| | |
|---|---|
| **Titre:** Title: | Importance des variables et régressions statistiques imitant l'heuristique de branchement fort dans un problème de rotation d'équipages |
| **Auteur:** Author: | Emeric Courtade |
| **Date:** | 2022 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Courtade, E. (2022). Importance des variables et régressions statistiques imitant l'heuristique de branchement fort dans un problème de rotation d'équipages [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10471/ |

**POLYTECHNIQUE MONTRÉAL**

affiliée à l'Université de Montréal

# Importance des variables et régressions statistiques imitant l'heuristique de branchement fort dans un problème de rotation d'équipages

**EMERIC COURTADE**

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie informatique

Août 2022

**POLYTECHNIQUE MONTRÉAL**

affiliée à l'Université de Montréal

Ce mémoire intitulé:

# Importance des variables et régressions statistiques imitant l'heuristique de branchement fort dans un problème de rotation d'équipages

présenté par **Emeric COURTADE**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

**Martine BELLAÏCHE**, présidente
**Daniel ALOISE**, membre et directeur de recherche
**François SOUMIS**, membre et codirecteur de recherche
**Frédéric QUESNEL**, membre externe

# REMERCIEMENTS

Je tiens à remercier sincèrement mon directeur de recherche, *M. Daniel Aloise*, pour le soutien qu'il m'a apporté tout au long de cette maîtrise, et pour l'expérience que j'ai pu acquérir grâce à lui.

Mes pensées vont aussi à mon co-directeur de recherche, *M. François Soumis,* et à *M. Frédéric Quesnel* pour leur aide dans la prise en main du logiciel GENCOL et dans la compréhension de certaines subtilités de la recherche opérationnelle.

Je remercie mon collègue et ami *Pierre Pereira,* qui a fait preuve d'une patience sans faille lors de mes nombreuses sessions de questionnements.

Mes remerciements sont tout particulièrement adressés à mes parents qui m'ont subtilement poussé à poursuivre mes études un peu plus longtemps et à venir découvrir la vie Montréalaise. Cette aventure n'aurait pas non plus été la même sans *Maëlle*, *Gauthier*, *Alice*, *Enzo*, *Matthieu*, *Guilhem*, *Ilan*, *Lucie* et *Tess*. Merci à vous d'avoir pu me faire profiter de cette expérience et de m'avoir laissé des souvenirs inoubliables.

Enfin, je remercie l'entreprise Ad Opt pour avoir financé mes travaux de recherche et l'École Polytechnique de Montréal pour m'avoir donné la chance de suivre ce parcours exceptionnel.

# RÉSUMÉ

Afin d'optimiser l'ensemble du processus de planification des plans de vols, les compagnies aériennes ont recours à des solveurs de plus en plus performants. Une des étapes de ce processus est la construction de rotations d'équipages où l'on cherche à trouver les ensembles de vols minimisant les coûts. Ces rotations doivent répondre à un ensemble de règles définies par des conventions entre la compagnie et ses employés afin d'être considérées valides.

Les techniques de recherche opérationnelle produisent aujourd'hui des solutions relativement bonnes en prenant en compte l'ensemble de ces contraintes. Le problème est résolu sous forme d'un arbre de *Branch-and-Bound*, et plus particulièrement un arbre de *Branch-and-Price*. En effet, à cause du grand nombre de rotations entrant en jeu dans la définition du problème, il n'est pas possible de résoudre ce dernier sous sa forme initiale. La méthode de la génération de colonnes est utilisée à chacun des nœuds de l'arbre de *Branch-and-Price* pour résoudre le problème sous une forme réduite et ainsi explorer efficacement l'arbre en des temps réduits. Des méthodes dites de *diving* peuvent aussi être utilisées pour réduire encore plus les temps de résolution: une fois qu'un nœud a été exploré, il n'est plus possible de revenir sur des nœuds précédemment mis de côté plus haut dans l'arbre. L'espace de recherche s'en retrouve ainsi grandement réduit.

Le choix de la variable à fixer lors de la résolution de l'arbre de *Branch-and-Price* peut se faire grâce à plusieurs heuristiques. La plus utilisée par les compagnies aériennes aujourd'hui est celle de la fixation de colonne qui consiste à brancher sur la colonne ou rotation ayant la plus grande valeur fractionnaire. Sa rapidité et ses bons résultats la rendent facilement exploitable dans un environnement industriel. L'heuristique du branchement fort propose quant à elle des résultats bien plus proches de l'optimalité mais souffre d'un temps d'exécution très élevé. Elle considère en effet un certain nombre de rotations candidates qui la mènent à résoudre successivement plusieurs problèmes linéaires avant d'étendre l'arbre. Cette particularité la rend donc inexploitable dans la pratique.

Afin de combiner les très bonnes performances du branchement fort et la rapidité de la fixation de colonne dans notre contexte de *Branch-and-Price* et de *diving*, l'utilisation d'un modèle d'apprentissage machine se présente comme une solution adéquate. Plus précisément, l'apprentissage par imitation est capable de prendre des décisions similaires à celle d'un expert, le

branchement fort. Un tel modèle n'a cependant pas besoin de résoudre chacun des nœuds associés aux rotations candidates, réduisant ainsi drastiquement son temps d'exécution. Afin de fournir un résultat, le modèle doit avoir à sa disposition des informations sur les candidats. Le choix de ces informations, ou variables, est crucial pour que les prédictions soient les plus justes possibles.

Le premier objectif de cette maîtrise est ainsi de déterminer les variables les plus importantes pour la prise de décision dans un arbre de *Branch-and-Price* exploitant le *diving* pour le problème de rotations d'équipages. Par le biais de plusieurs méthodes telles que l'analyse de coefficients de régressions, la sélection itérative ou la permutation de variables, nous montrons que quatre de nos variables semblent être d'une importance significative.

Le second objectif traité dans ce mémoire est l'évaluation de modèles simples de régressions linéaires entraînés à imiter l'heuristique de branchement fort. Nous montrons des résultats encourageants sur des instances comprenant plusieurs milliers de vols. Les solutions obtenues révèlent un bon compromis entre la fixation de colonne et le branchement fort tout en présentant des temps d'exécution similaires à ceux de la fixation de colonne.

# ABSTRACT

To optimize the whole planning process of their flight plans, airlines employ state-of-the-art solvers. One of the steps of this process consists in the building of crew pairings where one aims to find the lists of pairings that minimize operation costs. These pairings must satisfy a set of rules defined by collective agreements and airline regulations between the company and its employees to be judged valid.

Nowadays, operational research techniques generate very good solutions while considering these constraints. The problem is solved as a Branch-and-Bound tree, and especially a Branch-and-Price tree. Indeed, because of the large numbers of pairings present in the problem, it is not possible to solve it under its initial form. The column generation method is used at each node of the Branch-and-Price tree to solve the problem under a reduced form and thus efficiently explore the tree with a lower computational time. Diving methods can also be used to reduce even more the resolution time: once a node has been explored, it is not possible to come back to previous nodes which have been put aside higher in the tree anymore. The search space is thus greatly reduced.

The choice of the variable to fix during the resolution of the Branch-and-Price tree can be done thanks to several heuristics. The most used by airlines today is column fixing where we chose to branch on the column which has the highest fractional value. Its speed and its good results make it easily exploitable in an industrial environment. On the other side, the strong branching heuristic generates optimal or near-optimal results but suffer from a very high computational time. It considers a certain number of candidate pairings which lead it to solve successively several linear problems before expanding the tree. This particularity makes it unexploitable in practice.

To combine the very good results of strong branching and the speed of column fixing in a Branch-and-Price and diving context, the use of a machine learning model appears promising. Specifically, imitation learning can take decisions similar to those of an expert, the strong branching. However, such a model does not need to solve each candidate node, reducing drastically the computational time. To produce a result, the model needs information about the candidates. The choice of those features is crucial to have the most precise predictions.

The first goal of this dissertation is to determine the most important features to take a decision in a Branch-and-Price tree using a diving method for the crew pairing problem. Thanks to several methods such as the analysis of regression coefficients, iterative selection, or features permutation, we show that four of our variables seem to be of high significance.

The second goal tackled in the work is the evaluation of simple linear regression models trained to imitate the strong branching heuristic. We reveal encouraging results on instances containing several thousand flights. The solutions obtained show a good tradeoff between column fixing and strong branching while presenting computational times similar to those of column fixing.

# TABLE DES MATIÈRES

# LISTE DES TABLEAUX

# LISTE DES FIGURES

# LISTE DES SIGLES ET ABREVIATIONS

AIC Akaike Information Criterion (ou critère d'information d'Akaike)

BIC Bayesian Information Criterion (ou critère d'information bayésien)

B&B Branch-and-Bound

B&P Branch-and-Price

CAP Crew Assignment Problem

CPP Crew Pairing Problem

CSP Crew Scheduling Problem

ERV Élimination Récursive des variables

FCC Fraction Conflicting Columns

FCCPV Fraction Conflicting Columns with Positive Value

FPTF Frac Pairing Tasks Fixed

M3C Min Cost Conflicting Column

M3CPV Min Cost Conflicting Column with Positive Value

MAE Mean Absolute Error (ou erreur absolue moyenne)

ML Machine Learning (ou apprentissage machine)

MSE Mean Squared Error (ou erreur quadratique moyenne)

PA Précision Approximative

RF Random Forest (ou forêt aléatoire)

RMP Resticted Master Problem (ou problème maître restreint)

RO Recherche Opérationnelle

SB Strong Branching (ou branchement fort)

SSV Sélection Séquentielle des variables

# LISTE DES ANNEXES

# CHAPITRE 1 INTRODUCTION

Alors que la pandémie de la Covid-19 a fait connaître une crise sans précédent au domaine de l'aviation ces deux dernières années, le nombre de voyageurs aériens était à son pic en 2019 avec plus de 4.5 milliards de passagers contre seulement 1 milliard 30 ans plus tôt [1]. L'association du transport aérien international a indiqué que le flux de passagers en 2021 n'atteignait que 47% de celui de 2019, mais elle prévoit un retour à la normale d'ici 2024 [2]. Un nombre aussi élevé de voyageurs impose aux compagnies aériennes une logistique très complexe pour satisfaire la demande de leurs clients tout en dégageant un profit nécessaire à leur fonctionnement structurel. Parmi les frais qu'éprouvent les compagnies, ceux associés au carburant sont les plus importants mais ils sont suivis de près par les coûts d'équipage. Un des principaux objectifs des transporteurs aériens va alors être de minimiser ces derniers en cherchant notamment à optimiser les horaires des membres d'équipage. Ce problème est aussi connu sous le nom de *Crew Scheduling Problem* (CSP) et il est depuis de nombreuses années au cœur de la recherche dans le domaine de l'aviation. L'objet de ce mémoire sera l'étude du problème des rotations d'équipages ou *Crew Pairing Problem* (CPP), sous-problème du CSP.

Nous allons maintenant voir les composantes du CPP ainsi que sa place dans le processus de planification des compagnies aériennes et sa formalisation du point de vue de la recherche opérationnelle (RO).

## 1.1 Processus de planification d'une compagnie aérienne

Le processus de planification d'une compagnie aérienne se définit le plus souvent en 5 étapes résolues de façon séquentielle (figure 1.2). Dans un premier temps, les horaires de vols sont créés. Des avions sont ensuite affectés à chacun d'entre eux en respectant des contraintes de disponibilité et de coût. Le problème suivant consiste à déterminer la séquence de vols suivie par les avions afin d'en faciliter leur entretien. Enfin, les deux dernières étapes correspondent au CSP que nous avons introduit précédemment. Elles représentent respectivement le CPP et le *Crew Assignment Problem* (CAP).

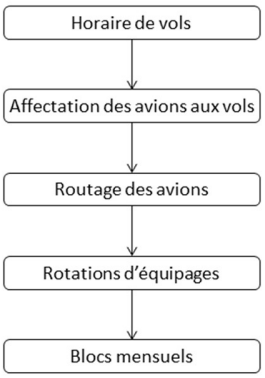


*Figure 1.1: Processus de planification d'une compagnie aérienne*

Le but du CPP est de construire des rotations anonymes à partir de l'ensemble des segments de vol définis lors des précédentes étapes tout en minimisant les coûts associés. Une rotation est un ensemble de vols séparés par des périodes de repos. Elle doit nécessairement commencer et se terminer dans une même base. La construction de ces rotations doit toutefois respecter des contraintes définies par la sécurité aérienne ainsi que par des conventions entre la compagnie et ses employés. Les membres d'équipage ne peuvent pas par exemple dépasser un certain nombre d'heures de vol par jour dans une rotation. Ils ne peuvent pas non plus dépasser un certain temps d'attente en correspondance, auquel cas la compagnie doit prendre en charge leur frais de logement lors des phases dites de *layover*.

Quant à lui, le CAP consiste à affecter les membres d'équipage aux rotations venant du CPP. Les contraintes de ce problème viennent du fait qu'un membre d'équipage ne peut être affecté qu'à certaines positions dans l'équipage. De plus, c'est à cette étape que sont prises en compte des contraintes plus personnelles comme les jours de congé ou les préférences des employés. Il existe ainsi trois approches (anonyme, personnalisée et personnalisée avec séniorité) couramment utilisées aujourd'hui considérant la situation de chacun des membres d'équipage.

## 1.2 Formalisation du CPP

Avant d'évoquer la formulation mathématique du CPP, nous allons expliciter les termes techniques employés dans ce domaine:

- Une base est un point de départ ou d'arrivée d'une rotation. Chaque membre d'équipage est rattaché à une base.
- Un segment de vol (*flight*) correspond à un vol entre deux aéroports et se définit par ses horaires de départ et d'arrivée.
- Une connexion (*sit time*) est le temps passé par l'équipage entre deux vols successifs.
- Un service de vols (*duty*) correspond à une journée de travail. Il peut donc comprendre plusieurs vols reliés par des connexions.
- Un *layover* permet de séparer deux services de vols. Une connexion devient un *layover* lorsque le temps d'attente est supérieur à 8 heures.
- Une rotation (*pairing*) est un enchaînement de plusieurs services de vols et *layovers*, généralement sur 2 ou 3 jours pour les courts et moyens courriers. La contrainte principale d'une rotation est qu'elle doit commencer et se terminer dans la même base. De plus, une rotation est dite réalisable si elle répond aux contraintes fixées par la régulation aérienne.
- Un vol de repositionnement (*deadhead*) a lieu lorsqu'un membre d'équipage doit réaliser un vol en tant que passager. Ce type de vol est coûteux pour la compagnie mais il peut permettre d'obtenir de meilleures solutions lors de la résolution du CPP.

La figure suivante illustre l'architecture que peut avoir une rotation:

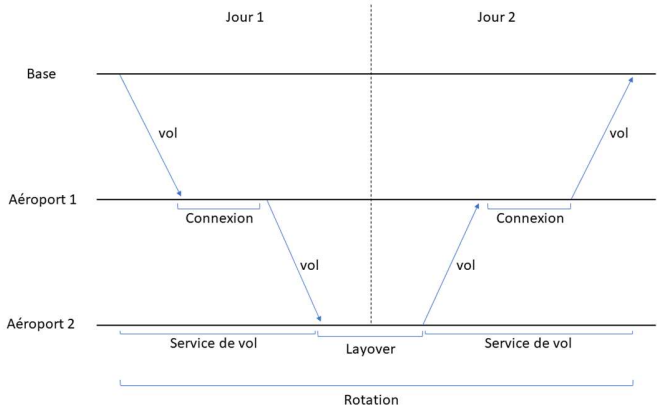


*Figure 1.2: Architecture d'une rotation d'équipage*

Soit $R$ l'ensemble des rotations réalisables, c'est-à-dire qui répondent aux contraintes fixées par la sécurité aérienne et les conventions entre la compagnie et ses employés, et $V$ l'ensemble des vols réalisables, le CPP se définit de la façon suivante:

$$\min \sum_{r \in R} c_r x_r$$

sous contraintes:

$$\sum_{r \in R} A_{v,r} x_r = 1 \qquad \forall \, \mathrm{v} \in V$$

$$x_r \in \{0,1\} \qquad \forall \, \mathrm{r} \in R$$

Le CPP est un problème de minimisation où la compagnie aérienne cherche à réduire les coûts engendrés par les rotations réalisables. $c_r$ correspond au coût de la rotation $\mathrm{r} \in R$ et n'est comptabilisé que si cette rotation est réalisée ($x_r = 1$). La matrice $A$ est une matrice binaire où le coefficient $A_{v,r}$ indique si le vol $v$ est présent dans la rotation $r$. La contrainte d'égalité nous indique que nous travaillons sur un problème de partitionnement et qu'un vol ne doit être couvert qu'une et une seule fois. Une autre formulation avec contrainte d'inégalité est possible (problème de couverture) ce qui permet à un vol d'être couvert plusieurs fois et donne lieu à l'ajout des vols de repositionnement au problème. Dans la suite de ce mémoire, nous traiterons uniquement le cas du problème de partitionnement.

## 1.3 Contribution de la maîtrise

Bien que les résultats obtenus aujourd'hui avec les méthodes traditionnelles de RO soient très proches de l'optimalité, les temps de résolution nécessaires à leur obtention sont cependant très élevés. En effet, le branchement fort ou *strong branching* (SB) est aujourd'hui une des meilleures heuristiques quant à l'optimalité des résultats qu'elle fournit, mais elle fait preuve d'un temps d'exécution trop grand pour qu'elle soit réellement utilisée en pratique. Cette heuristique est déployée lors de la phase de sélection de variable dans un arbre de *Branch-and-Bound* (B&B), phase relativement importante dans le processus de résolution d'un problème comme le CPP par exemple.

