Projet SOD 321 - Optimisation Discrète

Nicolas Barlier, Joshua Wolff

Octobre 2021

Table des matières

1 Sujet

L'objectif de ce projet est d'implémenter des méthodes de résolution d'un problème de type *plus court chemin*. Plus précisément, nous cherchons le plus court chemin à parcourir en avion reliant deux aéroports donnés. Plusieurs contraintes sont à respecter : il y a un nombre minimum d'aéroports à visiter, chaque aéroport doit être visité au plus une fois, il faut visiter toutes les régions possibles et notre avion ne peut pas effectuer des vols supérieurs à une certaine distance. Le problème ainsi posé diffère du *shortest-path problem* classique, notamment de par le fait qu'il n'est pas nécessaire de visiter tous les aéroports.

Nous nous intéresserons tout d'abord à une formulation du problème avec un nombre polynomial de contraintes puis avec un nombre exponentiel de contraintes. Dans ce second cas, nous proposerons trois méthodes de résolution du problème. Enfin, nous comparerons le temps d'exécutions de ces différentes méthodes sur des instances de taille variable.

2 Modélisation du problème

Pour $i, j \in \{1, ..., n\}$, on note :

- A_{min} : le nombre minimal d'aérodromes à visiter,
- R : la distance maximale que peut parcourir un avion sans se poser
- d : numéro de l'aérodrome de départ, f : numéro de l'aérodrome d'arrivé
- d_{ij} la distance entre les sommets (aéroports) i et j
- $x_{ij} = \begin{cases} 1 & \text{si } (i \to j) \text{ est un vol effectué} \\ 0 & \text{sinon} \end{cases}$, variable de sélection d'arcs sur le graphe
- Pour $a \in \{1, ..., m\}$ on note $\delta(a) = \{k \in \{1, ..., n\},$ *l'aérodrome k appartient à la région a*$\}$

Le problème P se formule donc :

$$
\begin{cases}
\min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \\
\text{Sous contraintes :} \\
\forall i \in \{1,..,n\},\ x_{ii} = 0 \\
\sum_{i=1}^{n} x_{id} = 0,\ \sum_{i=1}^{n} x_{di} = 1,\ \sum_{i=1}^{n} x_{if} = 1,\ \sum_{i=1}^{n} x_{fi} = 0 \text{ (Conditions au bords)} \\
\sum_{i=1}^{n} \sum_{i=1}^{n} x_{ij} \geq A_{\min} - 1 \text{ (Nombre minimal d'aérodrome à visiter)} \\
\forall i,j \in \{1,...,n\},\ x_{ij} d_{ij} \leq R \text{ (Distance maximale entre deux aérodromes)} \\
\forall i \in \{1,...,n\},\ \sum_j (x_{ij} - x_{ji}) = \begin{cases} 1 & \text{si } i = d \\ -1 & \text{si } i = f \\ 0 & \text{sinon} \end{cases} \\
\forall a \in \{1,...,m\},\ \sum_{i \in \delta(a)} \sum_{j=1}^{n} (x_{ij} + x_{ji}) \geq 1 \text{ (Passage par toutes les régions)} \\
\forall i,j \in \{1,..,n\},\ x_{ij} \in \{0,1\}
\end{cases}
$$

Pour compléter ce problème, il faut ajouter une contrainte de connexité. Il existe différentes formulations de cette contrainte : certaines entrainant un nombre polynomial de contraintes d'autres un nombre exponentiel de contraintes.

3 Nombre polynomial de contrainte

On ajoute au problème P formulé précédemment :

- une variable u_i qui renseigne sur l'ordre de passage dans chaque aérodrome
- une contrainte de connexité : $u_j \geq u_i + 1 + n(x_{ij} - 1)$ avec $i,j \in \{1,...,n\} \backslash d$

Remarque : cette contrainte assure bien la connexité car si l'arrête $i \to j$ fait partie du parcours alors on a $u_j > u_i$

4 Nombre exponentiel de contraintes

On peut formuler la contrainte de connexité avec un modèle d'élimination de sous tours. Le problème P (décrit en 2) se complète donc avec la contrainte suivante (inégalité de sous-tours) :

$$
\boxed{\forall S \subset \{1,...,n\} : |S| \geq 1, \qquad \sum_{i,j \in S} x_{ij} \leq |S| - 1} \tag{1}
$$

On ne peut pas écrire toutes ses contraintes pour une résolution informatique (il y en a 2^n). On procède donc à différentes méthodes de résolution qui s'appuient sur des sous problèmes.

4.1 Recherche de cycles dans le graphe

On présente dans cette partie une façon de résoudre le problème avec un nombre exponentiel de contraintes par génération de contraintes. Nous allons résoudre un sous-problème qui, à chaque nouvelle résolution va générer une nouvelle contrainte dans le problème principal. Plus précisément, le sous-problème que nous résolvons est un problème de détection de cycles dans un graphe. Le fonctionnement de l'algorithme de résolution du sous-problème est décrit ci-dessous :

Algorithme 1 : Obtention de l'indice suivant (Next(M,i) $\to j$)

Entrées : M matrice de déplacement sur le graphe, i indice du sommet actuel
Sorties : j indice du sommet suivant
si $\sum_j M_{ij} = 1$ **alors**
  $j \leftarrow$ indice j t.q. $M_{ij} = 1$
sinon
  $j \leftarrow -1$
fin
return j

