

AG41 - Transbordement

Staine Florian

29 mai 2016

Chapitre 1

Objectif

L'objectif de ce projet est de résoudre un problème d'optimisation exacte, dérivé d'un problème de transport.

1.1 Problème

En plus de devoir transiter d'un fournisseur à un client, les ressources doivent passer par une plateforme de transbordement.

Chaque arc possède :

- une capacité maximale
- un coût fixe ainsi qu'un coût unitaire d'utilisation
- un temps de transport

Chaque plateforme possède :

- un coût unitaire de transbordement
- un temps de transbordement

Le coût total est constitué de la somme de plusieurs coûts :

En effet, si un arc transporte un ou plusieurs produit, son coût fixe sera pris en compte, en plus du coût unitaire multiplié par le nombre de produits transportés. Pour chaque produit transitant par une plateforme, un coût unitaire de transbordement sera ajouté au coût total.

Nous avons aussi d'autres contraintes que dans le problème initial.

Tout d'abord, les arc ont tous une capacité maximale, ce qui signifie que qu'il ne peut transiter plus de produit que la capacité maximale. Les arc sont utilisés au maximum une fois, ils ne peut pas y avoir des aller-retour pour transporter plus de ressources.

Un temps maximal de transport devra être respecté pour le transport de toutes les ressources. De plus, tous les produits partent au même moment et sont transportés en parallèles. Cela signifie que pour chaque cheminement emprunté pour atteindre les clients, les temps de transport plus le temps de transbordement doivent être inférieurs au temps maximum autorisé.

Chapitre 2

Modèle mathématique

2.1 Paramètres

n : nombre total de nœuds

n_f : nombre de nœuds fournisseurs

n_p : nombre de nœuds plates-formes

n_c : nombre de nœuds clients

b_i : demande ou disponibilité du nœud

g_i : Coût unitaire de transbordement

s_i : Temps de transbordement

$u_{i,j}$: Capacité de l'arc (i, j)

$c_{i,j}$: Coût fixe de l'arc (i, j)

$h_{i,j}$: Coût unitaire de l'arc (i, j)

$t_{i,j}$: Temps de transport de l'arc (i, j)

2.2 Variables

$x_{i,j}$: nombre de produit transportés par l'arc (i, j)

$y_{i,j,k}$: Nombre de produits transportés par la route $(i, j) (j, k)$

2.3 Objectif

$$\min z = \sum_{i=1}^{n_f}, \sum_{j=1}^{n_p}, x_{i,j} \neq 0 (x_{i,j} \cdot h_{i,j} + c_{i,j} + x_{i,j} \cdot g_j) + \sum_{j=1}^{n_p}, \sum_{k=1}^{n_c}, x_{j,k} \neq 0 (x_{j,k} \cdot h_{j,k} + c_{j,k})$$

2.4 Contraintes

$$(C1) : \forall (i, j) \in n^2, x_{i,j} \leq u_{i,j}$$

$$(C2) : \forall i \in n, \sum_{j=1}^n (x_{i,j} - x_{j,i}) = -b_i$$

$$(C3) : \forall (i, j) \in n^2, \sum_{k=1}^{n_c} y_{i,j,k} + \sum_{k=1}^{n_f} y_{k,j,i} = x_{i,j}$$

$$(C4) : \forall (i, j, k) \in n^3, y_{i,j,k} \neq 0, t_{i,j} + t_{j,k} + s_j \leq T$$

Chapitre 3

Résolution algorithmique

3.1 Algorithme utilisé

Pour résoudre ce problème, un algorithme de type Branch and Bound sera utilisé. Cela consiste à séparer en deux l'ensemble des solutions selon s'il ont fait un choix ou son contraire. Certains choix sont évités s'ils possèdent une borne minimal de l'objectif plus grand qu'une solution déjà existante.

