Université de technologie Belfort-Montbéliard

Problème de Transbordement

AG41 Optimisation et Recherche Opérationnelle

Auteur :
GLANGINE Geoffrey
SENE Victor

Professeur :
GRUNDER Olivier

30 mai 2016

1 Problème Mathématique

1.1 Les Paramètres du problème

N est un Noeud
f est un Noeud fournisseur
c est un Noeud client
r est un noeud de routage
b_i est le nombre de produit disponible sur un Noeud. Si c'est négatif, le Noeud est un Noeud client, Si c'est positif, le Noeud est un Noeud fournisseur, sinon le Noeud est un Noeud de routage
u_{ij} est la capacité d'un arc
c_{ij} est le coût fixe d'utilisation d'un arc
h_{ij} est le coût unitaire de transport
t_{ij} est le temps de transport
g_i est le coût unitaire de transport sur un Noeud
s_i est le temps de transbordement sur un Noeud
T est le délai max de f à t

1.2 Les variables

X_{ij} est le nombre de produits sur un arc ij
$Y_{ij} = 1$ *si* $X_{ij} > 0$

1.3 Fonction Objectif

$$min\ z = \sum_{i=1}^{n} \sum_{j=1}^{n} (c_{ij} \times y_{ij} + X_{ij} \times h_{ij} + g_j \times X_{ij})$$

1.4 Contraintes

$\forall i\ depot\ \sum_{j=1}^{n} X_{ij} = -b_i$
$\forall j\ client\ \sum_{i=1}^{n} X_{ij} = b_j$
$X_{ij} \leq u_{ij} \times y_{ij}$
$\forall i\ plateforme,\ depot\ \sum_{j=1}^{n} X_{ji} = \sum_{j=1}^{n} X_{ij}$
$\forall i, j, k \quad t_{ij} + s_j > T_{ik} \Rightarrow y_{ij} + yjk \leq 1$

2 Résolution

Afin de pouvoir structurer la recherche, nous avons préféré une méthode de résolution généraliste tel que le Branch and Bound car l'utilisation de celui-ci pour des problèmes légèrement différents serait possible. Mais le problème est que les algorithme d'optimisation "généralistes" sont généralement moins efficaces que les algorithmes dits "dédiés". Un algorithme de branch and bound a pour objectif de diviser un problème en sous-problèmes plus simples à résoudre. Cette division se fait sous forme d'énumération des solutions sous forme d'un arbre. Chaque branche est donc un nouveau problème à résoudre et en programmation cela se fait facilement en utilisant la récursivité. Mais l'algorithme de séparation-évaluation ne se contente pas de simplement énumérer le problème, il essaie aussi, dès qu'il est possible de "couper une branche à l'arbre" si il juge que cette branche ne mènera qu'a des solutions qui sont moins bonne qu'une solution que l'on a déjà trouvé au préalable. Suivant l'efficacité de ce choix, l'algorithme aura beaucoup moins de sous problèmes à évaluer et gagnera donc en temps d'exécution.

3 Implémentation

L'objectif ici est de montrer l'évolution de la réflexion au cours du projet. Tout d'abord nous pouvons agir sur plusieurs points afin d'accélérer la recherche : l'efficacité de la borne min, le choix d'une bonne première solution, l'ordre de visite des arcs en fonctions de leur coût. Ainsi nous avons commencé par chercher un moyen d'améliorer notre borne min.

3.1 Recherche d'une borne min

Tout d'abord, pour se familiariser avec le problème et l'algorithme, la borne minimale était calculée de manière très simple. En effet, la borne minimale était seulement l'évaluation de la solution en cours de création. Il n'y avait donc aucun calcul à faire. Cette borne min ne parait pas très efficace, ni très réfléchie, mais elle nous à permis de prendre connaissance de l'algorithme d'énumération et de faire une première coupe. Notre temps d'exécution s'est un peu amélioré, mais nous nous attendions et nous avons donc remarqué que le temps d'exécution n'était pas à la hauteur des attentes concernant le projet. Cette borne min n'était pas très subtile, mais elle nous à permis de réaliser une première coupe sans avoir à implémenter de calcul complexe. C'était simplement un petit essai pour s'assurer d'avoir compris le problème avant de commencer le gros travail.

Ensuite, nous avons réfléchi à un calcul de borne minimale plus conséquent. Nous avons donc choisi de calculer le coût de tous les produits qui transitaient des fournisseurs vers les plate-formes de transbordement et ensuite de calculer combien, au minimum nous coûterais le transfert des produits depuis les plate-formes, vers les clients. Nous regardons donc, parmis tous les arcs, lequel coûte le moins cher et si la plate-forme possède moins ou autant de produit que le client en demande elle envoie tout ses produits sur cet arc et on passe à la plate-forme suivante. Si il y a plus de produit sur la plate-forme que le client en demande, alors on cherche le second arc le moins cher.