L'évolution continue de l'apprentissage machine, ou *machine learning* (ML), a permis de trouver des applications concrètes dans le domaine de la RO et plus particulièrement dans le cas du CPP. Un sous-domaine du ML, l'apprentissage par imitation, consiste à imiter les décisions d'une heuristique couramment utilisée en RO, autrement appelée expert. De manière générale, l'expert choisi est le SB. Le but est ainsi d'utiliser le ML pour avoir des résultats similaires à cette heuristique mais en un temps réduit.

L'approche développée dans ce mémoire est semblable puisqu'elle aura pour but de remplacer cette heuristique de SB par un des plus simples modèles de ML, la régression linéaire. Notons toutefois que la différence majeure avec l'état de l'art est que la résolution de notre arbre se fait par *strong diving* tout en utilisant de la génération de colonnes. Plus précisément, on ne considère qu'un sous-ensemble dynamique de rotations au cours de la résolution. De plus, une fois qu'une décision est prise à un nœud parmi un sous-ensemble de nœuds enfants, il n'est plus possible de revenir à des décisions précédentes afin d'explorer de potentiels meilleurs espaces de recherche. Une seconde méthode dite de *pure diving,* est aussi utilisée afin d'obtenir des résultats moins optimaux mais en des temps réduits. Cette approche plus simple peut être considérée comme gloutonne puisqu'elle ne prend en compte que la valeur fractionnaire des candidats pour faire son choix.

Ce mémoire a alors pour but de répondre à deux objectifs: le premier et plus important sera de déterminer, parmi un ensemble de caractéristiques ou *features* définissant les nœuds, celles qui donnent le plus d'informations lors du choix de branchement dans l'arbre. L'interprétabilité naturelle de la régression linéaire ainsi que plusieurs méthodes de réduction de dimensionnalité nous permettront de répondre à ce problème. Le second objectif sera de voir si notre régression linéaire permet de trouver un compromis entre optimalité de la solution et temps de résolution. Nous chercherons ainsi à voir si elle est capable d'obtenir des résultats similaires à ceux du *strong diving* en un temps de *pure diving*.

# CHAPITRE 2 REVUE DE LITTÉRATURE

Dans la première section de cette partie, nous allons aborder différentes méthodes de RO permettant de résoudre le problème de rotation d'équipage. Le demi-siècle de recherche dans ce domaine a donné lieu à des techniques relativement efficaces pour résoudre des instances dont la taille ne fait qu'augmenter.

Dans la seconde section, nous reviendrons sur les origines de la régression linéaire et les différentes méthodes qui s'y rapportent afin d'analyser l'importance des variables étudiées. Nous évoquerons aussi quelques exemples d'applications de ML dans le domaine du B&B.

## 2.1 Méthodes traditionnelles de construction de rotations d'équipage

### 2.1.1 B&B et heuristiques de sélection de variable

Le CPP est un problème de programmation linéaire en nombres entiers . Une formalisation plus générale que celle définie dans la section 1.2 et s'appliquant à tous les problèmes de ce type est la suivante:

$$\min \boldsymbol{c}^T \boldsymbol{x}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

$$\boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{x} \in Z^n$$

Ce type de problème est généralement résolu en utilisant la méthode du B&B, technique proposée par Land et al. [3] et la programmation linéaire. En partant d'un nœud racine correspondant à une solution non optimale, on résout le problème linéaire associé, grâce à la méthode du simplexe par exemple, et on procède itérativement en sélectionnant une variable du problème sur laquelle se brancher puis en choisissant parmi les nouveaux nœuds celui qui a le plus de potentiel (figure 2.1). Certaines règles permettent d'élaguer l'arbre afin d'éviter l'exploration des espaces non-optimaux. Une fois que l'arbre est entièrement exploré, nous sommes en mesure de donner une solution optimale. Différentes heuristiques existent lors de la sélection de la variable et du nœud afin de jouer sur un compromis entre temps de résolution et optimalité de la solution.

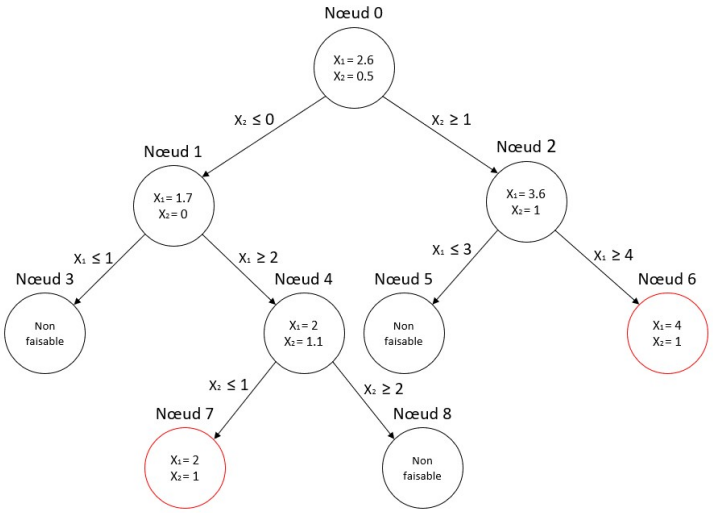


*Figure 2.1: Processus de résolution d'un arbre de B&B. Les cercles rouges correspondent aux solutions entières*

La plupart des méthodes de résolution ont été développées dans le but d'une résolution exacte. Leur objectif est donc de se brancher sur la variable proposant la meilleure amélioration de la borne inférieure afin d'obtenir une solution optimale. Cette borne inférieure est obtenue en résolvant la forme duale du problème primal exprimé précédemment. Il est toutefois possible d'adopter une approche plus heuristique où les branchements se font sur la variable augmentant le moins cette borne. L'intérêt est d'explorer des branches où l'on trouvera une bonne solution entière.

Parmi les méthodes de sélection de variable, les plus connues et utilisées d'après Furian et al. [4] et Huang et al. [5] sont les suivantes:

- Le *most infeasible branching* consiste à brancher sur la variable la plus fractionnaire.
- Le pseudocost branching [6] garde en mémoire un historique des actions prises lors de précédents branchements. Cela permet d'obtenir une estimation du gain pour chaque candidat. Cette méthode montre toutefois quelques limites en début de résolution car le manque de décisions rend l'historique peu fiable.
- Le SB est présenté pour la première fois par Applegate et al. [7] et consiste à résoudre le problème linéaire pour chacun des candidats afin d'obtenir celui qui mène à la meilleure amélioration de la borne duale. On parle de *full strong branching* lorsque la résolution se fait jusqu'à optimalité sur tous les candidats mais, bien que cette méthode donne de très

bons résultats notamment concernant la taille des arbres [8], elle est cependant très coûteuse en temps de calcul. Dans la pratique, on utilisera donc une version simplifiée qui ne considérera qu'un sous-ensemble de candidats et qui ne résoudra pas nécessairement les problèmes jusqu'à optimalité.

- Le *reliability branching* [9] combine les méthodes de *pseudocost* et de SB afin d'en retirer leurs avantages respectifs. Lorsque l'estimation du score d'une variable faite par le *pseudocost*, n'est pas assez fiable, typiquement en début d'arbre, on utilise le SB pour la résolution. Dans le cas où le candidat a déjà été sélectionné plusieurs fois par le passé, on pourra considérer son estimation fiable et ainsi utiliser le *pseudocost* afin d'accélérer la résolution.
- Le branchement hybride [10] est une méthode utilisant cinq scores différents afin d'appuyer son choix de variable. Les résultats obtenus avec cette approche sont relativement bons et c'est d'ailleurs pourquoi le solveur SCIP en a fait sa méthode de sélection de variable par défaut.

Parmi les méthodes précédentes, la première est uniquement utilisée dans le but de brancher sur la variable offrant la meilleure amélioration de la borne inférieure. Les autres approches, bien que développées initialement pour répondre à ce même objectif, sont adaptables au cas d'un branchement sur la variable augmentant le moins la borne inférieure et sont alors considérées heuristiques.

La seconde partie du XXème siècle a vu l'émergence d'une nouvelle technique, appelée génération de colonnes, permettant de prendre en charge des problèmes de grandes tailles dans le nombre de variables. Lorsque la génération de colonnes est associée au B&B, on parle alors de *Branch-and-Price* (B&P) comme l'a décrit Barnhart et al. [11]. Le principe de cette méthode est de ne considérer dans un premier temps qu'un sous-ensemble de colonnes du problème maître, ou rotations d'équipages dans le cadre du CPP, afin d'alléger les calculs, et de les placer dans le problème maître restreint (RMP). À chaque nœud de l'arbre, le RMP est résolu et il convient ensuite de résoudre un sous-problème dit de *pricing* grâce aux valeurs duales associées à la résolution. Le sous-problème de *pricing* consiste à trouver des colonnes à coûts réduits négatifs grâce à un algorithme de plus court chemin avec contraintes de ressources, où chaque chemin

correspond à une rotation. Si des colonnes à coûts réduits négatifs sont trouvées, elles sont ajoutées au RMP qui est de nouveau résolu et soumis au *pricing*. Ce processus est répété jusqu'à ce qu'il n'y ait plus de colonnes à coûts réduits négatifs ou jusqu'à rencontre d'un critère d'arrêt. La méthode du B&P a d'abord été appliquée au problème de tournées de véhicules par Desrosiers et al. [12] avant de faire ses preuves avec le CPP. Lavoie et al. [13] utilise ainsi cette méthode sur des instances comprenant jusqu'à 329 segments de vol et réussit à obtenir une amélioration de 4 à 5% des résultats déjà existants. Vance et al. [14] propose une version heuristique du B&P pour le CPP où il n'est plus nécessaire d'explorer tous les nœuds. En procédant de cette manière, des instances allant jusqu'à plus de 2000 segments de vol ont pu être résolues.

Par la suite, d'autres techniques ont été ajoutées au B&B afin d'en améliorer la rapidité d'exécution ainsi que les résultats obtenus. Sadykov et al. [15] présente par exemple des méthodes dites de *diving* permettant de réduire efficacement l'espace de recherche dans un arbre de B&P. Le principe est de brancher sur une ou plusieurs variables en réoptimisant le problème linéaire à chaque fois. Il n'est toutefois pas permis de faire de retour sur trace, c'est-à-dire de revenir sur des décisions prises plus haut dans l'arbre, sauf sous certaines conditions. Cela peut poser des limitations concernant l'optimalité de la solution mais le temps de calcul s'en trouve généralement réduit. Ces méthodes sont dites heuristiques car elles n'assurent pas de résoudre le problème jusqu'à optimalité. Les auteurs présentent ainsi plusieurs méthodes dont les deux suivantes que nous retrouverons dans notre travail de recherche sous des formes légèrement altérées:

- Le *pure diving* consiste à brancher sur la variable dont la valeur fractionnaire est la plus proche d'une valeur entière non nulle. Il est possible de brancher sur plusieurs variables en même temps afin d'accélérer le processus mais les résultats sont habituellement meilleurs lorsqu'une seule variable est retenue.
- Le *strong diving* se base sur l'heuristique du SB et va sélectionner un sous-ensemble de $n_c$ candidats. On branche temporairement sur chacun des candidats un à un en résolvant le problème linéaire à chaque fois. Le candidat retenu est celui qui présente la plus petite détérioration de la borne duale, contrairement à l'heuristique originale qui observe l'amélioration de cette même borne comme expliqué précédemment. Les candidats sont choisis sur le même critère que pour le *pure diving*, c'est-à-dire en retenant les $n_c$ variables dont la valeur fractionnaire est la plus proche d'une valeur entière non nulle.

### 2.1.2 Méthodes de résolution du CPP

À l'origine, le CPP était résolu sur différentes fenêtres de temps comme le présente Klabjan [16]. En effet, on considérait dans un premier temps le problème sur une journée. Les rotations ainsi obtenues étaient supposées se retrouver régulièrement d'un jour à l'autre. La deuxième étape se faisait dans les mêmes conditions mais à l'échelle d'une semaine pour finalement résoudre le problème à l'échelle du mois entier. Cette approche en trois phases se base sur la régularité des rotations mais n'est toutefois pas la plus optimale. Saddoune et al. [17] a ainsi proposé une évolution de cette méthode par le biais des horizons roulants. Le principe est de résoudre plusieurs fenêtres de temps, typiquement de quelques jours, en autorisant la superposition de ces dernières. Ainsi, uniquement les vols commençant dans une fenêtre de temps peuvent être considérés pour la construction des rotations de cette fenêtre. De plus, l'ajout de contraintes sur les conditions initiales permet la continuité entre les rotations des fenêtres successives.

Comme nous l'avons évoqué précédemment, le CPP et le CAP sont traditionnellement résolus l'un à la suite de l'autre. Cette approche présente l'avantage d'être bien plus rapide à résoudre que dans le cas où le CSP serait résolu en un seul bloc car on se limite à deux problèmes de tailles réduites. Cependant, des méthodes comme celles de Saddoune et al. [18] ou Özener et al. [19] ont été étudiées afin d'intégrer les deux problèmes dans une seule phase de résolution. En agissant de cette manière, les résultats obtenus sont plus proches de l'optimalité car la construction des rotations peut prendre en compte les contraintes imposées par l'assignation des membres d'équipage. L'inconvénient majeur de cette approche réside dans son temps de calcul très élevé et augmentant de façon exponentielle avec la taille du problème.

## 2.2 Régression linéaire et sélection de modèles

Un de nos objectifs est de remplacer notre heuristique de branchement actuelle pour gagner en rapidité tout en conservant des résultats proches de l'optimalité. Les méthodes employées de nos jours se concentrent sur le ML et c'est donc pourquoi nous allons orienter notre étude sur une approche similaire. La régression linéaire est une méthode statistique à la limite du domaine du ML. Sa simplicité permet d'avoir une bonne interprétabilité de ses prédictions tout en offrant une *baseline* acceptable pour les résultats que nous cherchons à obtenir.

### 2.2.1 Description de la régression linéaire

La régression linéaire voit ses origines remonter au XVIII$^{ème}$ siècle et est aujourd'hui encore très documenté. De nombreux livres [20]–[22] évoquent en détail toutes les subtilités qui lui sont associées. Hastie et al. [23] décrivent une régression linéaire par la formulation suivante:

$$\hat{\mathbf{y}} = \mathrm{f}(\boldsymbol{X}) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

où $\boldsymbol{X}$ est la matrice des données contenant N échantillons définis par $p$ variables et les $\beta_j$ correspondent aux paramètres ajustables du modèle. Ainsi, $\hat{\boldsymbol{y}}$ correspond à la prédiction du modèle que l'on pourra comparer ensuite à la valeur réelle $\boldsymbol{y}$.

Définissons maintenant la somme résiduelle des carrés (RSS), une des métriques donnant la performance du modèle:

$$\mathrm{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

La différence $y_i - \hat{y}_i$ correspond à la distance entre la prédiction et la valeur réelle. En notant $\boldsymbol{y} = \mathrm{f}(\boldsymbol{X}) + \boldsymbol{\epsilon}$, cette distance peut aussi s'écrire sous la forme d'un terme d'erreur $\boldsymbol{\epsilon}$. L'objectif est donc d'ajuster les paramètres du modèle afin de minimiser cette erreur et d'obtenir des prédictions proches des valeurs réelles. En supposant que $\boldsymbol{X}$ est une matrice de plein rang, c'est-à-dire qu'aucune de ses colonnes n'est linéairement dépendante des autres, on obtient une unique solution pour $\boldsymbol{\beta}$ telle que:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

Plusieurs hypothèses sont formulées dans le cadre de la régression linéaire:

- L'absence de multi-colinéarité entre les variables explicatives demande qu'une variable ne puisse pas s'écrire sous la forme d'une combinaison linéaire d'autres variables.
- Les termes d'erreur $\boldsymbol{\epsilon}$ ont une variance constante (homoscédasticité)
- Les variables explicatives ne sont pas corrélées aux termes d'erreur
- Les termes d'erreur sont indépendants les uns des autres

- Au-delà de supposer les termes d'erreur de variance constante, on peut considérer qu'ils suivent une loi normale centrée.

Une méthode pour essayer d'obtenir un modèle plus expressif est de considérer les interactions entre variables via des polynômes. À l'ordre 2 et avec $p = 2$, le modèle serait de la forme suivante:

$$f(\boldsymbol{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_1 X_2 + \beta_5 X_2^2$$

En augmentant la flexibilité du modèle, c'est-à-dire en augmentant l'ordre du polynôme, l'interprétabilité des coefficients a tendance à diminuer mais les prédictions peuvent gagner en précision. Il faut toutefois prendre en compte le compromis entre le biais et la variance du modèle. Dans le cas de la régression linéaire, ce problème peut directement être mis en parallèle avec le nombre de paramètres utilisés. En effet, une régression linéaire simple sera très biaisée puisque les prédictions ne dépendront que des $p$ variables disponibles. Une régression polynomiale d'ordre 2 aura quant à elle $\frac{(p+1)(p+2)}{2}$ paramètres (en comptant $\beta_0$), offrant plus de variabilité dans ses prédictions. Le nombre de paramètres augmente ainsi avec l'ordre du polynôme et la variance suivra cette même tendance.

