est 578, on a une moyenne de 578.800000 et un std de 2.400000
```

Et voici le résultats lorsqu'il y a la diversification:

```
Pour L = 12: le meilleur solution est 578, on a une moyenne de 581.800000 et un std de 6.838128
Pour L = 6: le meilleur solution est 578, on a une moyenne de 582.000000 et un std de 4.000000
Pour L = 10: le meilleur solution est 578, on a une moyenne de 578.800000 et un std de 2.400000
```

En ce qui concerne l'impact de L, on remarque que cela n'influe pas sur la fitness de la meilleur solution trouvée sur 10 itérations, puisque les 6 résultats ci-dessus indiquent tous un coût optimal de 578. Cependant on remarque déjà une petite différence sur la moyenne des 10 meilleurs fitness, puisqu'elle est la plus basse lorsque L = 10, cela peut s'expliquer par la valeur de l'écart-type correspondant qui est moins grande. Ceci traduit le fait qu'avec L = 10 l'algorithme à explorer un cheminement plus similaire durant ces 10 itérations. De plus, puisque la tabu liste avec L = 12 est plus restrictif, l'algorithme a sûrement du choisir des voisins qui n'ont pas été sélectionnés avec L = 10 car il

ne représentaient pas la meilleur option. On en déduit que la longueur de L n'est pas forcément signe de stabilité dans un algorithme de recherche. Avec L = 6, on remarque une moyenne plus élevé qui traduit le fait qu'avec une tabu liste moins restrictif l'algorithme peut être plus volatile et converge moins souvent vers la solution optimal.

Pour ce qui est de la valeur de l'écart-type, qui mesure la dispersion des valeurs par rapport à la moyenne, on remarque qu'elle est plus élevée lorsque L = 6. On observe donc que celle-ci a tendance à favoriser une diversification du résultat dans le mauvais sens, car on observe une moyenne significativement plus haute dans ses valeurs de fitness obtenues. Globalement sans phénomène de diversification établit, on observe que plus la valeur de l'écart-type est haute plus la valeur moyenne est élevé.

Avec la diversification, on remarque que pour L = 12 l'écart-type est plus élevé et donc qu'il y a plus de diversité dans les résultats obtenus dans les dix itérations. Cependant, ces changements se traduisent par une augmentation de la moyenne mais ne change pas la meilleur solution trouvée sur 10 itérations. L'augmentation de la valeur de l'écart-type parait logique après avoir ajouté la diversification, puisque celle-ci force la recherche à choisir un voisin qui n'est pas le meilleur au court terme et donc peut potentiellement changer toute la trajectoire de la recherche. Cependant, on observe avec L = 10 et L = 6 que l'écart-type à diminuer ou n'a pas changer (L=10), ceci peut s'expliquer par le fait que même si l'algorithme est forcé de changer de trajectoire à un moment, cela ne change que très rarement la destination finale et que même si celle-ci diffère, elle diffère de très peu.

On conclut donc que pour cette exemple l'algorithme est le plus stable lorsque L = 10 et que ceci se remarque par une valeur d'écart-type très basse. De plus on remarque que la diversification n'a pas eu d'impact majeur sur la recherche d'une meilleur solution puisque tous les différents essais donne un résultat de 578. Cependant, on remarque que cela influe sur l'écart-type et la moyenne dans les deux sens selon la valeur de L choisie.