Université Libre de Bruxelles
Faculté des Sciences
Département d'informatique

Coût minimum de transport multi-articles

Professeur : Dimitrios Papadimitriou

Cours : INFO-F310 – Algorithme et recherche opérationnelle

Remis le 17 mai 2024

LEJEUNE Quentin 000587045 passerelle M-INFO
BAHJA Ali 000537649 B-INFO
Année académique 2023-2024

Table des matières

1 Introduction

Ce rapport scientifique étudie le problème du *coût minimum de transport multi-articles*. Ce problème est une extension du problème de coût de transport, ajoutant une couche de complexité.

Le sujet est abordé selon deux approches : **agrégée** et **désagrégée**. Nous mettrons en évidence les différences entre ces approches.

Nous commençons par définir le modèle mathématique dans un programme linéaire, qui est ensuite résolu à l'aide de l'outil GLPK[1], dont nous expliquerons brièvement l'utilité.

Ensuite, nous comparons les résultats obtenus pour quatre instances de problèmes différents et les déductions tirées des observations.

2 Définition du problème

Le problème étudié ici est une version du *problème de transport*.
Dans cette version spécifique, qu'on nommera *problème de transport multi-articles avec nœuds intermédiaires*, ou *TMANI*.

Le but est toujours de minimiser le coût de transport, mais cette fois-ci le problème ne se limite plus au transport d'un unique article de n sources vers m destinations. Le problème est étendu au transport de k commodité, avec $k \geq 1$, de n sources (ou producteur) vers m destinations (ou consommateur) avec en plus la possibilité de transité par 0 où plus nœuds intermédiaires.

Le transport d'articles entre deux nœuds n'est pas limité directement en taille, mais en coût. Tant que la production et le flux le permet, la capacité de transport peut être infinie tant que ce montant n'est pas négatif.
Il faut cependant bien prendre en compte les coûts de transports de ces articles.

Si l'on crée un exemple de ce problème n'utilisant qu'une seule commodité et aucun nœud intermédiaires, nous retombons sur le *problème de transport* classique.

Cette variation du problème ajoute plusieurs complexités :
- L'ajout de nœuds intermédiaires permet au flux des commodités d'emprunter différents chemins, avec des coûts différents, menant à une même destination.
- Une source peut désormais produire 1 ou plusieurs articles différents.
- Une destination peut désormais nécessiter 1 ou plusieurs articles différents.
- Chaque nœud source ou destination peuvent produire ou consommé un nombre variable x d'articles, $1 \leq x \leq k$ articles

Afin de répondre à ce problème, deux approches sont étudiées.
La méthode *agrégée* (noté *TMANIA*), dans laquelle la composante de multiples commodités est agrégée en une seule.
Cela se traduit, pour les nœuds sources et destinations, à l'addition des différentes productions ou demandes.
Quant aux coûts de transports entre deux nœuds, la médiane de chaque coût sera utilisé.

La seconde méthode, *désagrégée* (noté *TMANID*), consiste à appliquer toutes les contraintes du problème sans modifications des données et réponds donc exactement au problème initial *TMANI*.

3 Modélisation

3.1 Les données du modèle

Afin de modéliser ce problème *TMANI*, nous utilisons un graphe dirigé G tel que $G = (V, E)$ avec V l'ensemble des sommets du graphe et E l'ensemble des arrêtes du graphe.

De plus, K représente la liste des articles de telle sorte que $k \in \{k_0, k_1, \ldots, k_n\}$ pour n articles.

Les nœuds

Il existe trois types de nœuds dans cette variation de problème :

- **Les nœuds sources** produisent les différents articles. Ils sont donc associés avec une valeur de production p par articles k, tel que $p_i^k \geq 0$.
- **Les nœuds destinations** consomment, et donc requiers, différents articles. Eux aussi de telle sorte que consommation$^k \geq 0$
- **Les nœuds intermédiaires** ne stockent pas de commodités. Ils ne servent que de points de redirections des articles. Toutes commodités qui entrent dans un nœud intermédiaire doivent donc être renvoyé vers une autre destination.

Rappel : avec l'approche agrégée, la production des nœuds sources et la consommation des nœuds destinations est sommé afin d'avoir une production et consommation d'article unique.

Les arcs dirigés

Les arcs représentent les chemins possibles des différents articles. Chaque arc est associé à un coût par article des k-articles et le coût peut être aussi bien positif que négatif.