Afin de déterminer si le modèle utilisé est trop biaisé ou présente au contraire trop de variance, il est d'usage de séparer les données en deux voire trois jeux de données distincts et de calculer l'erreur quadratique moyenne (MSE) pour chacun. Cette métrique se définit telle que:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

La MSE permet de s'affranchir de la taille du jeu de données contrairement à la RSS. Il est aussi possible de calculer l'erreur absolue moyenne (MAE) dont la formule est la suivante:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Grâce à l'utilisation de la valeur absolue, la MAE va calculer de manière semblable la distance entre prédiction et valeur réelle pour tous les points. La MSE va quant à elle être plus sensible aux valeurs aberrantes à cause de la fonction carrée.

Les jeux de données sont exploités lors des différentes phases du processus de création du modèle. Le premier est le jeu d'entraînement et correspond à celui utilisé pour calculer $\widehat{\boldsymbol{\beta}}$. C'est donc lors de cette étape d'entraînement que l'on obtient les paramètres de la fonction modélisée. La seconde étape du processus est la phase de test. On se servira ici du jeu de test afin d'observer la précision des prédictions du modèle face à des données qu'il n'a jamais rencontrées auparavant.

La figure 2.2 à droite illustre le compromis biais-variance grâce à trois variations plus ou moins complexes de régressions polynômiale. La courbe noire correspond à la MSE de test en fonction du degré du modèle. Sa forme en U est typique de tous les problèmes de ML et signifie qu'il existe un certain modèle minimisant la MSE de test (point rouge) présentant les prédictions les plus proches de la réalité. Un modèle trop simple (point vert) et donc trop biaisé ne sera pas assez expressif pour obtenir des prédictions proches de la réalité; on parle de sous-apprentissage. Un modèle trop complexe – un spline, régression déclinant ses variables sous de multiples degrés $X_j^m$ ou les transformant linéairement – (point bleu) modélisera presque parfaitement les variations des données d'entraînement mais aura du mal à généraliser à des données qu'il n'a jamais vues. On se retrouve dans un cas de sur-apprentissage. La figure 2.2 à gauche présente ces différents modèles face aux données de test et à la fonction réelle dans le cas d'une seule variable $(p = 1)$.

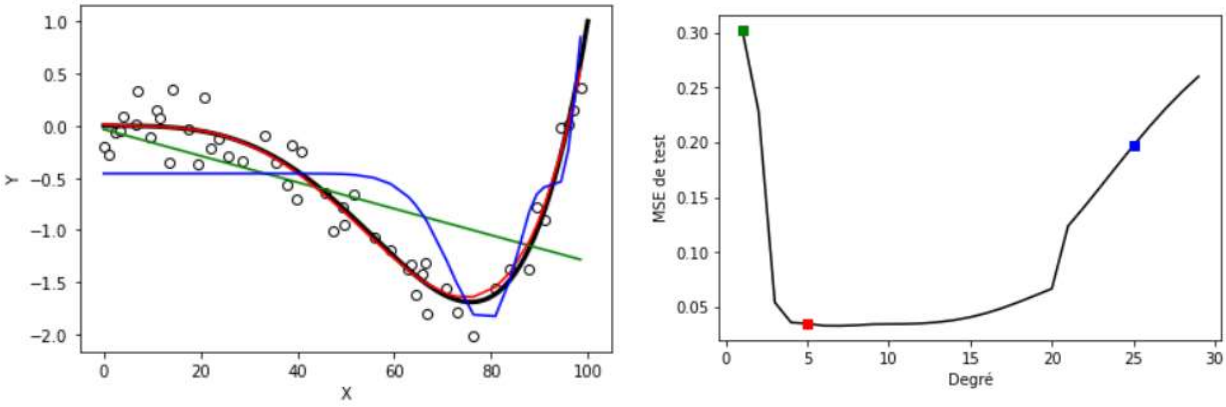


*Figure 2.2: À gauche, données simulées (cercles) de f, courbe en noir. Trois modèles sont présentés: une régression linéaire simple (courbe verte) et deux régressions polynomiales (courbes rouge et bleue). À droite, MSE de test (courbe noire). Les carrés correspondent aux MSE de test des modèles de gauche*

### 2.2.2 Analyse et sélection de modèles

La régression linéaire est un modèle qui permet d'interpréter facilement le poids des variables les unes par rapport aux autres. Ceci n'est cependant possible que lorsque ces variables sont réduites à une échelle similaire. Une étape de normalisation sera donc nécessaire pour parvenir à cet objectif et celle-ci sera détaillée dans le chapitre suivant.

Il existe des méthodes dont le but est de ne garder que les variables les plus importantes dans le modèle. De cette façon, les prédictions du modèle peuvent perdre en précision mais l'interprétation s'en trouve facilitée. La première approche est proposée par Haldar et al. [24] et se nomme la sélection des meilleurs sous-ensembles. Le principe est de trouver le sous-ensemble de $k$ variables qui retourne le meilleur score suivant une certaine métrique. Typiquement, cette métrique peut être l'erreur de prédiction sur le jeu de données test. Cette méthode propose de trouver les sous-ensembles pour chaque $k \in [0, p]$. Initialement, elle était limitée à des problèmes ne dépassant pas une trentaine de variables. Bertsimas et al. [25] ont toutefois montré que cette approche pouvait être formulée comme un problème d'optimisation mixte en nombres entiers, permettant ainsi d'appliquer la sélection des meilleurs sous-ensembles à des problèmes de grandes tailles.

La régression multiple ascendante utilise une approche similaire à celle de la sélection du meilleur sous-ensemble du fait qu'elle va étudier la performance de modèles contenant $k$ des $p$ variables ($k < p$) suivant une certaine métrique. Celle-ci peut une nouvelle fois être l'erreur de prédiction du modèle sur le jeu de test. L'algorithme suivi par cette méthode est itératif. Au départ, le modèle ne contient aucune variable puis l'algorithme va choisir la variable améliorant le plus les performances suivant la métrique retenue. Cette approche fonctionne de manière gloutonne, c'est-à-dire qu'une fois qu'une variable est insérée dans le modèle, elle ne pourra pas être retirée par la suite. Ce point est un des principaux désavantages de la méthode car des variables ajoutées au début du processus peuvent ne plus être significatives au fur et à mesure de l'ajout d'autres variables. Pour contrer ce problème, il existe une approche dite ascendante progressive qui donne la possibilité d'éliminer les variables précédemment introduites. Enfin, la régression multiple descendante propose un fonctionnement semblable à la méthode ascendante si ce n'est que le modèle de départ contient toutes les variables puis que celles-ci sont éliminées petit à petit en fonction de leur faible significativité.

Des méthodes dites de régularisation peuvent aussi être utilisées pour déterminer l'importance des variables. L'une des plus connues est la régularisation L2 ou Ridge développée par Tikhonov [26] puis par Hoerl et Kennard [27] afin de faire face aux situations où les variables explicatives sont fortement corrélées entre elles. Sa formulation est la suivante:

$$\widehat{\boldsymbol{\beta}}^{ridge} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda_{ridge} \sum_{j=1}^{p} \beta_j^2$$

On retrouve dans cette forme la RSS définit précédemment, où $\hat{y}_i$ est la prédiction de la régression linéaire au point $i$, à laquelle on ajoute un terme de pénalité. Le $\lambda$ correspond à un facteur indiquant la force de cette pénalité. Pour $\lambda = 0$, on se retrouve dans le cas d'une régression linéaire simple. Le principe est de pénaliser un coefficient trop grand s'il n'est pas important dans la prédiction. Cette régularisation aura ainsi tendance à créer des modèles avec des coefficients proches de 0.

Une version similaire se basant sur la valeur absolue a été proposée par Tibshirani [28] et se nomme régularisation L1 ou LASSO:

$$\widehat{\boldsymbol{\beta}}^{LASSO} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

$$s.t. |\boldsymbol{\beta}| \leq t$$

où $t$ est un hyperparamètre déterminant le degré de régularisation. Cette régularisation permet de réduire grandement l'importance des variables dans le modèle en fonction de la pénalité appliquée. Elle peut toutefois introduire un faible biais dans le modèle lorsque la réponse dépend trop d'une certaine variable explicative. Hastie et al. [29] a récemment montré que la méthode de sélection du meilleur sous-ensemble obtenait de meilleures performances que la régression multiple ascendante et que la régularisation LASSO dans le cadre d'un grand nombre de variables dans le problème.

Dans ses travaux sur les forêts d'arbres décisionnels, Breiman [30] développe une méthode permettant de déterminer l'importance de chacune des variables du problème pour ce type de modèle. Fisher et al. [31] propose une nouvelle version qui est indépendante du modèle. Le principe de cette approche consiste à entraîner un modèle à partir d'un jeu d'entraînement puis à calculer l'erreur de test sur un nouveau jeu de données dont on a permuté la variable d'intérêt. On peut alors comparer l'erreur de test avec et sans permutation et déterminer si la modification de la variable

produit une grande variabilité dans les résultats ou non. Si c'est le cas, cela signifie que le modèle base en grande partie ses prédictions sur cette variable.

Nous venons de voir des méthodes permettant de connaître l'importance des variables dans des modèles. Nous allons maintenant évoquer quelques techniques utilisées afin de comparer plusieurs modèles entre eux. La première d'entre elle est la validation croisée à $k$ blocs. Considérons un jeu de données d'entraînement que l'on divise en $k$ blocs. On utilise alors $k-1$ blocs afin d'entraîner le modèle puis le bloc restant dit de validation permet de calculer l'erreur de prédiction du modèle avec la MSE par exemple. On répète le processus en changeant le bloc de validation jusqu'à avoir utilisé les $k$ blocs. On peut alors calculer la moyenne et l'écart-type des scores de MSE pour ce modèle. En procédant de la même manière pour d'autres modèles, il est ainsi possible de déterminer celui ayant eu les meilleures performances sur les jeux de validation. Une méthode développée par Akaike [32] et appelée critère d'information d'Akaike (AIC) permet de comparer plusieurs modèles entre eux en tenant compte de leurs performances mais aussi du nombre de variables qu'ils comprennent. Sa formulation est la suivante:

$$\mathrm{AIC} = 2k - 2\ln(L)$$

où $k$ correspond au nombre de variables dans le modèle et $L$ est le maximum de la fonction de vraisemblance du modèle. Cette fonction correspond au produit des probabilités de chacune des observations. Dans le cas d'une régression linéaire, cela est équivalent à prendre la valeur de la MSE [33]. Le meilleur modèle est celui qui à l'AIC le plus faible. Schwarz [34] introduit une métrique très similaire nommée critère d'information bayésien (BIC):

$$\mathrm{BIC} = k\ln(N) - 2\ln(L)$$

où $N$ correspond au nombre d'observations dans l'échantillon. La différence avec l'AIC vient du terme $\ln(N)$ qui va introduire une pénalisation plus forte sur la complexité du modèle. Ainsi, un modèle ayant peu de paramètres sera privilégié à un modèle plus complet même si ses performances sont sensiblement moins bonnes. À l'instar de l'AIC, on choisira le modèle ayant le BIC le plus faible.

### 2.2.3 Sélection de variables et ML en RO

Les solutions obtenues via des méthodes de branchement telles que le SB sont souvent optimales ou très proches de l'optimalité. Toutefois, l'inconvénient majeur de ces approches est le temps de calcul nécessaire à la résolution de chacun des nœuds de l'arbre qui peut s'avérer très élevé sur des problèmes de grandes tailles. Choisir une méthode telle que la fixation de colonne, branchant sur la variable avec la meilleure valeur fractionnaire, permet d'accélérer grandement le processus mais souvent au détriment de la qualité de la solution. De nombreux compromis ont été explorés dans le domaine de la RO mais depuis quelques années, le ML commence à apporter des méthodes aux résultats intéressants. Plus précisément, la méthode la plus utilisée est celle de l'apprentissage par imitation. Dans un contexte de B&B, le principe est de résoudre dans un premier temps l'arbre avec un expert, c'est-à-dire une heuristique telle que le SB. Au fur et à mesure de l'avancement dans l'arbre, des variables décrivant l'état de la résolution ainsi que la solution à chaque nœud sont récoltées. En stockant les variables dans une matrice $\boldsymbol{X}$ et les solutions dans un vecteur $\boldsymbol{y}$, il est alors possible d'exprimer un problème de ML similaire à celui évoqué en section 2.2.1 où l'on recherche une fonction f telle que $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{X}) + \boldsymbol{\epsilon}$.

Alvarez et al. [35] propose ainsi de modéliser les données par le biais d'une simple régression linéaire. Ils définissent une liste des variables dont le but est de transmettre un maximum d'informations lors de la phase d'apprentissage. D'après les auteurs, les variables utilisées doivent répondre à plusieurs propriétés:

- Le **nombre de variables** doit être **indépendant** de la taille du problème pour éviter d'apprendre de nouvelles stratégies de branchement à chaque nouveau problème rencontré.
- Les **variables** doivent être **invariantes** lors de faibles modifications telles que des permutations de lignes ou de colonnes dans le problème.
- Les **variables** doivent être **indépendantes** de l'échelle du problème.

De plus, les auteurs déterminent trois catégories pour définir les différents types de variables et leurs propriétés:

- Les variables statiques du problème sont calculées une seule fois au départ de la résolution et ne se basent que sur les paramètres $\boldsymbol{A}$, $\boldsymbol{b}$ et $\boldsymbol{c}$ du problème.
- Les variables dynamiques du problème sont reliées à la solution du problème au nœud courant.
- Les variables dynamiques d'optimisation correspondent aux effets du branchement de la variable étudiée dans l'optimisation globale du problème.

Une version « *online* » sera ensuite présentée par Alvarez et al. [36]: au lieu de se baser sur des données générées avant l'apprentissage, le modèle va utiliser un mécanisme reposant sur la fiabilité de ses prédictions. Lorsque la prédiction sur un nœud de l'arbre est considérée comme fiable, le modèle va pouvoir déterminer directement sa valeur à partir des variables présentées dans [35].Dans le cas contraire, une heuristique de SB est utilisée afin de résoudre le nœud et de calculer son score. Les variables qui sont associées à ce nœud ainsi que son score sont stockés afin de servir de jeu de données pour l'entraînement du modèle et de ce fait, améliorer la fiabilité de ses prédictions.

Un modèle un peu plus complexe est proposé par Khalil et al. [37] sous la forme d'une machine à vecteurs de support (SVM). Ici, l'apprentissage du modèle est spécifique à l'instance et les données sont collectées sur les 500 premiers nœuds de l'arbre. Ces derniers sont résolus en utilisant une heuristique de SB afin d'obtenir un score. Les auteurs se placent dans un problème de classement où les labels des candidats sont binaires. Chaque candidat se voit attribuer son label en fonction de sa proximité avec le meilleur score d'après le SB. S'il se situe à moins de 20% de la solution du candidat optimal, il est alors considéré comme étant un choix valide. Les variables utilisées pour l'apprentissage font intervenir certaines des variables dynamiques de [35] tout en en introduisant de nouvelles. Au total, 72 variables atomiques sont utilisées pour décrire le problème de façon statique ou dynamique et des variables d'interactions, produit de deux variables atomiques, sont aussi calculées. L'ensemble des variables forme un noyau polynomial de degré 2 dans l'espace des variables atomiques. Yang et al. [38] introduit une méthode de classement des candidats similaire

à celle de [37] où plusieurs d'entre eux peuvent être considérés comme valides. Le but est alors de créer un modèle de ML basé sur une heuristique de SB afin de permettre le branchement sur plusieurs variables en même temps. Le modèle utilisé a une architecture XGBoost et les résultats obtenus sont meilleurs, aussi bien en nombre de nœuds explorés qu'en temps de calcul. Les auteurs ont repris certaines des variables présentées par [35] et en introduisent de nouvelles associées à l'état de l'arbre et à sa résolution. Ils montrent de plus que parmi ces variables, de nombreuses semblent de trop faible importance pour être réellement utiles au modèle.

En 2020, Zarpellon et al. [39] ont développé une architecture composée de deux réseaux de neurones profonds complémentaires. Le premier consiste à traiter les candidats potentiels pour la phase de sélection de variables en se basant sur leurs caractéristiques intrinsèques. Ces caractéristiques sont au nombre de 25. Au fur et à mesure de ce traitement, les informations concernant l'arbre de B&B en lui-même sont ajoutées via un second réseau de neurones à travers 61 variables descriptives afin de moduler les résultats du premier réseau. In fine, une distribution de probabilité est obtenue pour chacun des candidats afin de choisir celui qui a le meilleur potentiel. Les expériences montrent que le nombre de nœuds explorés par ce modèle est bien inférieur à ce que propose le state-of-the-art. Les auteurs démontrent ainsi l'importance du choix des variables définissant le problème, celles associées uniquement aux candidats ne suffisant pas.

La plupart des travaux faisant intervenir du ML sur des arbres de B&B ne sont pas appliqués à un contexte de *diving*. Récemment, Yilmaz et al. [40] ont toutefois proposé d'utiliser l'apprentissage par imitation afin de développer une heuristique de *diving*. Le modèle présenté est un simple perceptron mutlicouche qui utilise trois types de variables différents en entrée: les variables associées à la colonne (e.g. son coût réduit), celles propres au nœud puis des variables globales détaillant la situation de l'arbre (e.g. sa profondeur). Ces variables ont déjà été exploitées par les réseaux de neurones en graphes de Gasse et al. [41] qui obtenaient des résultats relativement meilleurs que les autres modèles de l'époque sur des problèmes d'optimisation linéaire mixte en nombres entiers. Les résultats obtenus dans [40] ne se sont pas révélés aussi bons que le state-of-the-art mais sont toutefois encourageants. Bien que les variables utilisées pour décrire ce problème n'étaient pas initialement appliquées à du *diving*, les auteurs montrent qu'elles peuvent tout de même être adaptées pour ce contexte dans une moindre mesure.

