Projet SOD321 : Course d'avions

Aicha BOUJANDAR - Khaoula BELAHSEN

October 21, 2019

1 Introduction

Dans ce projet, on modélise par un programme linéaire la course d'avions Breitling (100/24). On connait le nombre n d'aérodromes et le nombre m de régions. Pour chaque aérodrome, on a ses coordonnées (x,y) ainsi que la région à laquelle il appartient. On fixe les aérodromes de départ et l'arrivée.

L'objectif de ce problème est de minimiser la distance parcourue par un avion tout en respectant les contraintes :

- Passer par les m régions au moins une fois.
- Passer par au moins A_{min} aérodromes.

2 Modélisation du problème

2.1 Formulation polynomiale

1. **Paramètres** :
   - n, m : nombre d'aérodromes et de régions
   - d,a : aérodromes de départ et d'arrivée
   - amin : nombre d'aérodromes à parcourir
   - R : distance maximale pouvant être parcourue dans les airs
   - g : matrice nxm avec $g_{i,j} = 1$ si l'aérodrome i appartient à la région j, 0 sinon.
   - $dist_{i,j}$ distance entre les aérodromes i et j
   - $A = \{(i, j) \in [1, n], dist_{i,j} \leq R\}$
   - $A_d = \{i \in [1, n] : i \neq d\}$
   - $A_a = \{i \in [1, n] : i \neq a\}$

2. **Variables** :
   - $\delta_{i,j} = \begin{cases} 1 & si\ chemin\ de\ i\ à\ j \\ 0 & sinon \end{cases}$, binaire
   - u_i : nombre d'aérodromes visités en arrivant à i depuis l'aérodrome de départ.
   - $x_i = \begin{cases} 1 & si\ on\ passe\ par\ i \\ 0 & sinon \end{cases}$, binaire

3. **Fonction objectif** : distance = $\sum_{(i,j)\in A} dist_{i,j}\delta_{i,j}$

4. **Contraintes** :
   - Chaque région doit être visitée au moins une fois :
   $$\forall k \in [\![1, m]\!] : \sum_{i\in[\![1,n]\!]} g_{i,k} * x_i$$
   - L'avion doit atterrir dans au moins amin aérodromes :
   $$\sum_{i\in[\![1,n]\!]} x_i \geq amin$$
   - L'avion doit partir d'un aérodrome au plus une fois. Il ne peut partir que d'un aérodrome visité :
   $$\forall i \in A_a : \sum_{j,(i,j)\in A} \delta_{i,j} = x_i$$
   - L'avion doit atterrir dans un aérodrome au plus une fois. Il ne peut atterrir que dans un aérodrome visité :
   $$\forall j \in A_d : \sum_{i,(j,i)\in A} \delta_{j,i} = x_i$$
   - On ne peut ni aller au départ, ni partir de l'arrivée :
   $$\forall i \in [1, n]\ tq\ (i, d) \in A : \delta_{i,d} = 0$$
   $$\forall j \in [1, n]\ tq\ (a, j) \in A : \delta_{a,j} = 0$$
   - Contraintes d'élimination des sous-tours :
   $$\forall (i, j) \in A : u_j \geq u_i + x_j - n(1 - \delta i, j)$$
   $$u_a = \sum_{j=1}^{n} x_j$$
   $$\forall i \in [1, n] : u_i \leq nx_i$$

2.2 Formulation exponentielle

1. **Variables** :

   - $\delta_{i,j} = \begin{cases} 1 & si\ chemin\ de\ i\ à\ j \\ 0 & sinon \end{cases}$, binaire
   - $x_i = \begin{cases} 1 & si\ on\ passe\ par\ i \\ 0 & sinon \end{cases}$, binaire

2. **Fonction objectif** : minimiser distance $= \sum_{(i,j)\in A} dist_{ij}\delta_{ij}$

3. **Contraintes** :

   - Chaque région doit être visitée au moins une fois :

     $$\forall k \in [\![1,m]\!] : \sum_{i\in[\![1,n]\!]} g_{i,k} * x_i$$

   - L'avion doit atterrir dans au moins amin aérodromes :

     $$\sum_{i\in[\![1,n]\!]} x_i \geq amin$$

   - L'avion doit partir d'un aérodrome au plus une fois. Il ne peut partir que d'un aérodrome visité :

     $$\forall i \in A_a : \sum_{j,(i,j)\in A} \delta_{i,j} = x_i$$

     Cette inégalité n'est valable pour l'aérodrome d'arrivée car il doit être visité mais l'avion ne doit pas partir depuis cet aérodrome vers un autre.