Le coût de transport de l'objet k du nœud i vers j est représentait par la notation c_{ij}^k.
Dans la version agrégée, on omet à nouveau l'indice supérieur k. Le coût de transport entre deux nœuds devient la médiane des k-coûts des articles.

La notation f_{ij}^k représente le *flux* de l'objet k transporté du nœud i vers j.
Dans la version agrégée, on omet encore l'indice supérieur k et c'est la somme des flux des k-articles qui est utilisé.

Par exemple, pour le coût de transport du nœud i vers j avec les couts des k_1, k_2, k_3 articles étant respectivement, $100, 125$ et 200. La médiane et donc le coût résultant de cet arc utilisé durant la méthode agrégée sera 125.

K représente le *vecteur des articles* à transporter $k \in \{k_0, k_1, \ldots, k_n\}$ pour n articles.
Dans l'approche *agrégée*, du problème, il n'existe donc qu'un article k_0.

3.2 TMANIA

Regardons d'abord à l'approche *agrégée* du problème

Fonction objective

L'objectif du problème étant de minimiser le coût de transports des articles, nous pouvons représenter la fonction objective comme :

$$\text{min.} \quad \sum_{(i,j)\in E} c_{ij} f_{ij} \tag{1}$$

Avec c_{ij} représentant la médiane des cours pour les k-articles concernés étant donné l'approche agrégée.

Le but étant de minimiser le coût de transport de l'unique article des sources vers les destinations, la fonction objective tente de minimiser la somme de tous les coûts multiplié par le flux de l'article sur chaque arc.

Conservation des flux

Le modèle doit maintenir la conservation des flux. Autrement dit, les flux entrants et sortant doivent rester cohérent aux types de nœuds.

Un nœud source *produit* des articles. Il en sortira donc un plus grand nombre qu'à l'entrée (entrée $<$ sortie).

Un nœud destination *consomme* des articles. Il en sortira donc un plus petit nombre qu'à l'entrée (entrée $>$ sortie).

Enfin, **un nœud intermédiaire** ne produit et ne consomme aucun article. Il en sortira autant d'article qu'il y sera entré (entrée $=$ sortie).

Nous pouvons formuler cela sous la condition suivante :

$$\sum_{i:(i,j)\in E} f_{ij} - \sum_{i:(j,i)\in E} f_{ji} = b_i \begin{cases} < 0 & \text{si } i \text{ est une source} \\ > 0 & \text{si } i \text{ est une destination} \\ = 0 & \text{si } i \text{ est intermédiaire} \end{cases} \tag{2}$$

Pour tout $i \in V$.

L'équation 2 représente la conservation des flux de l'approche agrégée dans l'entièreté du système. L'on distingue trois résultats possibles pour b_i^k :

- **Inférieur à zéro** signifie qu'il s'agit d'une source et qu'il envoie plus de commodités qu'il n'en reçoit. Une source pouvant avoir un arc dirigés une autre source, la valeur d'entrée peut également être inférieur à la valeur de sortie.
- **Supérieur à zéro** signifie qu'il s'agit d'une destination, donc que le nœud reçoit plus de commodités qu'il n'en distribue, afin de satisfaire sa propre consommation.
- **Une valeur égale à zéro** représente un nœud intermédiaire, ne pouvant stocker aucune commodité, tout ce qui entre dans un nœud intermédiaire ressort.

Faisabilité du flux

Afin qu'un flux contenu dans le modèle soit réalisable, il doit être contraint par la capacité de l'arc concerné.

Dans notre contexte, les arcs eux-mêmes n'ont aucune limite positive, seulement une limite de non-négativité.

Cependant, un nœud ne peut pas transférer plus de commodités qu'il ne reçoit et produit ou consomme. Cette contrainte est représentée par l'équation suivante :

$$f_{ij} \leq b_i \quad \forall (i,j) \in E, i \in V \tag{3}$$

3.3 TMANID

Dans cette section, nous allons voir comment la fonction objective, les contraintes et variable du modèle s'appliquer pour l'approche *désagrégée* du problème.

Fonction objective

L'objectif, semblable à celui de l'approche agrégée, est toujours de minimiser le coût total de transports des articles, mais cette fois avec des coûts de transport d'articles dépendant de chaque article. Cela se traduit par l'ajout de la composante k dans la fonction objective, tel que :

$$\text{min.} \quad \sum_{k} \sum_{(i,j) \in E} c_{ij}^{k} f_{ij}^{k} \tag{4}$$

Où c_{ij}^{k} représente les coûts de déplacement du nœud i vers j de l'article k.
Et f_{ij}^{k} représente quant à lui le flux entre ces deux mêmes nœuds de l'article k, autrement dit : **la quantité**.