# CHAPITRE 3 CONTEXTE DE LA RECHERCHE

Dans cette partie, nous allons expliquer le contexte d'exécution – c'est-à-dire le logiciel utilisé, l'origine des données d'entraînement ainsi que les heuristiques de comparaison – dans lequel se place ce mémoire. Nous présenterons ensuite les variables qui seront utilisées dans le cadre de nos expériences.

## 3.1 Quelques mots sur l'environnement de travail

### 3.1.1 Définition des heuristiques utilisées

Dans notre étude, nous allons considérer 2 heuristiques de *diving* basées sur une énumération B&B. Elles nous serviront de points de comparaison lors de nos expériences:

- Fixation d'une colonne (*pure diving*): Le candidat retenu est celui ayant la valeur fractionnaire la plus élevée.
- Branchement fort (*strong diving*): On simule le branchement de chaque candidat l'un après l'autre en procédant à chaque fois à la résolution de ces nouveaux nœuds. Un score est obtenu à chaque fois et le plus élevé correspond au candidat retenu. Par la suite, nous ne ferons pas de distinction d'appellation entre le SB et le *strong diving*.

### 3.1.2 GENCOL et obtention des données d'entraînement

Le logiciel utilisé dans le cadre de cette recherche est GENCOL, développé par les chercheurs du GERAD, capable de réaliser de la génération de colonnes et de résoudre des problèmes de plus courts chemins. À partir d'un ensemble de vols et des contraintes associées aux bases, GENCOL retourne une solution faisable dont l'écart avec la solution optimale dépend entre autres de l'heuristique de branchement utilisée.

Notre problème est décrit à l'échelle d'un mois mais nous choisissons de le découper en fenêtres de temps [17] afin d'en simplifier la résolution. Le CPP mensuel est ainsi découpé en 8

fenêtres de temps avec superposition de chacune des fenêtres. Les dernières rotations de la fenêtre $f$ pourront alors servir à l'initialisation de la fenêtre $f + 1$ pour permettre une meilleure continuité dans la solution.

Afin d'obtenir des données utiles pour l'entraînement de nos modèles, des instances de tailles différentes ont été résolues grâce à l'heuristique de SB sous GENCOL. Pour chacun des nœuds associés aux rotations candidates, il a alors été possible d'obtenir la valeur de plusieurs variables détaillées en section 3.2 ainsi que la solution associée. L'utilisation du SB permet d'obtenir des solutions proches de l'optimalité. En outre, puisque cette heuristique n'est exploitée que pour la génération des données d'entraînement, il ne sera plus nécessaire de l'utiliser par la suite. Cela veut dire que la résolution des instances lors des phases de test ne dépendra que du temps d'inférence du modèle de ML.

### 3.1.3 Implémentation du B&P dans notre problème

Une illustration du B&P appliqué à notre problème ainsi qu'une explication du processus sont présentées ci-après:

1. À partir du nœud courant (nœud n), le RMP associé est résolu avec la méthode du simplexe. Il ne contient qu'un sous-ensemble des colonnes du problème maître.
2. À partir des solutions duales obtenues avec la résolution du RMP, des colonnes à coûts réduits négatifs sont recherchées telles que:

$$\bar{c} = \min_{r \in R}\{c_r - \pi^T A_r\}$$

   Cette étape consiste à résoudre un problème dit de *pricing*. Les colonnes à coûts réduits négatifs sont identifiées en résolvant un problème de plus court chemin avec contraintes de ressources où chaque chemin correspond à une rotation.
3. Les rotations obtenues sont ajoutées au RMP qui est de nouveau optimisé et le problème de *pricing* de nouveau résolu afin de trouver d'autres colonnes à coûts réduits négatifs. Les étapes 1 et 2 sont donc exécutées itérativement jusqu'à rencontre d'un critère d'arrêt ou jusqu'à optimalité, c'est-à-dire qu'aucune colonne à coût réduit négatif n'a été trouvée.

4. De nombreux candidats au nœud n+1 sont alors obtenus. Un premier tri est fait pour ne garder que les variables présentant le meilleur potentiel, en se basant sur leur valeur fractionnaire. Le nombre de variables gardées $n_c$ est un paramètre choisi lors de la résolution. Lorsque le B&P est appliqué à notre problème, ce tri ne retient que les candidats dont la valeur fractionnaire est comprise entre 0.5 et 1, les meilleurs étant proches de 1. Il est cependant possible qu'il y ait moins de $n_c$ rotations dont la valeur fractionnaire est comprise dans cet intervalle, auquel cas le nombre de candidats s'en trouvera réduit. Dans la situation où aucune rotation n'a une valeur fractionnaire supérieure à 0.5, il y aura uniquement un candidat correspondant à la rotation avec la valeur fractionnaire la plus élevée. Cette dernière situation a tendance à arriver vers la fin de l'arbre.
5. Une heuristique est enfin utilisée afin de déterminer le candidat offrant le plus de potentiel pour la suite de l'exploration de l'arbre. Dans notre cas, l'heuristique est celle du SB qui choisit le candidat augmentant le moins la borne inférieure.

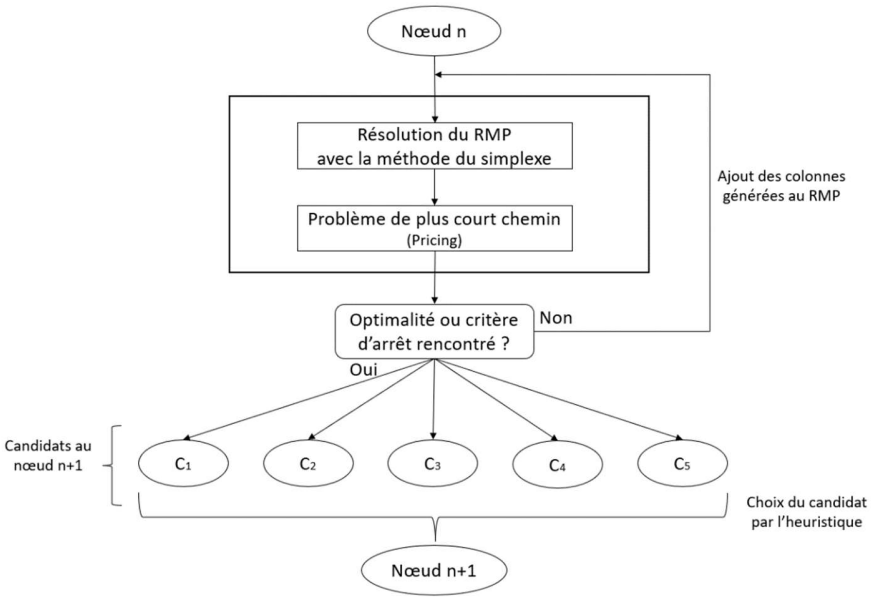


*Figure 3.1: Processus de résolution d'un arbre de B&P*

Dans la suite des travaux, nous fixerons $n_c = 5$. Cela nous permet de travailler avec un nombre suffisant de candidats pour l'apprentissage sans pour autant perdre trop de temps lors de l'obtention des données lors de l'utilisation du SB.

## 3.2 Présentation des variables du problème

Nous allons dans un premier temps définir les variables associées à chaque nœud de l'arbre de B&P. Celles-ci peuvent être séparées en deux groupes en s'appuyant sur le fait qu'elles sont le résultat d'un algorithme de programmation linéaire et d'un algorithme de *pricing*. Nous aurons alors des variables correspondant aux propriétés des colonnes du problème d'optimisation et d'autres découlant de ses lignes et ce, pour un total de 12 variables explicatives.

### 3.2.1 Variables associées aux colonnes

Dans le cadre du CPP, les colonnes d'un problème d'optimisation correspondent aux rotations d'équipages et sont plus généralement associées aux variables du problème. Il ne faut pas confondre les variables d'un problème d'optimisation c'est-à-dire les rotations d'équipages que nous cherchons à construire avec les variables que nous définissons actuellement. Ces-dernières sont celles dont nous voulons connaître l'importance et que nous utiliserons lors de l'entraînement de nos modèles de ML. En considérant que la rotation fixée correspond à la rotation qui est fixée dans la décision de branchement **i.e. le candidat**, nous travaillerons avec les variables suivantes:

- Var cost: coût de la rotation fixée.
- Frac val: valeur fractionnaire de la rotation fixée dans la solution de la relaxation linéaire au nœud parent.
- Fraction conflicting columns (FCC): fraction des rotations en conflit avec la rotation fixée dans la solution de la relaxation linéaire au nœud parent. Une rotation est en conflit lorsqu'elle a au moins un vol en commun avec la rotation fixée.
- Fraction conflicting columns with positive value (FCCPV): fraction des rotations en conflit avec la rotation fixée et de valeur non-nulle dans la solution de la relaxation linéaire au nœud parent.

- Min cost conflicting column (M3C): coût minimum d'une rotation en conflit avec la rotation fixée.
- Min cost conflicting column with positive value (M3CPV): coût minimum d'une rotation en conflit avec la rotation fixée et de valeur non-nulle dans la solution de la relaxation linéaire au nœud parent.
- Nb cols in MP: nombre de rotations dans le problème maître au nœud parent. La valeur de cette variable est constante parmi les candidats d'un même étage.

### 3.2.2 Variables associées aux lignes

À l'instar des colonnes, les lignes d'un problème d'optimisation occupent un rôle central puisqu'elles représentent les contraintes imposées lors de la résolution. Du point de vue de notre problème, chaque contrainte va ainsi correspondre à un vol. Parmi l'ensemble de nos variables, les suivantes caractérisent les lignes:

- Dual cost min: valeur minimum d'une variable duale parmi les contraintes de couverture de vol à laquelle contribue la variable.
- Dual cost max: valeur maximum d'une variable duale parmi les contraintes de couverture de vol à laquelle contribue la variable.
- Dual cost avg: valeur moyenne d'une variable duale parmi les contraintes de couverture de vol à laquelle contribue la variable.
- Frac pairing tasks fixed (FPTF): fraction des vols fixés dans le nœud parent (par les décisions précédentes). La valeur de cette variable est constante parmi les candidats d'un même étage.
- Nb pairing tasks: nombre de contraintes de couverture de vols à laquelle contribue la rotation, c'est-à-dire le nombre de vols dans la rotation.

### 3.2.3 Variable cible

À nos variables définies précédemment s'ajoute la '*value*' ou valeur objectif. Concrètement, cette valeur représente le score attribué par l'heuristique de branchement fort lors de l'évaluation de chaque nœud. Dans notre cas, ce score représente la valeur de la solution, c'est-à-dire le coût réel de l'ensemble des pairings optimaux retenus pour résoudre le CPP. Pour rappel, l'objectif du CPP est de diminuer cette valeur au maximum pour obtenir une solution proche de l'optimalité.

Nous allons donc définir cette variable comme étant notre cible dans la suite de nos expériences. En d'autres termes, nous chercherons à voir l'importance de chacune de nos variables par rapport à cette cible. Du point de vue de la régression, la '*value*' correspond au $y$ dans la formule $y = f(X) + \epsilon$ de la section 2.2.3.

# CHAPITRE 4 SÉLECTION DE VARIABLES ET CHOIX DU MODÈLE

Cette maîtrise propose de s'attaquer au problème de construction des rotations d'équipages en plusieurs étapes. Dans un premier temps, nous allons réaliser un pré-traitement des données afin de pouvoir les exploiter de façon adéquate. Nous développerons ensuite les méthodes permettant de déterminer les variables les plus importantes pour la prise de décision lors du branchement dans l'arbre de B&P. Nous évoquerons en même temps plusieurs modèles de ML que nous testerons dans l'optique d'obtenir une nouvelle heuristique capable de réaliser de bonnes décisions en un temps réduit. Nous aborderons enfin les différentes métriques définissant les performances de nos modèles ainsi que les méthodes pour déterminer les plus aptes à fournir les meilleurs résultats sous GENCOL.

## 4.1 Pré-traitement des données

L'objectif principal de notre étude est de déterminer l'impact qu'a chacune des variables à notre disposition lorsque l'on doit choisir la rotation sur laquelle brancher pendant la résolution du B&P. En effet, être en possession de cette information peut par exemple permettre d'éviter le calcul de certaines variables lors de l'exploration de l'arbre et ainsi réduire le temps d'exécution global. Cette première étape de sélection fait l'objet de différents tests mais se voit tout d'abord précéder d'une phase de normalisation.

### 4.1.1 Normalisation des données

La normalisation est une étape primordiale dans un processus impliquant de l'apprentissage machine. Son rôle est de rassembler les données dans un intervalle réduit, typiquement entre 0 et 1. Cela permet d'avoir des intervalles à des échelles similaires pour chacune des variables. Dans le cas d'une régression linéaire, les coefficients seront alors du même ordre de grandeur ce qui facilitera l'analyse de l'importance des variables comme nous le verrons par la suite.

Un premier coup d'œil à la distribution de chacune de nos variables nous confirme que nos données ne sont pas à la même échelle (figure 4.1). De plus, dans l'optique de l'entraînement de nos

modèles, il est préférable de se ramener à des distributions plus standards afin de faciliter l'apprentissage. Une répartition homogène sur l'intervalle étudié permettra au modèle de mieux distinguer de faibles variations entre les données et donc de gagner en précision.

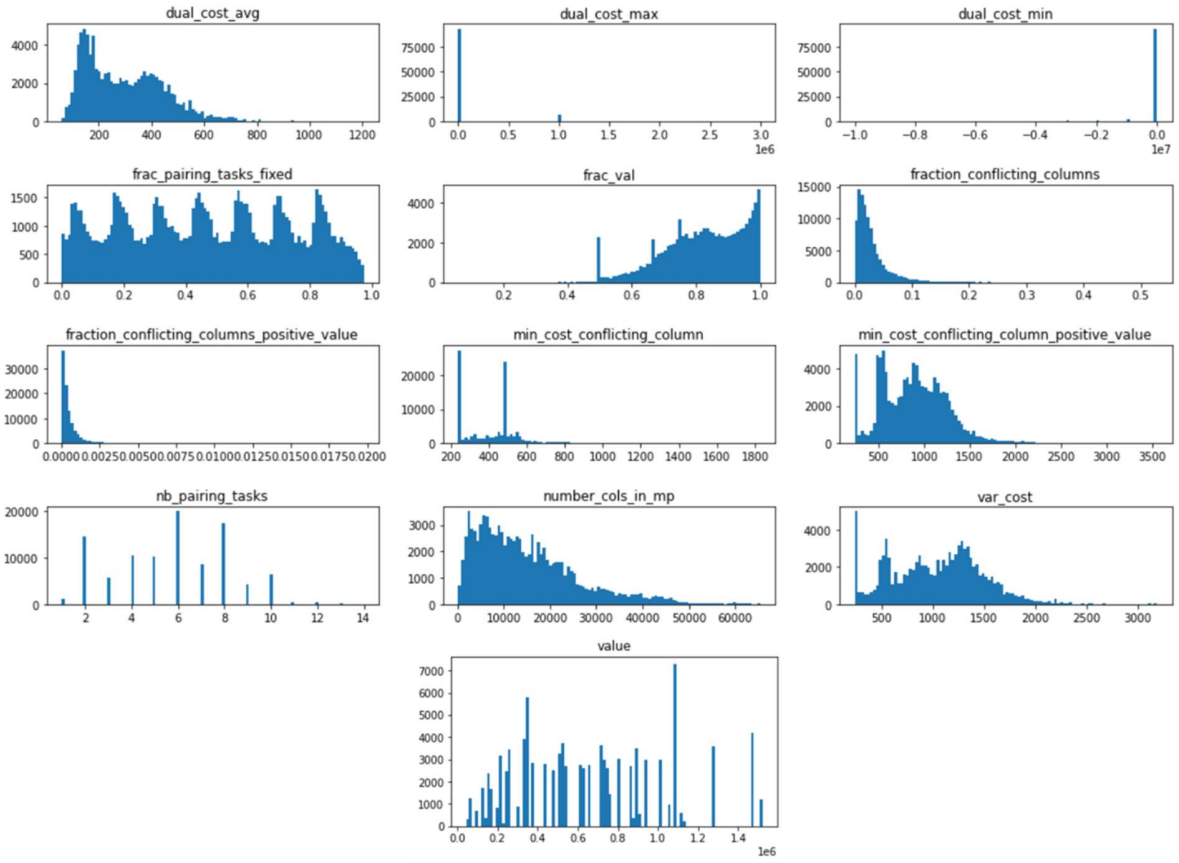


*Figure 4.1: Distribution des données non normalisées*

Puisque dans la pratique la résolution du problème se fait de façon séquentielle, nous serons dans l'incapacité de normaliser toutes les données d'un même arbre d'une seule traite. En effet, il est nécessaire de faire le branchement à l'étage $n$ et donc d'y normaliser les données au préalable afin d'avoir accès aux données de l'étage $n + 1$ pour les normaliser à leur tour.

