École des Ponts Paristech

Recherche Opérationnelle

Renault et ses emballages en Europe

Arnaud Rubin, Adrien Sadé, Aubin Tchoï

14 Janvier 2021

1 Questions

Question 1.

Supposons que $\ell_1 > L/2$. Le seul type d'emballage disponible est donc tel qu'il n'est possible de mettre qu'une seule pile dans les camions de transport. Pour chaque pile transportée, un camion est nécessairement mobilisé, et il doit s'arrêter chez le seul fournisseur livré. On a donc pour coût fixe sur chaque livraison c^{cam} et c^{stop}. Le problème ainsi simplifié peut être modélisé par un problème de flot.

À l'échelle d'une journée, on considère un graphe biparti avec, d'un côté, les usines, et de l'autre, les fournisseurs. Les arêtes reliant les usines aux fournisseurs sont de coût $\gamma.d_a$ (où γ est le coût kilométrique des trajets et d_a est la distance kilométrique de l'arête $a \in A$) auquel on ajoute c^{cam} et c^{stop} pour lancer et arrêter le camion de transport. À cette échelle quotidienne, des offres d'emballages sont associées aux usines et des demandes d'emballages sont associées aux fournisseurs.

Prenons désormais en compte l'horizon temporel J. Pour chaque $j \in J$, un graphe biparti de la forme mentionnée précédemment est nécessaire. On relie tous ces graphes bipartis grâce à des arcs bien particuliers : chaque usine est reliée à elle-même dans le graphe biparti du lendemain et on procède de la même façon pour chaque fournisseur. Pour les arcs reliants une usine à la même usine le lendemain, on impose un coût c^s_{eu} et pour les arcs reliants un fournisseur au même fournisseur le lendemain, on impose un coût c^s_{ef}. Cela permet de modéliser des gestions de stocks sur l'horizon temporel J.

Question 2.

On suppose que la « taille de l'instance » ne concerne que le nombre d'usines et de fournisseurs. On montre alors que la formulation de l'énoncé du PLNE est bien polynomiale en la taille de l'instance. Rappelons les contraintes de cette formulation du problème :

$$s_{euj} = s_{eu(j-1)} + b^+_{euj} - z^-_{euj}, \quad \forall e, u, j \tag{1}$$

$$0 \geq s_{euj}, \quad \forall e, u, j \tag{2}$$

$$s_{efj} = \max(s_{ef(j-1)} - b^-_{efj}, 0) + z^+_{efj}, \quad \forall e, f, j \tag{6}$$

$$z^-_{euj} = \sum_{R \in \delta^+(u,j)} x_R q_{eR}, \quad \forall e, u, j \tag{10a}$$

$$z^+_{efj} = \sum_{R \in \delta^-(f,j)} x_R Q_{efR}, \quad \forall e, f, j \tag{10b}$$

On constate que les contraintes (1), (2) et (10a) apparaissent un nombre de fois proportionnel au nombre d'usines ($\forall u$), de même les contraintes (6) et (10b) apparaissent un nombre de fois proportionnel au nombre de fournisseurs ($\forall f$). Le nombre de contraintes (et donc le nombre de variables du problème) est polynomial en la taille de l'instance.
Néanmoins, remarquons que « la taille de l'instance » peut aussi désigner en plus la taille des emballages et le nombre de jours étudiés. Auquel cas, constatons que le nombre de variables induit par les contraintes est alors en $\mathcal{O}(euj)$ et $\mathcal{O}(efj)$. Ce nombre n'est plus polynomial en la taille de l'instance.

Dans tous les cas, en notant que pour `europe.csv`, $U = 25$, $F = 576$, il est tout à fait impossible de résoudre ce problème optimalement à l'aide d'un solveur.

Question 3.