Nous avons donc bien l'objectif de minimiser le coût de tout déplacement d'article au travers du système.

Conservation des flux

La conservation des flux est, elle aussi, semblable à la méthode agrégée, mais cette fois en prenant en compte les différents articles :

$$\sum_{i:(i,j) \in A} f_{ij}^{k} - \sum_{i:(j,i) \in A} f_{ij}^{k} = b_i^{k} \begin{cases} < 0 & \text{si } i \text{ est une source} \\ > 0 & \text{si } i \text{ est une destination} \\ = 0 & \text{si } i \text{ est intermédiaire} \end{cases} \tag{5}$$

$\forall i \in V, \quad \forall (i,j) \in E, \quad \forall k \in K$
Où K représente toutes les commodités du problème.
le flux d'entrée de l'objet k dans un nœud doit être comparable au flux de sortie de ce même objet en respect du type de nœud dont il s'agit : **source**, **destination** ou **intermédiaire**.

Les mêmes observations qu'au point 3.2 s'appliquent à nouveau ici et sont repris dans l'accolade de l'équation 5.

Faisabilité du flux

Enfin, pour que le flux contenu dans le modèle soit réalisable, comme dans l'approche agrégée, il doit être contraint par la capacité de l'arc concerné et donc être strictement positif et borné supérieurement par les arcs lui étant incident pour l'item k spécifique.

$$f_{ij}^k \leq b_i^k, \tag{6}$$
$$\forall (i,j) \in E, \quad i \in V, \quad k \in K \tag{7}$$

4 Analyse des résultats

Le programme linéaire est résolu via l'utilisation de l'outil *GNU Linear Programming Kit*[1] ou GLPK. Ce software fournit une suite d'outil pour résoudre des problèmes d'optimisation linéaire.

Afin d'utiliser les outils proposés par GPLK, nous devons transformer les instances de problèmes. Pour cela, nous exécutons un script python qui permet de générer un fichier `.lp` au format CPLEX LP représentant le programme linéaire de l'instance du problème. Le format CPLEX LP peut ensuite être utilisé pour être résolu via la commande `glpksol`. Une description plus extensive du format est décrite dans la documentation de GPLK [1, Appendix C]

Une fois la solution obtenue, nous pouvons relever différentes volumétries.

4.1 Volumétrie du problème

Voici les différentes mesures qui ont pu être relevés durant l'analyse de la résolution par **GLPK** :
- Mémoires utilisé en bytes
- Nombre de variables du problème
- La taille triangulaire, donnant une idée de la densité de la matrice (creuse ou dense)

Remarque. La vitesse de résolution pouvant être un bon indicateur de performance, ne se trouve pas être utile dans notre cas étant donné quel est constamment de 0 pour toutes les instances étudiées [1].

Résultat

Commençons par comparer la méthode agrégée et désagrégée sur l'instance de problème '20_2_non-validly'. Regardons de plus près les caractéristiques du problème dont nom de l'instance est composé de trois éléments.

Premier élément, le chiffre 20 représente **le nombre de nœuds** dans l'instance du problème.
Second élément, le chiffre 2 représente **le nombre d'articles devant être transféré** des sources vers les destinations. Troisième et dernier élément, **le nom du fichier**.

Cette instance de problème contient donc 20 nœuds et concernent 2 articles pour le cas désagrégé et un seul article pour l'approche agrégée.

1. Pour référence, le système utilisé pour exécuter le solveur est un MacBook Air utilisant la puce M2.

Et voici le résultat pour les deux approches sur l'instance `nonvalidly` :

```
AGGREGATED
    Problem:
        Rows:       20
        Columns:    61 (61 integer, 0 binary)
        Non-zeros:  121
        Status:     INTEGER OPTIMAL
        Objective:  obj = 1694233 (MINimum)


DEAGGREGATED
    Problem:
        Rows:       40
        Columns:    122 (122 integer, 0 binary)
        Non-zeros:  242
        Status:     INTEGER OPTIMAL
        Objective:  obj = 1683245 (MINimum)
```

Les différentes parties du résultat signifie :
- **Rows** : représentent le nombre de lignes dans la matrice des contraintes du problème
- **Columns** : représente le nombre les variables du problème
- **Non-Zeros** : Le nombre de variables et contraintes ayant des coefficients non nuls
- **Status** : statut du résultat trouvé
- **Objective** : valeur résultat de la fonction objective fournit par le solveur

On observe dans le résultat agrégé que le solveur a considéré **20 rows, 61 columns dont 121 non-zeros**.
Dans le résultat désagrégé, on observe que le nombre de rows, columns et non-zeros **a doublé**. Effectivement, le nombre de variables et de contraintes dans la matrice est directement lié au nombre de nœuds, mais également au nombre d'articles impliqués.