Nous procèderons donc à une normalisation à la volée, au fur et à mesure que les données se présentent. Étant donné que nous souhaitons comparer un nœud à ses frères du même étage et non pas à l'ensemble des nœuds de l'arbre, cette méthode semble être la mieux adaptée à nos objectifs.

Afin d'avoir une répartition plus homogène des données, nous passons tout d'abord certaines variables au logarithme. Ceci est par exemple le cas pour FCC dont les valeurs sont essentiellement concentrées sur la partie gauche du spectre. Dans un second temps, nous pouvons appliquer les normalisations à proprement parler. La liste des transformations appliquées à chacune des variables est renseignée en annexe B. Nous allons en distinguer deux différentes:

- La standardisation où les variables sont centrées et réduites:

$$x_{i,j}^{stand} = \frac{x_{i,j} - \bar{x}_j}{\sigma_j}$$

où $\bar{x}_j$ et $\sigma_j$ sont respectivement la moyenne et l'écart-type de la variable $j$ à l'étage considéré.

- La MinMax:

$$x_{i,j}^{minmax} = \frac{x_{i,j} - \min(j)}{\max(j) - \min(j)}$$

où $\min(j)$ et $\max(j)$ correspondent aux minimum et maximum globaux de la variable $j$. Dans notre cas, cette normalisation est utilisée lorsque la variable étudiée a des valeurs constantes à chaque étage. Utiliser la standardisation produirait une forme indéterminée, c'est donc pourquoi il faut procéder à une normalisation globale. L'inconvénient majeur est le besoin d'une borne supérieure mais il est possible d'en approximer une à partir de l'ensemble de nos données d'entraînement. Étant donné que les variables concernées retournent des valeurs supérieures à 1, on peut définir une borne inférieure à 0.

Une fois les normalisations adéquates réalisées, nous choisissons d'appliquer la fonction softmax à la valeur objectif telle que:

$$\text{softmax}\left(x_{i,\,value}^{stand}\right) = \frac{\exp\left(x_{i,\,value}^{stand}\right)}{\sum_{j=1}^{n_c} \exp\left(x_{j,\,value}^{stand}\right)}$$

avec $n_c$ le nombre de candidats de l'étage. De cette façon, nous pouvons obtenir une distribution de probabilité à la place d'un rang discret. Cela revient à dire qu'une rotation a plus de chances de mener à une meilleure solution plutôt que dire qu'elle est nécessairement meilleure que les autres.

Un avantage à réaliser notre normalisation à la volée, étage par étage, est que l'on peut s'affranchir de la taille du problème. En effet, nous ne pourrions, par exemple, pas utiliser la valeur objectif telle quelle pour l'entraînement de nos modèles car un arbre de petite taille peut avoir des valeurs autour de $10^5$ tandis qu'un autre de plus grande taille serait aux alentours de $10^6$. Utiliser des données avec cette différence d'ordre de grandeur ferait que nos modèles linéaires prédiraient des solutions correctes pour les arbres de grandes tailles mais ne pourraient pas distinguer les subtilités

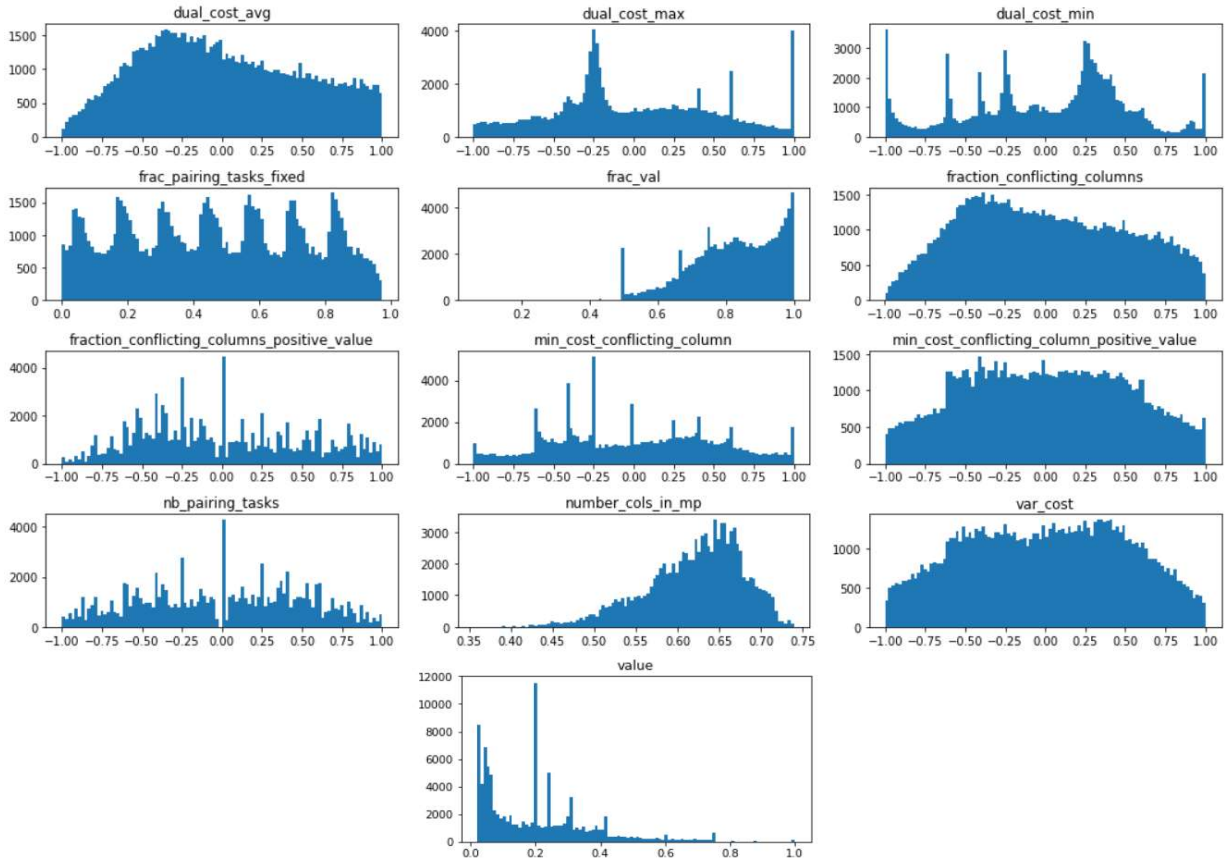


*Figure 4.2: Distribution des données normalisées*

des plus petits. La figure 4.2 représente les données normalisées.

Il est intéressant de voir que la normalisation de la valeur objectif présente un pic aux alentours de 0.2. En effet, nous choisissons de retenir $n_c = 5$ candidats à chaque nœud. Cela vient donc du fait que les étages sont principalement composés de cinq candidats et lorsque ceux-ci sont similaires, la probabilité de choisir le meilleur est de $1/5 = 0.2$. La forme exponentielle décroissante de cette même distribution s'explique par l'utilisation de la fonction softmax: la somme des probabilités

d'un étage vaut nécessairement 1. Lorsqu'un candidat obtient un bon score $s_{meilleur}$ tel que $s_{meilleur} \gg {}^{1}/_{n_c}$, les candidats restants auront un score moyen $\bar{s} = \frac{1-s_{meilleur}}{n_c-1} \ll {}^{1}/_{n_c}$. Ceci explique donc la forte densité de probabilité dans l'intervalle [0,0.2] et la densité plus faible dans l'intervalle [0.2, 1].

### 4.1.2 Normalisation de la valeur fractionnaire

Au vu des bons résultats obtenus avec l'heuristique de *pure diving*, la valeur fractionnaire semble être une variable assez prometteuse pour la sélection de la rotation de branchement. Nous allons donc créer une nouvelle variable, 'Norm frac val', avec une normalisation similaire avec celle de la valeur objectif :

$$x_{i,norm\_frac\_val} = \text{softmax}\left(x^{stand}_{i,\,frac\_val}\right) = \frac{\exp\left(x^{stand}_{i,\,frac\_val}\right)}{\sum_{j=1}^{n_c} \exp\left(x^{stand}_{j,\,frac\_val}\right)}$$

La figure 4.3 présente la superposition des distributions de 'Norm frac val' et de 'value'. Les similarités entre ces deux distributions semblent en effet démontrer une possible corrélation entre valeur fractionnaire et choix de la rotation de branchement.

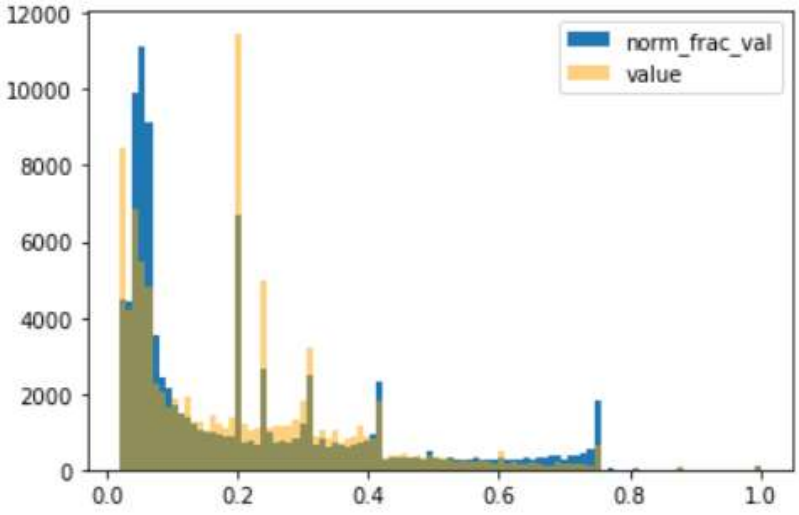


*Figure 4.3: Comparaison des distributions de Norm frac val et de la réponse*

## 4.2 Importance des variables dans la prise de décision

Afin de déterminer l'importance de nos variables vis-à-vis de la solution, il existe plusieurs méthodes que nous allons maintenant détailler.

### 4.2.1 Matrice de corrélation

Un premier moyen permettant de déterminer les liens entre les différentes variables est d'observer leurs coefficients de corrélation respectifs. Cette valeur se calcule telle que:

$$r_{i,j}^{pearson} = \frac{E(X_i\, X_j) - E(X_i)E(X_j)}{\sigma_{X_i}\sigma_{X_j}}$$

où $\mathrm{E}(X)$ correspond à l'espérance mathématique de $X$. Un bon estimateur non biaisé de cette valeur est la moyenne de $X$. Un coefficient de corrélation est compris dans l'intervalle $[-1,\ 1]$. Une valeur se situant vers l'un de ces extrêmes signifiant que les deux variables sont fortement, et inversement dans le cas négatif, corrélées. Dans cette situation, cela signifie qu'une variable explique l'autre relativement bien. Si le coefficient est trop élevé en valeur absolue, il est parfois nécessaire de retirer du problème l'une des variables concernées afin de faciliter l'interprétation du modèle. L'équation précédente correspond au coefficient de corrélation de Pearson est permet de déterminer des relations linéaires entre les variables. Il existe une version alternative appelée coefficient de Spearman dont le but est d'identifier des relations monotones. Une relation est dite monotone entre deux variables lorsqu'il existe une fonction $f$ les associant telle que pour tout couple $(x, y)$ de la première variable avec $x \leq y$, on a soit $f(x) \leq f(y)$ (monotonie croissante) sur tout l'intervalle, soit $f(x) \geq f(y)$ (monotonie décroissante) avec $(f(x), f(y))$ couple de la seconde variable. La fonction $f$ est donc soit uniquement croissante (ou constante) sur tout l'intervalle, soit uniquement décroissante (ou constante). Cette hypothèse moins restrictive peut montrer l'intérêt de modèles simples non linéaires dans la partie prédiction du problème. Le coefficient de Spearman s'exprime de la façon suivante:

$$r_{i,j}^{spearman} = \frac{Cov\left(Rg(X_i), Rg(X_j)\right)}{\sigma_{Rg(X_i)}\sigma_{Rg(X_j)}}$$

où $Rg(X)$ correspond au rang de la variable $X$ et $\sigma_{Rg(X)}$ à son écart-type. Considérons la variable $X_i = (5, -4, 3)$. Obtenir le rang de $X$ correspond à ordonner les valeurs en ordre croissant et à leur attribuer un rang tel que $Rg(X) = (3, 1, 2)$

### 4.2.2 Analyse des coefficients de régression

Une méthode relativement simple pour déterminer l'importance des variables explicatives vis-à-vis de la réponse est de modéliser le problème grâce à une régression linéaire. Lorsque toutes les variables sont mises à la même échelle via l'étape de normalisation, il est alors possible de comparer la contribution de chacune d'entre elles lors de la prédiction. Nous noterons IM$P_i$ l'importance de la variable i dans la régression linéaire telle que:

$$IMP_i = 100 * \frac{|\beta_i|}{\sum_{j=1}^{p} |\beta_j|}$$

Pour la suite de notre étude, nous considérerons plusieurs modèles aussi bien pour leur interprétation que pour leur utilisation sous GENCOL en remplacement de l'heuristique de SB. Nous pouvons dès lors distinguer trois groupes dont les objectifs diffèrent. Le premier groupe tend à montrer l'importance de la valeur fractionnaire dans la prise de décision de la rotation de branchement. Il veut aussi montrer l'intérêt de la normalisation que nous avons choisie pour cette variable. Nous évaluerons ainsi les performances théoriques des quatre modèles suivants:

- Modèle 1: Les variables **initialement présentes** sont utilisées. Norm frac val n'en fait donc pas partie.
- Modèle 2: Les variables **initialement présentes** sont utilisées **sauf Frac val**.
- Modèle 3: **Toutes les variables** sont présentes, y compris Norm frac val, **sauf Frac val**.
- Modèle 4: **Toutes les variables** sont présentes.

Le deuxième groupe que nous allons explorer nous permettra de voir si la limitation de nos modèles grâce à de la régularisation a un effet positif sur les performances:

- Modèle 5: **Toutes les variables** sont présentes et utilisation de la **régularisation L1**.
- Modèle 6: **Toutes les variables** sont présentes et utilisation de la **régularisation L2**.

Le dernier groupe a pour but d'observer l'effet d'interactions entre variables. Des relations non-linéaires entre les variables pourraient en effet apporter plus d'expressivité au modèle et ainsi mener à des prédictions plus précises:

- Modèle 7: Polynôme de **degré 2** contenant **toutes les variables**.
- Modèle 8: Polynôme de **degré 2** limité aux **variables les plus importantes** d'après les précédents modèles.

### 4.2.3 Arbres de décision

Avec les régressions linéaires, les arbres de décisions sont les modèles de ML permettant une bonne interprétation de l'importance des variables. Ils sont de plus capables de modéliser des interactions plus complexes entre les variables. Le principe est relativement simple et fonctionne de façon itérative. Dans le cadre d'un problème de régression comme le nôtre, l'algorithme choisit tout d'abord la variable qui va réduire au maximum la MSE ou la MAE du modèle. Il va ensuite créer deux branches afin de distinguer deux cas à partir de la valeur de cette variable: dans l'observation considérée, cette valeur est-elle supérieure ou inférieure à un certain seuil? En fonction du cas, un chemin différent est emprunté dans la suite de l'arbre. Le même procédé va être utilisé dans chacune des branches jusqu'à atteindre une certaine profondeur définie en hyperparamètre.

Les forêts aléatoires (ou random forests RF) [30] présentent les mêmes avantages que les arbres de décisions tout en réduisant la complexité des modèles produits pour éviter des problèmes de sur-apprentissage. Au lieu de ne considérer qu'un seul arbre, les RF en utilisent de nombreux entraînés sur des jeux de données différents, tous issus du jeu de données originel. Leurs profondeurs sont généralement assez faibles pour ne pas les rendre trop complexes. Une fois que tous les arbres ont été entraînés, une prédiction est faite en prenant la moyenne des prédictions de chacun des arbres.

### 4.2.4 Sélection itérative

Les méthodes de sélection itérative des variables peuvent se distinguer en deux approches au fonctionnement similaire. Leur but commun est d'ajuster successivement des modèles contenant un nombre défini de variables en choisissant celles qui amélioreront le plus un certain critère.

- La sélection séquentielle des variables (SSV) utilise un *scorer* qui sera dans notre cas une fonction de perte, telle que la MSE ou la MAE, que l'on voudra minimiser. Cette méthode peut partir d'un modèle à une variable et en ajouter au fur et à mesure ou au contraire évaluer initialement un modèle contenant toutes les variables et les éliminer petit à petit. La variable ajoutée sera alors celle minimisant le plus la fonction de perte tandis que celle retirée sera celle qui contribue le moins à la réduction de cette même fonction.
- L'élimination récursive des variables (ERV) va se baser uniquement sur l'importance des coefficients du modèle. En partant d'un modèle contenant toutes les variables du problème, l'algorithme va éliminer successivement celles dont l'importance est la plus faible.

Dans les deux cas, une fois qu'une variable est ajoutée ou retirée du modèle, il n'est plus possible d'interagir avec elle ensuite. Puisque l'élimination récursive des variables se base sur leur importance, nous devrons utiliser des modèles disposant de ces caractéristiques. Nous évaluerons donc les deux méthodes grâce à des régressions linéaires simples et des arbres de régressions.