On introduit des variables qui dépendent des arêtes du graphes :

— x_{ejkab} : nombre d'emballages de type e transportés le jour j par le camion k entre les sites a et b avec $a, b \in U \cup F$
— $\delta_{ejkab} = 1$ si $x_{ejkab} > 0$
— $\Delta_{jk} = 1$ si le camion k a été utilisé le jour j
— $s_{eaj}, s'_{eaj}, s"_{eaj}$ les stocks en emballages de type e du site a le jour j

Avec ces variables : $\sum_{k,a}(x_{ejkaf} - x_{ejkfa})$ représente ce qui est livré au site f.
Les stocks s'_{eaj}, $s"_{eaj}$ vont nous être utiles pour linéariser les max dans la fonction objectif, pour linéariser $max(s, 0)$ on peut replacer s par une variable s' que l'on va contraindre à être supérieure à s et à 0, pour peu que dans la fonction objectif on minimise ce max s' va correspondre à $max(s, 0)$. Pour le terme dans l'objectif correspondant aux stocks des fournisseurs (elle contenait à la fois une partie positive et une partie négative) à la somme avec la partie positive de $(b^-_{efj} - s_{ef(j-1)})$ j'ai réécrit : $\max(b - s, 0)$ comme $\big(\max(s - b, 0) - (s - b)\big)$.
On peut ensuite simplifier les b dans la fonction objectif.

On peut exprimer la fonction objectif avec ces nouvelles variables :

$$\begin{aligned}\min \sum_{j,k} \Big(c^{cam}\Delta_{ej} + \sum_{e,a,b} \delta_{ejkab}(d_{ab}\gamma + c^{stop}) \Big) \\ + \sum_{j,e} \Big(\sum_{u \in U} c^s_{eu}(s'_{euj} - r_{euj}) \\ + \sum_{f \in F} \Big(c^s_{ef}(s'_{efj} - r_{efj}) + c^{exc}_{ef}(s"_{efj} - b^-_{ef(j+1)}) - c^{exc}_{ef}(s_{efj} - b^-_{ef(j+1)}) \Big) \Big)\end{aligned}$$

Il suffit ensuite de traduire les contraintes portant sur l'évolution des stocks (pour les usines et pour les fournisseurs), sur le chargement des camions (capacité inférieure à L et pas plus de 4 fournisseurs par camion).
Il reste à contraindre δ à bien être égal à 1 si x est positif et 0 sinon. On utilise pour cela les inégalités de McCormick (de même pour Δ) :

— $\forall e, j, k, a, b \quad \delta_{ejkab} \in 0, 1$
— $\forall e, j, k, a, b \quad x_{ejkab} \geq 0$
— $\forall e, j, k, a, b \quad x_{ejkab} \leq L\delta_{ejkab}$
— $\forall e, j, k, a, b \quad x_{ejkab} \geq \delta_{ejkab}$

2 Algorithme

2.1 Approche n°1

Dans le but de réduire la taille du problème, on commence par par créer des clusters géographiques, chacun comportant 4 fournisseurs et toutes les usines. Par ailleurs, pour commencer avec simplicité, on ne mélange pas les emballages dans les camions.

En travaillant de cette façon, 2 problèmes surviennent :

— En travaillant sur les clusters, il est possible d'épuiser les stocks d'une usine à destination d'un cluster puis de les épuiser à nouveau en passant au cluster suivant.
— De la même façon, il est possible de remplir plusieurs fois le même camion avec des piles d'emballages différentes. Les contraintes associées ne sont pas prises en compte, ni actualisées.

Une solution pour résoudre ces problèmes consiste à créer des listes pour adapter les contraintes à chaque boucle sur les clusters et emballages : on met à jour une variable qui contient la contenance

du camion k et la contrainte serait alors de ne pas dépasser la contenante restante. Il s'agit alors d'un algorithme glouton qui dépend alors de l'ordre dans lequel sont traités les clusters et les emballages.

Il reste néanmoins un problème sur les performances du code car la résolution d'un PLNE sur l'instance `europe.csv` prend $t_{PLNE} = 40s$. La résolution complète prend donc $E \times N_{clusters} \times t_{PLNE}$, soit environ 4,5h.

En traitant le problème sur de plus petites instances on observe que le coût obtenu est majoritairement dû aux stocks excédentaires : il faut donc plutôt vider les stocks des usines et des fournisseurs plutôt que de chercher à minimiser la distance parcourue (le facteur écologique est alors de second plan). Par ailleurs, le fait de recombiner les emballages dans les camions est aussi à l'origine de coûts importants, qui s'élève ici à plus de 12,8 millions.

2.2 Approche n°2

On change la façon de procéder. L'idée est alors de respecter de façon exacte les demandes des fournisseurs avec un certain nombre de camions transportant des emballages. Une fois une solution obtenue, on cherche à améliorer localement la solution obtenue pour l'instance `europe.csv`.