   - l'avion doit atterrir dans un aérodrome au plus une fois. Il ne peut atterrir que dans un aérodrome visité :

     $$\forall i \in A_d : \sum_{j,(i,j)\in A} \delta_{j,i} = x_j$$

     Cette inégalité n'est pas valable pour l'aérodrome de départ car il est visité mais on ne doit jamais y retourner.
     Les deux dernières inégalités représentent également la contrainte de conservation dans les aérodromes visités (hors ceux de départ et d'arrivée).

   - On ne doit jamais retourner à l'aérodrome de départ :

     $$\forall i \in [\![1,n]\!], (i,d) \in A : \delta_{id} = 0$$

   - On ne doit jamais partir de l'aérodrome d'arrivée :

     $$\forall j \in [\![1,n]\!], (a,j) \in A : \delta_{aj} = 0$$

- Les aérodromes de départ et d'arrivée doivent être obligatoirement visités et on ne peut pas passer directement de l'aérodrome de départ à celui d'arrivée :

$$x_a = 1$$

$$x_d = 1$$

$$\delta_{da} = 0$$

Elimination des sous-tours

Pour assurer la connexité de notre graphe, on procède à l'élimination des différents sous-tours en utilisant une approche itérative. Notre programme différe néanmois de l'approche vue en cours (celle du sous-problème). En effet, dans notre fichier .run, on crée une boucle itérative dont les étapes sont les suivantes :

- Choix d'un aérodrome i visite i.e tq : $x_i = 1$
- Création d'un ensemble des points S_k tels qu'il existe un chemin entre eux et cet aérodrome i : ce qui nous donne un ensemble candidat à être un sous-tour.
- Test sous-tour :
  - Si l'inégalité des sous-tours est violée, cet ensemble S_k est un sous-tour et on ajoute l'inégalité
    $$\sum_{i \in S_k, j \in S_k} \delta_{ij} \leq |S_k| - 1, \forall k \in [\![1, i]\!]$$
    , puis on résout notre problème une nouvelle fois.
  - Sinon, cet ensemble n'est pas un sous-tours et on continue.
- Si l'ensemble choisi incluant le départ et l'arrivée n'est pas un sous-tours et que son cardinal est supérieur ou égal à Amin, le programme s'arrête.

A chaque étape i, on résout le problème P_i constitué des contraintes ci-dessus + les i contraintes de sous tours ajoutés correspondant aux sous ensembles $(S_k)_{1 \leq k \leq i}$:

$$\sum_{i \in S_k, j \in S_k} \delta_{ij} \leq |S_k| - 1, \forall k \in [\![1, i]\!]$$

.

3 Résultats

On résume les résultats obtenus pour les deux formulations dans les deux tableaux qui suivent pour différentes instances du problème :

Instance	temps (s)	itérations	Branch n Bound	Objectif	borne inf à la racine
n=6 a=5 d=1	0.01	13	0	12	9.22
n=20 a=15 d=1	2.68	78821	8093	96	38.2
n=20 a=10 d=1	0.34	16174	1919	90	58
n=20 a=10 d=19	4.39	139320	16067	91	34.4
n=30 a=10 d=19	25,52	713140	55322	130	54.6
n=40 a=8 d=27	3,01	84563	8899	114	77.25
n=50 a=34 d=49	36,87	365076	143749	168	93.24

Table 1: Résultats pour la formulation polynomiale

Instance	temps (s)	n sous_tours	Objectif	borne inf à la racine
n=6 a=5 d=1	0.16	3	12	12
n=20 a=15 d=1	14.25	75	96	61.16
n=20 a=10 d=1	14.43	96	90	61.33
n=20 a=10 d=19	104.44	207	91	49
n=30 a=10 d=19	1647.52	461	130	78
n=40 a=8 d=27	45.53	64	114	102.59
n=50 a=34 d=49	593.08	166	168	115.22

Table 2: Résultats pour la formulation exponentielle

4 Comparaison des deux formulations

En terme de temps

La formulation exponentielle prend beaucoup plus de temps pour la résolution. En effet, sa connexité est assurée par un processus itératif alors que pour la polynomiale la connexité est assurée par un ensemble de contraintes. En terme de temps seulement, la formulation polynomiale est donc meilleure.

En terme de borne à la racine

La formulation exponentielle en relaxation continue fournit une borne inférieure à la racine bien meilleure que celle de la formulation exponentielle. C'est en effet un des avantages de la formulation exponentielle, du fait qu'elle borne la solution exacte intière plus facilement.

Aller plus loin

Nous n'avons pas pu utiliser des instances avec un nombre de sommets plus grand que 50. En effet, le temps de calcul est très important, et le stockage des résultats intermédiaires (branch and bound par exemple dans la formulation polynomiale) dépasse la capacité des machines. Celà peut être corrigé en optimisant les formulations de nos modèles en minimisant les espaces à parcourir au maximum. Nous n'avons pas eu le temps de parcourir ces possibilités.