### 4.2.5 Permutation de variables

La permutation de variables est une technique comparant la variation de l'erreur de prédiction lorsqu'une variable est permutée. Si l'erreur augmente beaucoup lorsque les valeurs d'une variable sont mélangées, cela signifie que cette dernière a une grande importance dans la prédiction de la réponse. Plus précisément, l'importance d'une variable se définit par la différence entre l'erreur prédite après permutation et celle avant permutation. Nous considérerons la MSE comme fonction de perte pour calculer l'erreur de prédiction.

## 4.3 Sélection du modèle

Dans l'optique d'obtenir de bonnes performances lors de l'utilisation d'une nouvelle heuristique sous GENCOL, il est nécessaire d'utiliser des méthodes de comparaison de modèles adéquates. Nous avons déjà défini la MAE et la MSE qui seront nos deux premiers indicateurs de performances. En nous basant sur [37], nous définissons une métrique que nous appellerons précision approximative ou PA. Nous partons du principe qu'après sélection suivant leur valeur fractionnaire (étape 4 du processus de B&P, section 2.1.1), plusieurs rotations peuvent être considérées comme acceptables à chaque étage lorsque les conditions sont réunies. En effet, si tous les candidats ont obtenu un score similaire d'après le SB, il est logique de ne pas en valoriser un plus que les autres. Soit $SB^{acc}$ la liste des candidats acceptables d'après le SB telle que:

$$SB^{acc} = \left\{c \,\middle|\, SB_c \geq (1-\alpha) * SB_{opt}\right\}$$

où $SB_{opt}$ est le score du meilleur candidat d'après le SB et $\alpha$ un hyperparamètre définissant la tolérance pour qu'une solution soit acceptable. Nous utiliserons $\alpha = 0.1$ pour la suite de nos expériences. La précision approximative pour l'étage $i$ se calcule alors de la façon suivante:

$$PA^i = \begin{cases} 1, & si\ ML_i\ dans\ SB^{acc} \\ 0, & sinon \end{cases}$$

où $ML_i$ correspond au candidat retenu par le modèle de ML (score le plus élevé) à l'étage $i$.

Deux autres indicateurs de performance que nous allons utiliser sont l'AIC et le BIC décrits dans la section 2.2.2.

Pour avoir un point de comparaison entre les performances de chacun de nos modèles, nous devons définir des performances de référence ou *baseline*. Celle-ci sera définie comme étant les performances d'un modèle qui choisit aléatoirement un candidat à chaque étage.

Afin de pouvoir comparer les modèles entre eux, il est aussi nécessaire de définir la performance globale d'un modèle. Pour ce faire, nous utiliserons la technique de la validation croisée sur nos $k$ instances. Le modèle est entraîné sur $k-1$ instances et est testé sur la k$^{ème}$ pour obtenir les métriques de test définies plus haut. Ce processus est réalisé successivement pour que chaque instance soit utilisée une fois comme jeu de test. On associe ensuite la moyenne de ses métriques comme étant les performances globales du modèle.

Cette phase de comparaison est réalisée dans un premier temps pour obtenir les performances théoriques de chacun des modèles et déterminer le meilleur dans chacun des groupes définis en section 4.2.2. Une fois la sélection faite, nous pourrons ensuite tester ces modèles en conditions réelles sous GENCOL.

# CHAPITRE 5 RÉSULTATS ET DISCUSSIONS

Nous allons maintenant présenter les résultats obtenus concernant l'analyse d'importance de variables ainsi que les performances théoriques et pratiques de nos modèles. Nous distinguerons les méthodes que nous utiliserons uniquement pour l'analyse, les régressions linéaires et les performances sous GENCOL.

Les données utilisées dans le cadre de nos expériences sont dérivées de celles décrites par Kasirzadeh et al. [42] et proviennent d'une compagnie aérienne américaine. Elles sont regroupées en 7 instances contenant entre 1000 et 7800 vols répartis sur 3 bases à chaque fois. Nous pourrons considérer que les 3 premières instances ont des tailles relativement faibles tandis que les 4 dernières contiennent un plus grand nombre de vols. De plus, l'ensemble de ces vols sont réalisés sur une échelle de temps mensuelle. Comme décrit dans la section 3.2.2, le CPP est découpé en 8 fenêtres de temps et est résolu grâce à l'heuristique de SB. Une particularité de notre méthode d'exécution pour le SB et les modèles de ML provient de l'initialisation de l'arbre de B&P. En effet, le nœud racine nécessite un sous-ensemble de rotations produisant une solution faisable pour que la résolution de l'arbre puisse se faire. La génération de ces rotations peut être sujette à quelques variations et peut donc mener à emprunter des chemins différents lors de la résolution de l'arbre. Ainsi, chaque instance a été résolue 3 fois avec le SB pour obtenir plus de données d'entraînement, c'est-à-dire plus de chemins de parcours de l'arbre. L'analyse des données et la création des modèles ont été réalisées en partie grâce à plusieurs librairies Python telles que scikit-learn et statsmodels. Enfin, la mise en conditions réelles des modèles sous GENCOL a été réalisée sur des PC Linux avec des processeurs Intel® Xeon® E3-1226 v3 3.30 GHz.

## 5.1 Importance des variables

Dans cette section, nous allons déterminer les variables les plus importantes d'après les méthodes décrites précédemment. Puisqu'il n'est pas nécessaire d'obtenir les performances d'un modèle dans cette partie, nous pouvons utiliser l'entièreté de données à notre disposition.

### 5.1.1 Coefficients de corrélation

La tableau 5.1 présente les coefficients de corrélation des variables explicatives les plus corrélées à la réponse. Nous considérons les coefficients de Pearson pour les relations purement linéaires et ceux de Spearman pour les relations monotones.

Parmi ces coefficients, nous pouvons retrouver les deux formes de la valeur fractionnaire, 'Norm frac val' et 'Frac val'. La forme normalisée présente un coefficient de corrélation plus élevé mais ceci est dû au fait que sa distribution est plus proche de celle de la solution grâce à l'utilisation de la softmax. La valeur fractionnaire tend ainsi à être un bon indicateur pour le choix de la rotation de branchement d'après cette méthode. Les informations apportées par FCCPV et FCC semblent aussi contribuer à l'explication de la réponse. Puisque leurs coefficients sont négatifs, cela signifie que plus les valeurs de ces deux variables sont faibles, meilleure sera la solution.

| Variable | Pearson | Spearman |
|---|---|---|
| **Norm frac val** | 0,26 | 0,22 |
| **FCCPV** | -0,19 | -0,21 |
| **FCC** | -0,12 | -0,13 |
| **Frac val** | 0,12 | 0,17 |
| **Autres variables** | < 0,1 | < 0,1 |

*Tableau 5.1: Variables les plus corrélées à la réponse*

Il est intéressant de noter que le plus grand coefficient ne dépasse pas 0.3. Cela signifie donc que l'hypothèse d'une relation linéaire ou simplement monotone entre les variables explicatives et la réponse n'est pas fortement vérifiée. La plupart des variables présentent ainsi des coefficients inférieurs à 0.1 en valeur absolue.

### 5.1.2 Arbres de décision

Nous allons maintenant voir l'importance des variables d'après un modèle de RF. Nous considérons des forêts de 50 arbres de régression non limités en taille. Les résultats sont mis en évidence dans le tableau 5.2. La colonne de gauche indique l'importance des variables lorsque l'apprentissage de la RF a été fait en utilisant la MAE comme fonction de perte tandis que la colonne de droite correspond à un apprentissage basé sur la MSE.

| **MAE** | **MSE** |
|---|---|
| Norm frac val<br>0,13 | Norm frac val<br>0,17 |
| Frac val<br>0,12 | Frac val<br>0,09 |
| FCCPV<br>0,085 | FCCPV<br>0,09 |
| FPTF<br>0,085 | Nb cols in MP<br>0,089 |
| Nb cols in MP<br>0,077 | FPTF<br>0,085 |
| Autres variables<br>< 0,07 | Autres variables<br>< 0,065 |

*Tableau 5.2: Importance des meilleures variables par les critères MAE et MSE d'une RF*

Dans les deux situations, ce sont les 5 mêmes variables qui présentent le plus d'importance dans l'explication de la réponse. 'Norm frac val' et 'Frac val' sont les deux variables les plus significatives d'après cette méthode. À l'instar des résultats sur la corrélation, nous retrouvons aussi 'FCCPV' en tête du classement. Il est intéressant de souligner que les deux autres variables considérées importantes ici sont 'Nb cols in MP' et 'FPTF'. Leur particularité est d'être de valeur constante pour tous les candidats d'un même étage. Ces variables décrivent donc en quelque sorte l'avancement de la résolution plutôt que l'état d'un nœud en particulier. Cela peut donc signifier que les variables propres à un candidat ne seraient pas suffisantes pour une décision de branchement et qu'il serait utile de savoir la position dans l'arbre pour ajuster la prise de décision.

### 5.1.3 Sélection itérative

Les deux méthodes utilisées précédemment indiquaient qu'essentiellement 5 variables participaient à l'explication de la réponse. Les méthodes de SSV et de ERV nécessitant le nombre de variables à considérer dans les modèles comme hyperparamètre, nous choisirons donc de n'en garder que 5 pour notre étude. Le tableau 5.3 présente le classement des variables de la plus importante à la moins significative. Nous distinguons le cas d'une SSV *forward* où l'on ajoute les variables au fur et à mesure de la version *backward* où ces dernières sont successivement retirées. De plus, les 3 méthodes sont appuyées par l'utilisation d'une régression linéaire simple et d'une RF de 50 arbres. La SSV utilise la MSE comme métrique d'évaluation.

| **SSV forward** | | **SSV backward** | | **ERV** | |
|---|---|---|---|---|---|
| **Reg. Lin.** | **RF** | **Reg. Lin.** | **RF** | **Reg. Lin** | **RF** |
| Norm frac val | Norm frac val | Norm frac val | Norm frac val | Norm frac val | Norm frac val |
| Nb cols in MP | Nb cols in MP | Nb cols in MP | Nb cols in MP | Nb cols in MP | Nb cols in MP |
| FCCPV | FCCPV | FCCPV | FCCPV | FCCPV | FCCPV |
| Frac val | Frac val | Frac val | Frac val | Frac val | Frac val |
| Dual cost min | Dual cost max | Dual cost min | FCC | FPTF | FPTF |

*Tableau 5.3: SSV et ERV pour un modèle à 5 variables*

Pour toutes les approches, nous retrouvons la même tête de classement avec 'Norm frac val', 'Nb cols in MP', 'FCCPV' et 'Frac val'. Le choix de la 5$^{ème}$ variable reste toutefois plus mitigé. L'ERV montre des résultats similaires à ceux de la section précédente, ce qui est logique puisque cette méthode se base sur une mesure d'importance des coefficients. Dans le cas de la régression linéaire pour la SSV, 'Dual cost min' semble apporter des informations aussi bien en *forward* qu'en *backward.* D'autre part, la RF valorise 'Dual cost max' ou la variable 'FCC'. Cette diversité de résultats nous montre qu'au-delà des 4 premières variables évoquées plus haut, les modèles ne gagnent pas beaucoup plus d'informations lorsqu'une nouvelle variable est considérée.

### 5.1.4 Permutation des variables

La dernière méthode que nous utilisons dans cette partie est celle de la permutation des variables. À l'instar de la SSV, nous avons choisi la MSE comme métrique d'évaluation. L'importance sera donc définie comme la différence entre la MSE avant et après permutation. Plus cette différence est élevée, plus la variable permutée est significative dans la prise de décision. Les variables les plus importantes d'après cette méthode sont présentées dans le tableau suivant:

| **Reg. Lin.** | **RF** |
|---|---|
| Norm frac val<br>$2{,}52.10^{-3}$ | Norm frac val<br>$2{,}20.10^{-2}$ |
| FCCPV<br>$9{,}84.10^{-4}$ | FCCPV<br>$1{,}58.10^{-2}$ |
| Frac val<br>$1{,}99.10^{-4}$ | Frac val<br>$1{,}28.10^{-2}$ |
| Nb cols in MP<br>$1{,}80.10^{-4}$ | Nb cols in MP<br>$7{,}73.10^{-3}$ |
| Var cost<br>$1{,}31.10^{-4}$ | FPTF<br>$5{,}65.10^{-3}$ |
| Autres variables<br>$< 10^{-4}$ | Autres variables<br>$< 5.10^{-3}$ |

*Tableau 5.4: Importance des meilleures variables par la méthode de permutation des variables*

Nous retrouvons encore une fois les 4 variables les plus significatives des sections 5.1.2 et 5.1.3. Concernant la régression linéaire, on remarque que la 1ère et la 3ème variables sont séparées d'un facteur 10. Cela montre une nouvelle fois que la 'Norm frac val' est décisive pour la prise de décision. Cet écart est cependant moins marqué dans les résultats de la RF qui ne présente qu'un facteur proche de 2 entre ces mêmes variables. Comme évoqué dans la section 4.2.3, cela vient du fait que les RF permettent de modéliser des interactions plus complexes entre les variables qu'une simple régression linéaire. L'utilisation d'un modèle plus expressif et moins biaisé devrait donc sûrement pouvoir tirer parti d'un plus grand nombre de subtilités cachées dans les données.

## 5.2 Régressions linéaires

Dans chacune des sous-parties suivantes, nous présenterons successivement les variables les plus importantes à partir de la valeur des coefficients des modèles puis nous observerons les performances théoriques de chacune des régressions.

### 5.2.1 Importance de la valeur fractionnaire

Dans un premier temps, nous allons observer les effets de la présence de la valeur fractionnaire, normalisée ou non, dans la prédiction de la réponse. Pour rappel, cette variable a été choisie car elle semble apporter le plus d'informations lors du choix de branchement. Les analyses de la section 5.1 ont de plus montré de façon unanime l'importance de l'intégration de cette variable dans la prise de décision. Le tableau 5.5 rapporte les 5 variables les plus importantes d'après la valeur IMP définie en section 4.2.2 et représentée par le pourcentage sous le nom des variables.

| **Modèle 1** | **Modèle 2** | **Modèle 3** | **Modèle 4** |
|---|---|---|---|
| Nb cols in MP<br>48,16 % | Nb cols in MP<br>36,91 % | Norm frac val<br>51,18 % | Norm frac val<br>33,03 % |
| Frac val<br>28,66 % | FCCPV<br>27,96 % | Nb cols in MP<br>17,36 % | Nb cols in MP<br>29,82 % |
| FCCPV<br>8,84 % | Nb pairing tasks<br>8,53 % | FCCPV<br>11,64 % | Frac val<br>14,08 % |
| FPTF<br>3,79 % | Var cost<br>7,14 % | Var cost<br>4,31 % | FCCPV<br>8,03 % |
| Nb pairing tasks<br>2,62 % | Dual cost min<br>4,60 % | Nb pairing tasks<br>3,68 % | Var cost<br>2,98 % |
| Autres variables<br>7,93 % | Autres variables<br>14,86 % | Autres variables<br>11,83 % | Autres variables<br>12,06 % |

Mod. 1: Toutes variables sauf 'Norm frac val'

Mod. 2: Toutes variables sauf 'Frac val' et ' Norm frac val'

Mod. 3: Toutes variables sauf 'Frac val'

Mod. 4: Toutes variables

*Tableau 5.5: Importance des coefficients de régression (Modèles 1 à 4)*

En supplément des observations de la section précédente, 'Var cost' et 'Nb pairing tasks' apparaissent comme étant des variables intéressantes pour la prédiction. Elles restent cependant

généralement négligeables devant le haut du classement. En effet, ‘Nb cols in MP’ devance assez largement les autres variables. Elle cède toutefois sa place lorsque ‘Norm frac val’ est aussi prise en compte dans le modèle. Dans les 4 situations, on peut remarquer que seulement 5 variables représentent au moins 85% du poids total des coefficients. Les 3 dernières variables de chacun des modèles (‘M3CPV’, ‘Dual cost avg’ et ‘Dual cost max’) représentent en moyenne 2% de ce total.

Le tableau 5.6 regroupe les performances des 4 modèles ainsi que ceux de la *baseline* et d’un modèle de *pure diving* ne contenant que la valeur fractionnaire. Les résultats ont été obtenus par validation croisée.

| Métrique | Baseline | *Pure diving* | Modèle 1 | Modèle 2 | Modèle 3 | Modèle 4 |
|---|---|---|---|---|---|---|
| **MAE** | 0,127 | 0,130 | 0,126 | 0,128 | 0,124 | **0,123** |
| **MSE** | 0,027 | 0,029 | 0,027 | 0,027 | **0,026** | **0,026** |
| **PA** | 39,72 % | 50,20 % | 54,08 % | 49,54 % | **54,58 %** | 54,56 % |
| **AIC** | 9,23 | 9,09 | 31,24 | 29,21 | 31,35 | **33,35** |
| **BIC** | 18,74 | 16,26 | 117,24 | 108,05 | 117,35 | **126,52** |

*Tableau 5.6: Performances théoriques des modèles 1 à 4*

La *baseline*, dont le choix de la rotation de branchement est fait aléatoirement, présente une MAE et MSE relativement faibles quand on considère la simplicité du modèle. Sa PA est toutefois assez basse en comparaison des autres modèles. Au contraire, le *pure diving* a une PA très élevée alors que le modèle ne prend en compte que la valeur fractionnaire. En comparaison, le modèle 2 qui ne contient aucune forme de cette variable présente une PA plus faible. Cela peut être vu comme un indicateur de l’importance de la valeur fractionnaire. De plus, c’est lorsqu’elle est présente sous sa forme normalisée dans les modèles 3 et 4, que les performances sont les meilleures.