Étape 1 : On commence par chercher à résoudre un problème de flots : dans chaque usine on libère des emballages et chez chaque fournisseur on consomme des emballages ; on peut donc chercher à savoir combien de piles de type e doivent être envoyées de l'usine u vers le fournisseur f le jour j. Puisqu'il s'agit d'un flot, ce problème est résolu de façon exacte et assez rapidement avec un PLNE. Les variables de décisions comportent alors les quantités transportées entre une usine et un fournisseur pour chaque jour et chaque type d'emballage ainsi que les stocks. On peut ensuite calculer le coût objectif associé de la même manière que dans la question 3 avec des contraintes qui régissent l'évolution des stocks. En pratique on résout $|E|$ PLNE indépendants, un pour chaque type d'emballage, afin de ne pas surcharger la mémoire vive de nos machines.

Étape 2 : Il s'agit ensuite de trouver les routes qui vont répondre à cette demande. On regarde pour chaque couple (u, f) comment charger optimalement les camions (car la solution du problème de flot indique combien d'emballages doivent être transportés sur l'arc reliant u à f chaque jour), ce problème peut se formuler comme un PLNE où les variables de décisions seraient le nombre de piles de chaque type envoyées par chaque camion ainsi que les camions utilisés (sous forme de variable binaire), les contraintes porteraient sur le chargement, chaque camion possède une longueur maximale et on chercherait alors à minimiser le nombre de camions puisqu'à (j, u, f) fixé le coût de transport est obtenu en multipliant le nombre de camions par une constante (qui dépend du triplet (j, u, f) choisi. On résout alors tous ces PLNE avant d'envoyer les camions tout droit d'une usine vers un fournisseur. Le coût lié aux routes est alors élevé mais en tout on obtient un bien meilleur résultat qu'avec l'ancien algorithme pour un temps de de calcul bien inférieur.

Étape 3 : La solution obtenue est alors admissible, on peut chercher à l'améliorer localement en se déplaçant dans un voisinage bien défini. On trouve deux algorithmes à cet effet ; le premier consiste à diminuer le nombre de routes ; à l'étape précédente on a résolu un PLNE qui assigne un certain chargement à un certain nombre de camions pour chaque triplet (j, u, f). Dans les solutions de ce PLNE certains camions sont visiblement très vides, ce qui est prévisible car on cherche absolument à envoyer un certain nombre d'emballages donné par la résolution du premier problème de flots (on l'impose en contrainte). Il est cependant possible qu'un camion (donc une route) ne soit en fait pas rentable, et que malgré les problèmes de stocks il soit en fait préférable de ne rien envoyer plutôt que

d'envoyer un camion presque vide à cause du coût de location du camion, du coût par arrêt et du prix kilométrique. On peut ainsi parcourir l'ensemble des routes pour regarder si le fait de retirer une route améliore notre solution.
On peut tomber dans un minimum local, cet algorithme est en effet dépendant de l'ordre dans lequel on étudie les routes, il n'est pas impossible que garder la route i et retirer la route $i+1$ soit préférable à l'inverse, mais on peut ne jamais s'en rendre compte si l'on parcourt les routes sur i croissant et que retirer la route i améliore déjà la solution de peu.
Cet algorithme améliore assez rarement la solution, il faut vraiment trouver une route qui contient un chargement très petit, mais dans ce cas le gain en coût est important, de l'ordre du millier.

Une autre possibilité d'amélioration locale consiste à regrouper les camions ; on peut étudier les routes obtenues et étudier pour chaque couple de routes s'il ne serait en fait pas préférable de les combiner en faisant partir un seul camion qui effectuerait un trajet dans lequel il s'arrêterait au premier fournisseur puis au suivant (cette observation s'applique uniquement à un couple de routes de fournisseurs différents, la résolution des PLNE nous assure que cette modification ne peut pas combiner des camions qui font le même trajet le même jour).

Algorithm 1 Amélioration Locale

Input : une solution S du problème sur l'instance `europe.csv`
    for route r **in** S :
        if $coût(S \text{ privé de } r) \leqslant coût(S)$:
            $S' \leftarrow S$ privé de r
        end
    end
    return S'
Output : une solution S' meilleure que S

2.3 Bilan des scores obtenus

	`maroc.csv`	`espagne.csv`	`france.csv`	`europe.csv`
Meilleur score	407 058	1 691 557	2 808 519	7 955 963
Temps de calcul	1 min	3 min 30	15 min	35 min