3.2 Problèmes à résoudre

En premier lieu il est nécessaire de calculer une première solution. Cela permet d'éviter toutes les solutions plus mauvaises que cette première. Plus cette première solution sera proche de la solution finale, plus il sera facile de converger rapidement vers la solution optimale car un plus grand nombre de choix seront évités.

Ensuite, une borne minimum la plus élevée possible doit être définie pour permettre d'éliminer les solution moins efficace que la solution déjà existante. Tout le temps de résolution du problème est dépendant de ces deux algorithmes, qui sont le choix d'une première solution efficace et le calcul d'une borne inférieure.

Ensuite, il est possible d'éliminer certains choix lorsqu'il n'est pas possible de résoudre le problème à partir d'un sous-ensemble en solutions, cela permet encore un gain de temps non négligeable dans la résolution.

Chapitre 4

Résolution informatique

4.1 Langage utilisé

Pour la réalisation du projet, le langage Haskell (<https://www.haskell.org/>) à été utilisé. Il s'agit d'un langage fonctionnel pur, ce qui est une approche totalement différente de la programmation orientée objet.

Le compilateur utilisé est le plus courant pour ce langage : GHC (<https://www.haskell.org/ghc/>)

Le but était de découvrir la programmation fonctionnelle, ainsi que ses avantages et inconvénients, à travers un projet concret, car c'est un paradigme que l'on n'a pas l'habitude de voir dans nos différents cours.

4.2 Solution initiale

Il n'y a pas à proprement parler de solution initiale trouvée grâce à une méthode différente, cependant, le critère de choix sur les arcs n'est pas le même selon si on a déjà trouvé une solution initiale ou pas.

En effet, les chemins sont choisis de manière à maximiser le nombre de ressources transportés, ce qui permet de converger plus rapidement vers une solution initiale, même si celle-ci n'est pas d'excellente qualité.

Même si le principe de l'algorithme est évidemment de couper le plus haut possible dans l'arbre des solutions pour éviter de perdre du temps dans des solutions loin de l'optimum, l'ordre des choix n'a pas d'incidence sur la meilleure solution trouvée du fait que toutes les possibilités sont envisagées puis testées.

Lorsqu'une solution initiale a été trouvée, l'algorithme se concentre sur la recherche des branches les moins chères permettant de se diriger plus rapidement vers la solution optimale.

4.3 Borne minimale

La borne minimale est calculée en additionnant le coût de la solution partielle ainsi que le coût minimum de transport pour les nœuds qui ne sont pas encore vides.

Pour estimer le coût des nœuds, les arcs sont triés suivant leur coût de transport. Ensuite, on prends les chemins par ordre du moins chère au plus chère jusqu'à ce que toutes les ressources soient transportées.

4.4 Solutions déduites

Le principe du Branch and Bound est de réduire l'ensemble des solutions en limitant la liberté sur certaines variables.

Ici, la recherche de chemins est limitée par une liste de chemins évités, chemins que l'on ne peut pas emprunter.

Lors de la recherche de chemins, si tous les chemins évités ne peuvent pas être sélectionnés.

4.5 Bugs, limites et problèmes rencontrés

Lors de la recherche de chemin à emprunter, toutes les possibilités ne sont pas testées. En effet, seul la valeur maximale d'utilisation d'un chemin est testée, et si celle-ci fait partie de la liste des chemins évités, elle est retirée.

Cependant, il est possible qu'un même chemin avec une quantité transportée inférieure soit utilisable mais ne soit pas considérée dans les chemins possibles.

En considérant aussi ceux-ci, le temps de calcul est souvent beaucoup plus long et ne permet, dans un grand nombre de cas, même pas d'obtenir une solution initiale.

L'algorithme actuellement en place est souvent trop lent pour permettre de trouver des solutions aux problèmes de taille supérieure à 50 nœuds. Cela vient en partie de la relativement faible efficacité de l'élimination des solutions, ce qui conduit à l'évaluation d'un grand nombre de possibilités.