Algorithme 2 : Détection de cycles

Entrées : M matrice de déplacement sur le graphe, d et f sommets de départ et d'arrivée
Sorties : L liste des cycles dans le graphe
$N \leftarrow$ nombre de lignes de M
visited $\leftarrow$ liste de False de taille N
visited[d] $\leftarrow$ True
$L \leftarrow$ []
$l \leftarrow [s]$
$j \leftarrow$ Next(M,s)
tant que $j \neq -1$ *et* $j \notin l$ **faire**
  Ajoute j à l
  visited[j] $\leftarrow$ True
  $j \leftarrow$ Next(M,j)
fin
pour chaque $i \in \{1, ..., N\}$ **faire**
  si visited[i]=False *et* $\sum_j M_{ij} = 1$ **alors**
    $l \leftarrow [i]$
    $j \leftarrow$ Next(M,i)
    tant que $j \neq -1$ *et* $j \notin l$ **faire**
      Ajoute j à l
      visited[j] $\leftarrow$ True
      $j \leftarrow$ Next(M,j)
    fin
    On ajoute l à L
  fin
fin
return L

À chaque itération, on résout le sous-problème de détection de cycles. Chaque cycle fournit un sous-ensemble de sommets S, qui est utilisé pour formuler une contrainte d'élimination de sous-tours qui est ajoutée au problème principal P :

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1$$

4.2 Inégalités de sous-tours

On procède à une séparation des inégalités. On cherche un ensemble S qui ne respecte pas l'inégalité de sous tours. En notant a_i la variable binaire = 1 ssi $i \in S$, cela revient à résoudre le sous problème suivant :

$$\begin{cases} \max \Delta = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} a_i a_j - \sum_{i=1}^{n} a_i + 1 \\ \text{s.c.} \\ \sum_{i=1}^{n} a_i \geq 1 \\ a_i \in \{0, 1\} \end{cases}$$

Ce sous problème en variable binaire peut être implémenté dans le solveur.

4.3 Inégalités de sous-tours (généralisation)

Une autre méthode consiste a remplacer l'inégalité de sous tours (1) par une version améliorée (générale) :

$$\sum_{i,j \in S} x_{ij} \leq \sum_{i \in S} y_i - y_{i_0}, \quad \forall S \subset \{1, ..., n\}, \forall i_0 \in \{1, ..., n\} : i_0 \in S \tag{2}$$

où y est une variable binaire de selection de sommets ($y_i = 1$ ssi le parcours parcours passe par i)
Leur signification est la suivante : étant donné un ensemble S et un sommet i_0 de S. Si le parcours passe par i_0 ($y_{i_0} = 1$) alors le nombre d'arcs à l'intérieur de S ne dépasse pas le nombre de sommets de S parcourus moins 1.

Sinon ($y_{i_0} = 0$) alors l'inégalité est vérifiée aussi mais elle est redondante avec certaines contraintes déja précisées dans le problème P.
Il s'agit maintenant de séparer ces inégalités de sous tours généralisées. On cherche donc un ensemble S et un sommet i_0 de S qui ne respècteent pas l'inégalité (2). En notant h_i la variable binaire telle que $h_i = 1$ ssi i est l'unique sommet qui jour le rôle de i_0, cela revient à chercher (a, h) solution du sous problème suivant :

$$\begin{cases} \max \Delta_g = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} a_i a_j - \sum_{i=1}^{n} y_i a_i + \sum_{i=1}^{n} y_i h_i \\ \text{s.c.} \\ h_i \leq a_i \quad \forall i \in \{1, .., n\} \\ \sum_{i=1}^{n} h_i = 1 \\ a_i, h_i \in \{0, 1\} \end{cases}$$

Relaxation continue : Pour des questions d'efficacité algorithmique, on utilise la relaxation continue du problème principal. Tant que le critère du sous-problème associé est strictement positif, on continue la relaxation continue. Une fois le critère du sous-problème devenu négatif la résolution se poursuit en variable binaire jusqu'à atteindre l'optimalité. Cette méthode permet d'avoir très rapidement une bonne borne inférieur lors de nos résolutions du problème.

5 Comparaison des méthodes

La Table 1 récapitule les temps d'exécution des différentes méthodes sur des instances plus ou moins grands. Les mentions *n.e.* (pour non-évalué) indique que les temps d'exécutions ne semblaient pas raisonnable et qu'ils n'ont donc pas été évalués pour éviter des calculs trop lourds.

Formulation	6 aéroports	20 aéroports	40 aéroports	70 aéroports
Polynomiale	300 ms	1 min 17 s	10 s	*n.e.*
Exponentielle (cycles)	250 ms	*n.e.*	25 s	*n.e.*
Exponentielle (classique)	370 ms	*n.e.*	36 s	*n.e.*
Exponentielle (généralisé)	370 ms	27 s	7 s	*n.e.*

TABLE 1 – Temps d'exécutions des méthodes sur différentes instances

Les valeurs optimales obtenues sont les suivantes : 12 pour l'instance à 6 aéroports, 122 pour l'instance à 20 aéroports et 112 pour l'instance à 40.

6 Conclusion

Au cours de ce projet, nous avons formulé un problème de "course d'avion", qui s'approche du problème de plus court chemin tout en imposant de nouvelles contraintes bien spécifiques. Nous proposons quatre formulations distinctes pour ce problème (une avec un nombre polynomial de contraintes, et trois avec un nombre exponentiel de contraintes). Nous avons notamment imaginé une méthode qui assure la connexité de la solution en s'appuyant sur la detection de cycles dans des graphes.

On observe finalement que la méthode la plus efficace correspond à la formulation avec les inégalités de sous-tours généralisée, qui assure de très bons résultats sur quatre des trois instances proposées. Nous n'avons pas réussi à trouver de solution pour l'instance à 70 aérodromes en un temps de calcul raisonnable (inférieur à une heure).