Regardons maintenant l'instance du problème `26_4_barbettes`. En suivant la logique expliquée précédemment, on déduit que le problème implique 26 nœuds et 4 articles lors de l'approche désagrégée.

Voici les résultats obtenus de l'instance `barbettes` :

```
AGGREGATED
Problem:
    Rows:       26
    Columns:    76 (76 integer, 0 binary)
    Non-zeros:  150
    Status:     INTEGER OPTIMAL
    Objective:  obj = 4743833.5 (MINimum)

DESAGGREGATED
    Problem:
        Rows:       104
        Columns:    304 (304 integer, 0 binary)
        Non-zeros:  600
        Status:     INTEGER OPTIMAL
        Objective:  obj = 4811434 (MINimum)
```

Une nouvelle fois, on observe que le nombre de rows, columns et non-zeros **a quadruplé**, lorsque l'on passe de l'approche agrégée à l'approche désagrégée.

Cela se remarque également sur la mémoire utilisée par les instances du problème étant donné que pour l'instance `barbettes`, la mémoire utilisé passe de 117120 bytes en agrégée à 363974 bytes pour le désagrégé, soit **plus du triple de mémoire utilisé**.

Mémoire utilisé

On observe une tendance semblable à moins grande échelle pour l'instance `20_2_nonvalidly` passant de 103595 bytes à 164600 bytes. Une augmentation bien moindre étant donné le nombre d'articles étant plus restreint également.

Remarquons cependant que, lors de l'approche agrégée, la mémoire utilisée par les deux instances ci-dessus sont bien plus proches que lors de l'utilisation de la méthode désagrégée qui donne un résultat bien différent.

Taille triangulaire

Enfin, une dernière composante intéressante à observer est la taille triangulaire calculée par le solveur qui permet de donne un indice sur la disparité ou densité de la matrice étudié. Plus celle-ci est élevée, plus la complexité de résolution est élevée.
Dans le cas de l'instance `nonvalidly`, la taille triangulaire est de 20 pour l'approche agrégée et de 40 pour l'autre. La considération d'un seul article est donc, logiquement, plus simple que la considération de deux articles.
Nous observons la même tendance dans l'instance `barbettes`, passant d'une taille triangulaire de 24 à 96.

Observons qu'à nouveau, la méthode désagrégée donne des résultats beaucoup plus disparate que la méthode agrégée qui pour un nombre accru de nœuds restent proches de la constance dans sa complexité.

4.2 Aucun résultat trouvé

Penchons-nous maintenant sur l'instance du problème `20_3_didactics`. Ce problème est composé de 20 nœuds et 3 articles, dont on a le résultat suivant :

```
AGGREGATED
    Problem:
        Rows:       20
        Columns:    60 (60 integer, 0 binary)
        Non-zeros:  120
        Status:     INTEGER UNDEFINED
        Objective:  obj = 0 (MINimum)

DESAGGREGATED
    Problem:
        Rows:       60
        Columns:    180 (180 integer, 0 binary)
        Non-zeros:  360
        Status:     INTEGER UNDEFINED
        Objective:  obj = 0 (MINimum)
```

On observe ici une valeur 'Objective' de 0 et un 'Status' **INTEGER UNDEFINED**.
Le solveur n'a pas pu trouver de solution pour aucune des deux approches.

Regardons plus en détails les informations fournies par cette instance et plus particulièrement à la définition des nœuds sources et destinations :

```
SOURCES 5
ID  CAPACITY_ITEM_0 CAPACITY_ITEM_1 CAPACITY_ITEM_2
4   919             307              817
10  404             158              55
0   0               0                854
3   921             0                0
1   0               349              0

DESTINATIONS 5
ID  DEMAND_ITEM_0   DEMAND_ITEM_1    DEMAND_ITEM_2
9   489             0                512
14  538             328              356
8   0               0                368
16  0               281              0
12  1216            204              488
```

Dans l'extrait du fichier représentant l'instance, on voit qu'y est décrit les sources et les destinations.

