Optimisation discrète - Course d'avions

Quentin Burthier
ENSTA Paris
quentin.burthier@ensta-paris.fr

Antoine Gauthier
ENSTA Paris
antoine.gauthier@ensta-paris.fr

Abstract

Nous étudions un problème d'optimisation semblable au voyageur de commerce. Nous le modélisons comme un programme linéaire en nombres entiers 0-1, puis nous comparons deux algorithmes de résolution basés sur des formulations différents des contraintes de sous-tours.

1 Introduction

1.1 Notations

On note le symbole de Kronecker δ.

Une instance du problème est composée des données suivantes :
- Nombre d'aérodromes : n
- Aérodrome de départ : s [1]
- Aérodrome d'arrivée : a
- Nombre d'aérodromes à parcourir : A_{min}
- Nombre de régions : m
- Région de l'aérodrome i : l_i [2]
- Distance maximale de vol sans se poser : R
- Coordonnées cartésiennes de l'aérodrome i : (x_i, y_j)

On suppose que les aérodromes se situent sur un plan et que la distance entre deux aérodromes est la distance euclidienne. On note $d_{ij} := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ la distance entre l'aérodrome i et l'aérodrome j.

2 Modélisation

2.1 Graphe du problème

On modélise le problème par un graphe orienté, ayant pour sommets les aérodromes. Deux aérodromes sont reliés par une arête si et seulement si la distance entre ces aérodromes est inférieure à la distance de vol maximale.

Le graphe est donc $G = (V, E)$ où $V = \{1, \dots, n\}$ etc $E = \big\{(i, j) \in V^2, i \neq j : d_ij \leq R\big\}$.

2.2 Objectif

On cherche à minimiser la distance parcourue, soit

$$\mathcal{Z} = \min \sum_{i,j:(i,j)\in E} x_{ij} d_{ij} \tag{1}$$

où $x_{ij} \in \{0, 1\}$.

1. pour *start*
2. pour *Land*

2.3 Contraintes du problème

Unicité des visites Chaque aérodrome est visité au plus une fois, sauf les aéroports d'arrivée et de départ qui sont visités exactement une fois.

$$\forall i \notin \{s, a\}, \sum_{j:(i,j)\in E} x_{ij} \leq 1 \tag{2a}$$

$$\forall i \notin \{s, a\}, \sum_{j:(j,i)\in E} x_{ji} \leq 1 \tag{2b}$$

Connexité L'avion doit partir de s

$$\sum_{j:(s,j)\in E} x_{sj} = 1 \tag{3a}$$

$$\sum_{j\neq a:(j,s)\in E} x_{js} = 0 \tag{3b}$$

$$x_{as} = \delta_{as} \tag{3c}$$

et arriver en a

$$\sum_{j:(j,a)\in E} x_{ja} = 1 \tag{3d}$$

$$\sum_{j\neq a:(a,j)\in E} x_{aj} = 0 \tag{3e}$$

$$\tag{3f}$$

La contrainte 3c autorise que les aérodromes d'arrivée et de départ soient identiques. Cela suppose que $d_{as} \leq R$, sinon le problème n'est par réalisable.

Un avion repart d'un aérodrome seulement si et seulement si il s'est posé à cet aérodrome.

$$\forall i \notin \{s, a\}, \sum_{j:(j,i)\in E} x_{ji} = \sum_{j:(i,j)\in E} x_{ij} \tag{3g}$$

Nombre minimum d'aérodromes visités L'avion doit visiter au moins A_{min} aérodromes.

$$\sum_{i,j:(i,j)\in E} x_{ij} \geq A_{\text{min}} - 1 + \delta_{sa} \tag{4}$$

Ici encore δ_{sa} permet de tenir compte du cas où $s = a$ et que la solution est un cycle.

Diversité des régions La contrainte de visiter au moins une fois chaque région une fois s'écrit, en notant $\mathcal{A}_l := \{i : l_i = a\}$ l'ensemble des aéroports appartenant à la région a, et $\mathcal{L}$ l'ensemble des régions comportant au moins un aérodrome.

$$\forall l \in \mathcal{L} \setminus \{l_s, l_a\}, \sum_{i\in\mathcal{A}_l:(i,j)\in E} x_{ij} \geq 1 \tag{5}$$

On a supprimé les contraintes sur les régions r_s et r_a, qui sont redondantes avec les contraintes d'arrivée et de départ.

2.4 Contraintes d'élimination des sous-tours

Il nous reste à supprimer la possibilité d'avoir solutions composées de plusieurs cycles.

Nombre polynomial de contraintes

$$\forall i \neq s, j \neq a : (i,j) \in E,\ u_j \geq u_i - n(1 - x_{ij}) \tag{6a}$$

$$u_i \in \mathbb{Z} \tag{6b}$$

Ceci force $u_i > u_j$ quand l'avion va de i à j, et interdit donc les cycles.

Nombre exponentiel de contraintes On peut aussi interdire les sous-tours au sein de tous les sous-graphes.

$$\forall S \subset G, \sum_{i \in S, j \in S} x_{ij} \leq |S| - 1 \tag{7}$$

Cependant, le nombre de sous-graphe croit exponentiellement avec la taille du graphe.

Il faut donc générer donc des contraintes à chaque nœud de l'algorithme *branch & cut*. Pour cela on détecte d'abord les contraintes violées parmi (7), puis on ajoute ces contraintes au modèle.

3 Étude numérique