D’après ces premiers résultats, nous pouvons considérer que la valeur fractionnaire est d’une grande importance dans la prise de décision. Toutefois, il est nécessaire de compléter un modèle par d’autres variables afin de gagner en expressivité et en précision.

### 5.2.2 Ajout de la régularisation

Nous allons maintenant observer les effets d'une régularisation appliquée sur un modèle contenant l'ensemble de nos variables. Plus précisément, nous allons utiliser la régularisation L2 (modèle 6). Nous avions aussi évoqué l'utilisation de la régularisation L1 dans la section 4.2.2. Toutefois, les résultats obtenus avec cette méthode n'ont pas été concluants. En effet, la valeur de régularisation était tellement faible que l'on obtenait des résultats similaires à ceux du modèle 4. Nous ne considérerons donc pas le modèle 5 dans la suite de nos expériences.

La recherche du meilleur hyperparamètre λ en fonction de la PA du modèle est montrée dans la figure 5.1 ci-après pour la régularisation L2. Le modèle a été successivement entrainé avec 1000 valeurs de λ différentes pour imposer une pénalisation de plus en plus forte et voir l'effet sur la PA.

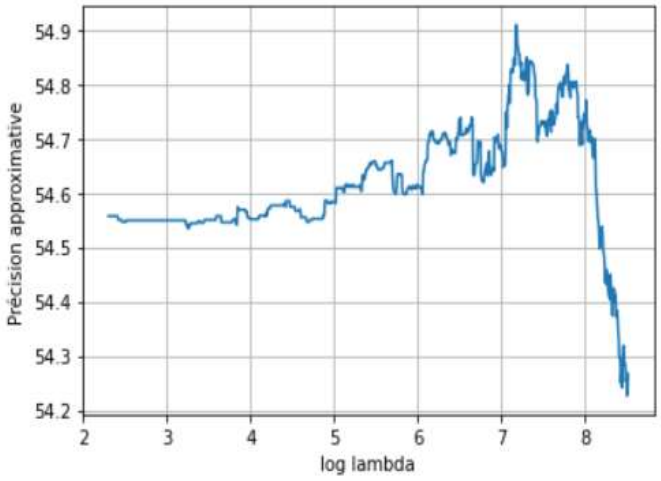


*Figure 5.1: Recherche du meilleur facteur de régularisation pour la L2*

Nous obtenons l'hyperparamètre $ln(\lambda_{ridge}) = 7.18$ soit $\lambda_{ridge} = 1{,}31.10^3$ comme étant optimal au regard de la PA. Une fois cette valeur obtenue, il est possible d'entraîner notre modèle pour obtenir l'importance des variables du tableau 5.7. L'utilisation de la régularisation L2 montre une nouvelle fois que la valeur fractionnaire normalisée occupe une place très importante dans la prise de décision. Son IMP représente quasiment la moitié du poids total des coefficients. Cette place centrale est obtenue au détriment d'une importance beaucoup plus faible de la variable 'Nb cols in MP', habituellement en 2ème position et maintenant en 4ème.

| Modèle 4 | Modèle 6 |
|---|---|
| Norm frac val<br>33,03 % | Norm frac val<br>46,77 % |
| Nb cols in MP<br>29,82 % | FCCPV<br>15,01 % |
| Frac val<br>14,08 % | Frac val<br>13,26 % |
| FCCPV<br>8,03 % | Nb cols in MP<br>5,82 % |
| Var cost<br>2,98 % | Var cost<br>3,98 % |
| Autres variables<br>12,06 % | Autres variables<br>15,16 % |

Mod. 4: Toutes variables

Mod. 6: Toutes variables + régularisation L2

*Tableau 5.7: Importance des coefficients de régression (Modèles 4 et 6)*

Le modèle 6 a des performances relativement identiques à celles du modèle 4 mais fait preuve d'une PA plus élevée. Une cause pouvant expliquer cette augmentation doit provenir de la plus grande importance de 'Norm frac val' dans la décision. Nous garderons donc ce modèle comme étant le meilleur pour la suite de nos expériences.

| Métrique | Modèle 4 | Modèle 6 |
|---|---|---|
| **MAE** | **0,123** | 0,124 |
| **MSE** | **0,026** | **0,026** |
| **PA** | 54,56 % | **54,91 %** |
| **AIC** | **33,35** | 33,33 |
| **BIC** | **126,52** | 126,50 |

*Tableau 5.8: Performances théoriques des modèles 4 et 6*

### 5.2.3 Interactions entre variables

Les derniers modèles que nous allons étudier ont pour but de voir l'effet des interactions entre variables. Cela revient en quelque sorte à augmenter artificiellement le nombre de variables participant à la prise de décision en intégrant des relations non linéaires. En travaillant avec des polynômes d'ordre 2, on obtient tout d'abord une régression avec 105 variables (modèle 7). Dans un second temps, on choisit de ne retenir que les variables qui se sont révélées les plus significatives jusqu'à maintenant, c'est-à-dire 'Norm frac val', 'Frac val', 'Nb cols in MP' et FCCPV, pour former un modèle à 15 variables (modèle 8).

Le tableau 5.9 regroupe les IMP des variables les plus importantes. Nous pouvons directement constater que les modèles 7 et 8 présentent le même classement. Les IMP sont aussi du même ordre de grandeur. Parmi les 10 premières variables, 6 d'entre elles sont des interactions pour les 2 modèles. Cela montre que ces relations non linéaires occupent une place relativement importante. On peut aussi noter que ce sont les variables 'Norm frac val' et 'Nb cols in MP' qui semblent être les plus significatives dans la prise de décision. En outre, lorsque l'on pousse les observations aux 8 premières variables, on voit apparaître (Norm frac val)², Frac val et (Frac val)².

| Modèle 6 | Modèle 7 | Modèle 8 |
|---|---|---|
| Norm frac val<br>46,77 % | Nb cols in MP x Norm frac val<br>16,56 % | Nb cols in MP x Norm frac val<br>20,45 % |
| FCCPV<br>15,01 % | Norm frac val<br>16,17 % | Norm frac val<br>19,74 % |
| Frac val<br>13,26 % | Nb cols in MP<br>11,33 % | Nb cols in MP<br>13,54 % |
| Nb cols in MP<br>5,82 % | Nb cols in MP x Nb cols in MP<br>9,62 % | Nb cols in MP x Nb cols in MP<br>11,29 % |
| Var cost<br>3,98 % | Frac val x Norm frac val<br>8,69 % | Frac val x Norm frac val<br>10,53 % |
| Autres variables<br>15,16 % | Autres variables<br>37,63 % | Autres variables<br>24,45 % |

Mod. 6: Toutes variables + régularisation L2

Mod. 7: Polynôme de degré 2 avec toutes les variables

Mod. 8: Polynôme de degré 2 avec le sous-ensemble des meilleures variables

*Tableau 5.9: Importance des coefficients de régression (Modèles 6 à 8)*

On constate une très nette amélioration des performances lors de l'évaluation des modèles polynomiaux. En effet, pour un nombre de variables quasiment similaire à celui du modèle 6, le modèle 8 voit ses MAE et MSE descendre de quelques points. Le modèle polynomial complet obtient quant à lui une PA 1.28% plus élevée que celle du modèle 6. Les valeurs de l'AIC et du BIC du modèle 7 viennent du fait que le nombre de variables est plus grand que pour les autres modèles. À MSE équivalente, ces métriques vont privilégier des modèles contenant plus de paramètres.

| Métrique | Modèle 6 | Modèle 7 | Modèle 8 |
|---|---|---|---|
| **MAE** | 0,124 | **0,119** | 0,120 |
| **MSE** | 0,026 | **0,024** | **0,024** |
| **PA** | 54,91 % | **56,19 %** | 55,50 % |
| **AIC** | 33,33 | **215,46** | 35,44 |
| **BIC** | 126,50 | **960,85** | 135,78 |

*Tableau 5.10: Performances théoriques des modèles 6 à 8*

À la lumière des analyses faites précédemment, nous pouvons remarquer que les variables principalement retenues sont celles qui communiquent des informations sur les colonnes du problème de relaxation linéaire. Les données concernant les rotations candidates en elles-mêmes semblent donc être plus significatives que celles concernant les vols ou lignes du problème.

## 5.3 Expérimentations sous GENCOL

Nous allons maintenant pouvoir évaluer les performances réelles de nos modèles sous GENCOL et les comparer aux heuristiques de *pure* et *strong diving*. Les modèles retenus pour cette évaluation sont les modèles 4, 6 et 7 ayant présentés les meilleurs résultats dans leur partie respective. Comme évoqué en début de chapitre, l'initialisation de l'arbre de B&P fait face à quelques variations lors de la génération des premières rotations, impactant la suite de la résolution. Afin de prendre en compte la variance des résultats engendrée par ce phénomène, chaque instance a été résolue 4 fois dans le but d'obtenir des valeurs moyennes.

Dans un premier temps, le tableau 5.11 récapitule le pourcentage d'amélioration des solutions moyennes des différentes heuristiques par rapport au *strong diving* tel que:

$$val_h = 100 * \frac{sol_h - sol_{SD}}{sol_{SD}}$$

Les résultats en gras correspondent aux meilleures valeurs et ceux soulignés indiquent les deuxièmes meilleures valeurs. Nous pouvons tout d'abord constater que le *pure diving* présente des résultats meilleurs que ceux du *strong diving* sur les plus petites instances (1 et 3). Il est dès lors plus compliqué de comparer nos heuristiques à celles du *pure diving* pour ces instances. Nos modèles ayant été entraînés sur les résultats du SB, il n'est techniquement pas possible de battre de manière significative cette heuristique et donc d'obtenir les performances du *pure diving*. Toutefois, nous pouvons voir que sur les plus grandes instances, nos modèles présentent en moyenne de meilleurs résultats, occupant les 1ère et 2ème places du classement. Cela semble signifier que nos heuristiques tendent à bien généraliser pour des problèmes de grandes tailles. Nous pouvons aussi certainement mettre en avant la nécessité d'une prise de décision qui n'est pas uniquement basée sur la valeur fractionnaire.

| | **Instance 1** | **Instance 2** | **Instance 3** | **Instance 4** | **Instance 5** | **Instance 6** | **Instance 7** | **Moyenne** |
|---|---|---|---|---|---|---|---|---|
| ***Pure diving*** | **-0,25** | **0,03** | **-0,61** | 0,76 | 0,23 | 0,47 | **0,27** | **0,13** |
| **Modèle 4** | 0,87 | 0,58 | 0,42 | **0,56** | 0,12 | 0,36 | 0,31 | 0,46 |
| **Modèle 6** | 1,85 | 0,69 | 0,26 | 0,75 | **0,05** | **0,31** | 0,53 | 0,64 |
| **Modèle 7** | 0,79 | 0,11 | 0,24 | 0,90 | 0,08 | 0,34 | **0,27** | 0,39 |

*Tableau 5.11: Comparaison des solutions moyennes par rapport au strong diving (%)*

Le tableau 5.12 reprend la comparaison des solutions de chacune des heuristiques par rapport au *strong diving* mais en regardant uniquement les meilleures solutions au lieu de la moyenne. Les résultats semblent encourageants pour les instances de toute taille. Plus particulièrement, le modèle 4 propose des performances qui sont en moyenne à égale distance entre les heuristiques de *pure diving* et de SB. Le modèle 7 se place aussi relativement bien, notamment sur les plus grandes instances. La présence d'interactions permettrait de fournir de meilleures performances lorsque le nombre de vols est élevé. Bien que le *pure diving* ait proposé de bons

résultats en moyenne, nous pouvons voir que sa meilleure solution est généralement assez éloignée de celle du SB.

| | Instance 1 | Instance 2 | Instance 3 | Instance 4 | Instance 5 | Instance 6 | Instance 7 | Moyenne |
|---|---|---|---|---|---|---|---|---|
| ***Pure diving*** | 0,04 | 0,19 | 0,04 | 0,85 | 0,30 | 0,52 | 0,37 | 0,33 |
| **Modèle 4** | **-0,06** | 0,23 | 0,26 | **0,32** | **0,05** | 0,24 | 0,29 | **0,19** |
| **Modèle 6** | 0,59 | 0,21 | 0,65 | 0,56 | 0,06 | 0,29 | 0,47 | 0,4 |
| **Modèle 7** | 0,80 | **0,06** | **-0,02** | 0,79 | 0,10 | **0,10** | **0,10** | 0,28 |

*Tableau 5.12: Comparaison des meilleures solutions par rapport au strong diving (%)*

Nos modèles seraient essentiellement utiles aux compagnies aériennes sur des instances de grandes tailles car ce sont sur ces dernières que l'écart entre les temps de résolution du *pure* et *strong* diving se creuse. Notons de plus que les compagnies utilisent en pratique une heuristique de branchement sur la meilleure valeur fractionnaire semblable au *pure diving*. Lorsque nous considérons uniquement les instances 4 à 7, le *strong diving* améliore la solution d'environ 0.5% par rapport au *pure diving* tandis que meilleur modèle (modèle 4) propose une amélioration d'environ 0.28% par rapport à cette même heuristique. Sachant qu'une amélioration de la solution de 1% mène généralement à une économie de 5 à 10 millions de dollars par an pour une compagnie aérienne, notre modèle le plus simple mais le plus efficace pourrait permettre un gain de 1.5 à 3 millions de dollars par an.

Le tableau 5.13 présente la comparaison des temps d'exécution par rapport au *pure diving.* Nous pouvons voir que malgré ses bonnes solutions, le SB n'est pas une heuristique viable car elle met en moyenne trois fois plus de temps à arriver au bout de la résolution. Nos modèles montrent quant à eux des performances similaires à celles du *pure diving* et arrivent même régulièrement à résoudre les instances plus rapidement. La différence de temps de résolution entre le *strong diving* et les autres heuristiques s'explique par les relaxations supplémentaires à résoudre pour le *strong diving*. Le temps d'évaluation des nœuds est grandement négligeable en comparaison.

| | Instance 1 | Instance 2 | Instance 3 | Instance 4 | Instance 5 | Instance 6 | Instance 7 | Moyenne |
|---|---|---|---|---|---|---|---|---|
| ***Strong diving*** | 265,69 | 264,27 | 318,00 | 331,25 | 378,10 | 349,62 | 351,99 | 322,70 |
| **Modèle 4** | -1,97 | -0,93 | 8,30 | -5,44 | -1,35 | 2,26 | -2,01 | -0,16 |
| **Modèle 6** | **-5,10** | **-1,24** | 8,40 | -3,31 | **-2,47** | -0,39 | 1,51 | **-0,37** |
| **Modèle 7** | 0,21 | 2,88 | **6,67** | **-5,50** | -1,79 | **-1,31** | **-3,23** | -0,29 |

*Tableau 5.13: Comparaison du temps de résolution moyen par rapport au pure diving (%)*

Le tableau 5.14 regroupe le nombre de nœuds moyen évalués lors de la résolution. Nous pouvons voir que lorsque le nombre de nœuds visités est élevé, le temps d'exécution suit une tendance similaire. Ainsi les modèles 6 et 7 qui sont en moyenne les heuristiques les plus rapides, sont aussi celles qui parcourent le moins de nœuds dans l'arbre. La figure 5.2 montre une relation relativement linéaire ($R^2 = 0.85$) entre ces deux variables.

| | Instance 1 | Instance 2 | Instance 3 | Instance 4 | Instance 5 | Instance 6 | Instance 7 |
|---|---|---|---|---|---|---|---|
| ***Strong diving*** | 900 | 828 | 2 108 | 6 838 | 7 194 | 6 699 | 9 435 |
| ***Pure diving*** | 148 | 135 | 327 | 1 326 | 1 420 | 1 291 | 1 738 |
| **Modèle 4** | 131 | 121 | 326 | 1 258 | 1 304 | 1 267 | **1 714** |
| **Modèle 6** | 134 | 123 | 325 | **1 256** | **1 255** | **1 248** | 1 777 |
| **Modèle 7** | **127** | **116** | **320** | 1 287 | 1 313 | **1 248** | 1 750 |

*Tableau 5.14: Comparaison du nombre de nœuds moyen*

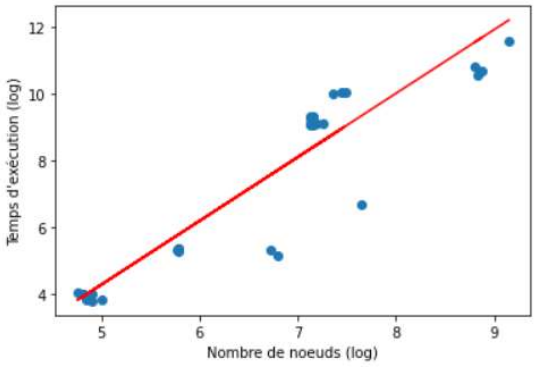


*Figure 5.2: Temps d'exécution en fonction du nombre de nœuds explorés*

# CHAPITRE 6 CONCLUSION

Dans cette maîtrise, nous avons étudié le problème des rotations d'équipages. Plus précisément, nous avons considéré la résolution de ce problème dans un environnement de B&P combiné à une méthode de *diving*.