Lorsque l'on fait la somme agrégée des productions et consommations des articles, on obtient :

- $919 + 404 + 921 + 307 + 158 + 349 + 817 + 55 + 854 = 4784$ total des articles agrégés produits par les sources
- $489 + 538 + 1216 + 328 + 281 + 204 + 512 + 356 + 368 + 488 = 4780$ total des articles consommé par le problème dans les nœuds destinations

On peut observer que si l'on fait l'addition de la production totale au niveau des sources, celle-ci est toujours inférieure à la somme des demandes. Ce qui rend la résolution du problème **impossible à réaliser** étant donné que les sources ne pourront jamais satisfaire toutes les demandent.

Il est donc normal que le solveur ne puisse fournir aucune solution satisfaisant l'une ou l'autre approche.

4.3 Optimal agrégé uniquement

Penchons-nous maintenant sur l'instance `25_4_ducdame`, dont voici les résultats :

```
AGGREGATED
    Problem:
        Rows:       25
        Columns:    76 (76 integer, 0 binary)
        Non-zeros:  152
        Status:     INTEGER OPTIMAL
        Objective:  obj = 4052795.5 (MINimum)

DISAGGREGATED
    Problem:
        Rows:       100
        Columns:    304 (304 integer, 0 binary)
        Non-zeros:  608
        Status:     INTEGER EMPTY
        Objective:  obj = 0 (MINimum)
```

Cette fois, on observe un statut **INTEGER OPTIMAL** pour l'approche agrégée signifiant une solution au problème trouvé et un I**NTEGER EMPTY** pour le désagrégé représentant une solution impossible.

À nouveau, regardons en détail les spécifications du problème :

```
SOURCES 6
ID  CAPACITY_ITEM_0 CAPACITY_ITEM_1 CAPACITY_ITEM_2 CAPACITY_ITEM_3
7   0               116             534             80
15  87              160             615             0
0   197             792             426             526
8   330             199             840             591
10  443             330             935             732
1   92              399             0               369

DESTINATIONS 6
ID  DEMAND_ITEM_0   DEMAND_ITEM_1   DEMAND_ITEM_2   DEMAND_ITEM_3
24  458             549             1056            352
17  312             0               1254            749
5   351             601             0               96
3   0               0               0               653
12  26              484             1038            382
14  0               385             0               0
```

Cette fois-ci, si nous faisons la somme agrégée des capacités des nœuds sources, le résultat est bien plus grand que la somme des demandes des nœuds destinations.

Cependant, lorsqu'on fait la somme des articles non-agrégés, on se rend compte que la demande de l'article 1 est strictement plus grande que la capacité de production de ce même article :

- $116 + 160 + 792 + 199 + 330 + 399 = 1996$ capacité de production
- $549 + 601 + 484 + 385 = 2019$ consommation

La version agrégée, composant le problème en sommant toutes les productions d'articles, permet de répondre à la demande.
Cependant, la version désagrégée, ne pouvant pas puiser dans la production des autres commodités, ne parvient pas à satisfaire la demande de cet article 1.

5 Conclusion

Lors de ce travail, nous avons pu observer les différents résultats et possibilités découlant de l'approche du problème de coût minimum de transport de plusieurs articles, par l'approche agrégée et l'approche désagrégée.

Nous avons remarqué la complexité et l'influence du nombre d'articles sur une même instance du problème. Il est à noter que dans certains cas, ni l'approche agrégée ni l'approche désagrégée ne peuvent fournir de solution. Dans d'autres cas, seule l'approche agrégée permet une solution.

Une approche agrégée offre également une résolution du problème plus stable, avec moins de disparités dans la taille triangulaire, l'espace mémoire utilisé et la taille de la matrice de contraintes. En revanche, l'approche désagrégée, bien qu'elle réponde précisément au problème initial, connaît une croissance beaucoup plus rapide en complexité dès que l'on ajoute des nœuds et des articles.

Des expériences avec des instances du problème contenant davantage de nœuds et d'articles pourraient permettre une étude plus approfondie sur le temps d'exécution du solveur GLPK, mais l'on peut s'attendre à une croissance similaire du temps d'exécution à celle observée pour d'autres volumétries.

Références

[1] Andrew O. Makhorin. *GLPK (GNU Linear Programming Kit)*, 2024. Version 5.0, retrieved on May 17, 2024.