C'est notre solution actuelle pour calculer notre borne minimale. Mais nous savons qu'elle n'est pas forcément la meilleure borne minimale pour ce problème. En effet, même si l'on regarde si la demande du client est plus grande que la disponibilité sur la plate-forme et que l'on cherche d'autres arcs pour envoyer le reste des produits au autres client si un arc est saturé, On enregistre pas quels clients sont déjà livrés en passant à la seconde plate-forme. Il est donc possible que ce calcul de borne minimal essaie d'envoyer des produits à un client depuis plusieurs plate-formes même si celui-ci est déjà satisfait. Ce problème fait que notre borne minimale est un peu moins haute que si elle était optimale, mais il n'influe en rien sur le résultat final. En effet ce n'est que si la borne minimale est trop haute que l'on peux "oublier" de passer par le résultat optimal. Par contre ce problème influe un petit peu sur l'efficacité de notre algorithme. Mais vis à vis de la difficulté à le mettre en place et du risque d'avoir une borne minimale trop haute, Nous avons décidé de ne pas changer notre méthode de calcul de borne minimale.

3.2 La première solution

Comme pour le calcul de la borne minimale, nous avons du passer par deux étapes pour calculer la première solution. En effet la fonction de calcul de première solution proposée dans les annexes fournies avec le projet ne donnait pas de solution correcte. En effet cette solution ne respectait pas la capacité des arcs. Donc nous avons commencé par créer une solution respectant la capacité des arcs. pour créer cette solution nous avons fait ceci :

```
Pour chaque Depot :
    Pour chaque arc :
        si(-depot.demande>=arc.capacité)
            depot.demande+=arc.capacité
            remplir l'arc jusqu'a sa capacité maximale
```

```
                sinon
                    remplir l'arc avec tous les produits disponibles
                    dans le depot
            Pour chaque plateforme:
                compter le nombre de produits qui arrivent sur cette plateforme
                satisfaire la demande client en commençant de la
                plateforme n1 vers le client n1
```

Mais une fois de plus cette première solution était naïve, et il fallait trouver quelque chose de mieux. C'est pour cela que nous avons décidé de choisir le chemin le moins cher depuis chaque dépôt pour aller au plate-formes. Nous estimons ce coût en fonction d'un ratio entre le nombre de produits transbordés, le coût unitaire et le coût fixe. Cette solution initiale donne des résultats qui ne sont pas très éloignés des solutions optimales pour les petits problèmes, mais il est une fois de plus possible de l'améliorer en implémentant l'algorithme de ford fulkerson qui est un algorithme utilisé pour le problème du flot maximum qui ressemble beaucoup à notre problème. Mais nous avons choisi de ne pas implémenter cet algorithme car sa difficulté d'implémentation ne parait pas vraiment justifiée pour simplement calculer une première solution.

3.3 Nos solutions

Nous avons remarqué que nos résultats approchent la solution optimale demandée mais ne la donne pas toujours car notre méthode d'énumération des chemins possibles n'est pas complètement fiable. Cependant pour un algorithme généralisé il nous semble que le temps de calcul des solutions est relativement raisonnable :

data/transshipment1.txt solution optimale 233, en 6 secondes
data/transshipment2.txt solution optimale 248 en 3 secondes
data/tshp006-01.txt solution optimale 3316 en moins de 1 seconde
data/tshp010-01.txt solution optimale 4756 en 20 secondes
data/tshp020-01.txt bloque à 9450
à partir des instances à 50 noeuds et plus le programme se bloque dès la première solution et ne trouve plus rien.

4 Des problèmes et des solutions

4.1 Les problèmes rencontrés

Nous avons décidé d'utiliser le code fourni pour nous baser dessus et le rendre rapidement opérationnel. Cependant le volume à assimiler que représentait déjà à la base le programme nous a fortement ralenti. Si nous avions à refaire le projet nous repartirions de zéro pour une meilleur maîtrise de l'ensemble du code.

4.2 Les optimisations possibles

Il nous est clairement apparu que nous pourrions améliorer notre projet en revoyant notre méthode d'énumération mais cela changerait aussi l'essence même du parcours du graphe impliquant des changements assez profond dans notre code. Il nous restera encore aussi à afficher un message d'erreur lorsque les solutions ne sont pas possibles.