Nos objectifs étaient les suivants: déterminer les variables les plus significatives de notre environnement permettant d'expliquer les solutions obtenues, et construire un modèle simple d'apprentissage machine pour remplacer l'heuristique de SB.

Après avoir fait un état de l'art des méthodes usuellement employées dans cette optique, nous avons procédé à une phase de normalisation afin de procéder de la manière la plus adéquate à nos analyses. Afin de s'adapter au contexte de résolution séquentielle rencontré dans l'arbre de B&P, nous avons opté pour une normalisation réalisée entre chaque candidat d'un même étage à la place d'une normalisation globale de l'arbre. Nous avons ensuite pu conclure que quatre de nos variables étudiées présentent un intérêt relativement important dans l'explication des solutions. La forme normalisée de la valeur fractionnaire de la rotation candidate présente unanimement le plus de poids dans la prise de décision. Les variables représentant la valeur fractionnaire non normalisée et la fraction des rotations en conflit avec la rotation fixée et de valeur non-nulle dans la solution de la relaxation linéaire au nœud parent font aussi preuve d'un intérêt particulier. La dernière variable retenue indique le nombre de colonnes dans le problème maître au nœud parent. La particularité de cette variable est d'être constante au sein d'un même étage, démontrant ainsi l'intérêt de communiquer des informations sur l'état de la résolution et non pas uniquement sur le candidat.

Les résultats présentés pour répondre à notre deuxième objectif sont très encourageants. Nous avons été capables de proposer des modèles linéaires pouvant atteindre des performances à égale distance entre les heuristiques de *pure diving* et de SB voire très proches du SB sur des problèmes de grandes tailles. Le temps d'exécution requis pour résoudre les instances avec nos modèles est quant à lui quelque peu meilleur que celui du *pure diving*. Ceci s'explique par le fait que nos modèles explorent moins de nœuds lors de la résolution. D'un point de vue financier, notre meilleur

modèle, aussi simple soit-il, pourrait faire économiser jusqu'à 3 millions de dollars par an aux compagnies aériennes sans impacter leur temps de résolution actuel.

Plusieurs pistes de recherche sont à explorer pour la suite. Des méthodes peuvent ainsi être envisagées pour chercher à réduire encore plus le temps de résolution. Permettre au modèle de choisir plusieurs candidats sur lesquels se brancher en même temps rendrait certainement le processus plus efficace. Il faudrait toutefois trouver un critère adéquat pour éviter de faire des branchements inutiles qui réduiraient la qualité de la solution. D'autre part, nous pourrions envisager une heuristique hybride entre SB et ML. Cette dernière pourrait utiliser le ML sur une première partie de l'arbre puis passer au SB afin de terminer la résolution de façon plus fine. De plus, les problèmes en fin d'arbre sont plus petits ce qui signifie que le temps de résolution du SB est moindre. Cette méthode pourrait donc permettre d'obtenir de meilleures solutions que celles proposées actuellement par le ML seul sans pour autant avoir un impact trop significatif sur le temps d'exécution.

Des limites peuvent apparaître dans notre étude et il serait intéressant d'y apporter des solutions dans des travaux ultérieurs. La première est relative au nombre de variables que nous avons à notre disposition. Alors que plus de 70 variables sont utilisées dans les travaux de [37] et [39], n'en exploiter que 13 peut restreindre les performances de nos modèles. De plus amples recherches afin de déterminer de nouvelles variables pourraient permettre d'accéder à une meilleure modélisation des données. Nous pourrions par exemple prendre en compte le nombre de vols car nos résultats ont montré que nos modèles ont des performances plus ou moins bonnes en fonction de la taille de l'instance. De plus, la normalisation utilisée sur la valeur fractionnaire semble avoir porté ses fruits en améliorant les performances de nos modèles. Il serait judicieux de continuer d'explorer cette piste en recherchant une normalisation adéquate pour chacune des variables à notre disposition. Cette étape est primordiale car elle est à la base de tous les résultats obtenus. Concernant les modèles étudiés, nous avons pu montrer l'importance des interactions afin de gagner en expressivité. Une régression polynômiale permet d'avoir une porte d'entrée sur les relations non-linéaires et au regard des résultats, il pourrait être intéressant d'exploiter des modèles d'apprentissage machine plus complexes. Une simple évaluation du coefficient $R^2$ de nos régressions nous montre des valeurs ne dépassant jamais 0.1 ce qui prouve les limites de nos

modèles linéaires. Comme nous l'avons vu, les arbres de décisions réussissent à capturer des relations que les régressions linéaires ne peuvent pas voir. D'autres modèles tels que des machines à vecteurs de support ou une adaptation de l'algorithme des K plus proches voisins pour la régression pourraient donner de meilleurs résultats au détriment de l'interprétabilité. La structure séquentielle du problème sous forme d'arbre et le contexte de *diving* rendent même envisageable l'utilisation de modèles d'apprentissage profond comme les réseaux récurrents neuronaux. Une dernière limite que nous pouvons évoquer se trouve dans le nombre de candidats évalués à chaque étage. Nous en considérons actuellement 5 au maximum. Toutefois, il pourrait être intéressant d'augmenter ce nombre, d'autant plus que l'impact sur le temps de résolution ne sera pas assez grand pour être rédhibitoire.

Nous pouvons faire une dernière remarque sur les résultats obtenus. Comme nous l'avons constaté dans les analyses de la section 5.3, les modèles réussissent à relativement bien généraliser pour de grandes instances mais leurs performances restent assez en deçà de celles du SB. Cet écart pourrait probablement être expliqué par du *distributional shifting*. Lors de l'entraînement, les modèles sont forcés de suivre les mêmes décisions que le SB. Sous GENCOL, ces mêmes modèles font leurs propres décisions de branchement et dévient vers des données qu'ils n'avaient pas l'habitude de voir pendant l'entraînement. Ils rencontrent alors de nouvelles distributions de données pour lesquelles leurs prédictions sont moins précises et cette différence de distribution s'accentue de plus en plus au fur et à mesure de l'avancement de la résolution. Le *distributional shifting* se retrouve dans le domaine de l'apprentissage par renforcement où un agent évolue dans un environnement dynamique. Pour contrer ce problème, certaines approches utilisent l'algorithme DAgger (Dataset Aggregation) proposé par Ross et al. [43]. L'idée consiste à laisser l'agent évoluer dans l'environnement puis à utiliser les données qu'il vient de voir pour affiner ses connaissances, ou autrement dit les paramètres du modèle. Appliqué à notre cas, cela reviendrait à laisser notre modèle, entraîné initialement par SB, résoudre des instances dans lesquelles il rencontrerait de nouvelles distributions de données (figure 6.1). On considérerait les trajectoires suivies par le modèle, c'est-à-dire les valeurs de variables associées à chacun des nœuds explorés comme nouveau jeu d'entraînement pour ajuster les paramètres du modèle. Il serait de plus nécessaire d'obtenir la solution de chacun des nœuds grâce à l'heuristique de SB pour continuer d'apprendre à partir des décisions de l'expert. Cette approche s'appuie sur les récents travaux de Huang et al. [5] qui propose d'utiliser l'algorithme DAgger pour obtenir plus de données d'entraînement et le

transfert de connaissances afin d'ajuster les paramètres d'un modèle pré-entraîné à ces nouvelles données.

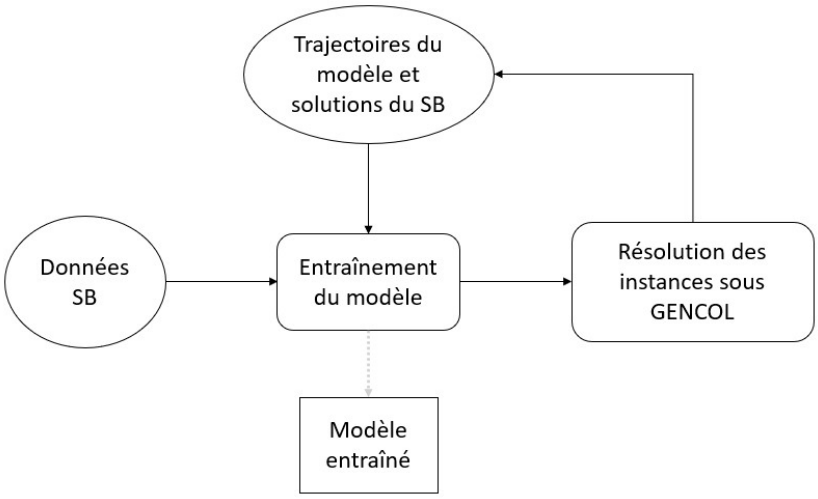


*Figure 6.1: Processus de l'algorithme DAgger*

# RÉFÉRENCES

[1] « Transport aérien, voyageurs transportés | Data ». https://donnees.banquemondiale.org/indicator/IS.AIR.PSGR (consulté le 9 mars 2022).

[2] « Air Passenger Numbers to Recover in 2024 ». https://www.iata.org/en/pressroom/2022-releases/2022-03-01-01/ (consulté le 5 juin 2022).

[3] A. H. Land et A. G. Doig, « An automatic method of solving discrete programming problems », *Econometrica*, 1960.

[4] N. Furian, M. O'Sullivan, C. G. Walker, et E. Çela, « A machine learning-based branch and price algorithm for a sampled vehicle routing problem », *Spectr.*, vol. 43, nº 3, p. 693-732, 2021.

[5] L. Huang *et al.*, « Branch and Bound in Mixed Integer Linear Programming Problems: A Survey of Techniques and Trends. », *ArXiv Learn.*, nov. 2021.

[6] P. G. Falk, « Experiments in mixed integer linear programming in a manufacturing system », *Omega-Int. J. Manag. Sci.*, vol. 8, nº 4, p. 473-484, janv. 1980.

[7] D. Applegate, R. E. Bixby, V. Chvátal, et B. Cook, « Finding Cuts in the TSP (A preliminary report) », *Cent. Discrete Math. Theor. Comput. Sci.*, avr. 1995.

[8] S. S. Dey, Y. Dubey, M. Molinaro, et P. Shah, « A Theoretical and Computational Analysis of Full Strong-Branching. », *ArXiv Prepr. ArXiv211010754*, oct. 2021.

[9] T. Achterberg, T. Koch, et A. Martin, « Branching rules revisited », *Oper. Res. Lett.*, vol. 33, nº 1, p. 42-54, janv. 2005.

[10] T. Achterberg et T. Berthold, « Hybrid Branching », *CPAIOR*, p. 309, mai 2009.

[11] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, et P. H. Vance, « Branch-And-Price: Column Generation for Solving Huge Integer Programs », *Oper. Res.*, vol. 46, nº 3, p. 316-329, mars 1998.

[12] J. Desrosiers, F. Soumis, et M. Desrochers, « Routing with time windows by column generation », *Networks*, vol. 14, nº 4, p. 545-565, déc. 1984.

[13] S. Lavoie, M. Minoux, et E. Odier, « A new approach for crew pairing problems by column generation with an application to air transportation », *Eur. J. Oper. Res.*, vol. 35, nº 1, p. 45-58, avr. 1988.

[14] P. H. Vance *et al.*, « A Heuristic Branch-and-Price Approach for the Airline Crew Pairing Problem », 1997.

[15] R. Sadykov, F. Vanderbeck, A. Pessoa, I. Tahiri, et E. Uchoa, « Primal Heuristics for Branch and Price: The Assets of Diving Methods », *Inf. J. Comput.*, vol. 31, nº 2, p. 251-267, avr. 2019.

[16] D. Klabjan, « Large-Scale Models in the Airline Industry », in *Column Generation*, Springer, 2005, p. 163-195.

[17] M. Saddoune, G. Desaulniers, et F. Soumis, « Aircrew pairings with possible repetitions of the same flight number », *Comput. Oper. Res.*, vol. 40, nº 3, p. 805-814, mars 2013.

[18] M. Saddoune, G. Desaulniers, I. Elhallaoui, et F. Soumis, « Integrated Airline Crew Pairing and Crew Assignment by Dynamic Constraint Aggregation », *Transp. Sci.*, vol. 46, nº 1, p. 39-55, févr. 2012.

[19] O. Ö. Özener, M. O. Matoglu, G. Erdoğan, M. Haouari, et H. Sözer, « Solving a Large-Scale Integrated Fleet Assignment and Crew Pairing Problem », *Ann. Oper. Res.*, vol. 253, nº 1, p. 477-500, juin 2017.

[20] D. Montgomery et E. A. Peck, « Introduction to Linear Regression Analysis », *J. R. Stat. Soc. Ser. C-Appl. Stat.*, déc. 1981.

[21] F. H. C. Marriott, J. Neter, W. Wasserman, et M. H. Kutner, « Applied Linear Regression Models », *Biometrics*, janv. 1983.

[22] J. Frost, *Regression Analysis: An Intuitive Guide for Using and Interpreting Linear Models*. Statistics By Jim Publishing, 2019.

[23] T. Hastie, R. Tibshirani, et J. H. Friedman, « The Elements of Statistical Learning: Data Mining, Inference, and Prediction », *Math. Intell.*, mars 2005.

[24] S. Haldar et A. J. Miller, « Subset Selection in Regression », *J. Mark. Res.*, vol. 29, nº 2, p. 270, mai 1992.

[25] D. Bertsimas, A. King, et R. Mazumder, « Best subset selection via a modern optimization lens », *Ann. Stat.*, vol. 44, nº 2, p. 813-852, avr. 2016.

[26] A. Tikhonov, « Solution of incorrectly formulated problems and the regularization method », *Sov. Math*, vol. 4, p. 1035-1038, 1963.

[27] A. E. Hoerl et R. Kennard, « Ridge Regression: Biased Estimation for Nonorthogonal Problems », *Technometrics*, 2000.

[28] R. Tibshirani, « Regression Shrinkage and Selection via the Lasso », *J. R. Stat. Soc. Ser. B-Methodol.*, vol. 58, nº 1, p. 267-288, janv. 1996.

[29] T. Hastie, R. Tibshirani, et R. J. Tibshirani, « Extended Comparisons of Best Subset Selection, Forward Stepwise Selection, and the Lasso », *ArXiv Prepr. ArXiv170708692*, juill. 2017.

[30] L. Breiman, « Random Forests », in *Machine learning*, vol. 45, Springer, 2001, p. 5-32.

[31] A. Fisher, C. Rudin, et F. Dominici, « All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously », *J. Mach. Learn. Res.*, vol. 20, nº 177, p. 1-81, 2019.

[32] H. Akaike, « A new look at the statistical model identification », *IEEE Trans. Autom. Control*, vol. 19, nº 6, p. 716-723, déc. 1974.

[33] I. Goodfellow, Y. Bengio, et A. Courville, « Conditional Log-Likelihood and Mean Squared Error », in *Deep Learning*, MIT Press, 2016, p. 131-132. [En ligne]. Disponible sur: http://www.deeplearningbook.org

[34] G. Schwarz, « Estimating the Dimension of a Model », *Ann. Stat.*, vol. 6, nº 2, p. 461-464, mars 1978.

[35] A. M. Alvarez, Q. Louveaux, et L. Wehenkel, « A Supervised Machine Learning Approach to Variable Branching in Branch-And-Bound », *ECML*, 2014.

[36] A. M. Alvarez, L. Wehenkel, et Q. Louveaux, « Online Learning for Strong Branching Approximation in Branch-and-Bound », 2016.

[37] E. B. Khalil, P. Le Bodic, L. Song, G. L. Nemhauser, et B. Dilkina, « Learning to branch in Mixed Integer Programming », *AAAI*, vol. 30, nº 1, p. 724-731, févr. 2016.

[38] Y. Yang, N. Boland, B. Dilkina, et M. W. P. Savelsbergh, « Learning Generalized Strong Branching for Set Covering, Set Packing, and 0-1 Knapsack Problems », *Eur. J. Oper. Res.*, nov. 2021.

[39] G. Zarpellon, J. Jo, A. Lodi, et Y. Bengio, « Parameterizing Branch-and-Bound Search Trees to Learn Branching Policies », *AAAI*, vol. 35, nº 5, p. 3931-3939, 2021.

[40] K. Yilmaz et N. Yorke-Smith, « A Study of Learning Search Approximation in Mixed Integer Branch and Bound: Node Selection in SCIP », *Artif. Intell.*, vol. 2, nº 2, p. 150-178, avr. 2021.

[41] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, et A. Lodi, « Exact Combinatorial Optimization with Graph Convolutional Neural Networks », *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[42] A. Kasirzadeh, M. Saddoune, et F. Soumis, « Airline crew scheduling: Models, algorithms, and data sets », *EURO J Transp Logist*, p. 1-29, avr. 2014.

[43] S. Ross, G. J. Gordon, et J. A. Bagnell, « A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning », *AISTATS*, vol. 15, p. 627-635, juin 2011.

# ANNEXE A NORMALISATION DES VARIABLES

| VARIABLE | LOGARITHME | STANDARDISATION | MINMAX | SOFTMAX |
|---|---|---|---|---|
| VAR COST | | X | | |
| FRAC VAL | | | | |
| FCC | X | X | | |
| FCCPV | X | X | | |
| M3C | | X | | |
| M3CPV | | X | | |
| NB COLS IN MP | X | | X | |
| DUAL COST MIN | X | X | | |
| DUAL COST MAX | X | X | | |
| DUAL COST AVG | | X | | |
| FPTF | | | | |
| NB PAIRING TASKS | | X | | |
| NORM FRAC VAL | X | X | | X |
| VALUE | | | | X |