

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

[polytech.univ-tours.fr](http://polytech.univ-tours.fr)

**Projet Recherche & Développement**

**2019-2020**

# **Recherche Opérationnelle et Machine Learning**

**Le cas d'une matheuristique pour un problème de  
flowshop**

**Tuteurs académiques**

Romain RAVEAUX

Vincent T'KINDT

Nicolas RAGOT

**Étudiant**

Guillaume CHEVALLIER (DI5)



# Liste des intervenants

| Nom                  | Email  | Qualité  |
|----------------------|--|--|
| Guillaume CHEVALLIER | <a href="mailto:guillaume.chevallier@etu.univ-tours.fr">guillaume.chevallier@etu.univ-tours.fr</a> | Étudiant DI5                                   |
| Romain RAVEAUX       | <a href="mailto:romain.raveaux@univ-tours.fr">romain.raveaux@univ-tours.fr</a>                     | Tuteur académique,<br>Département Informatique |
| Vincent T'KINDT      | <a href="mailto:vincent.tkindt@univ-tours.fr">vincent.tkindt@univ-tours.fr</a>                     | Tuteur académique,<br>Département Informatique |
| Nicolas RAGOT        | <a href="mailto:nicolas.ragot@univ-tours.fr">nicolas.ragot@univ-tours.fr</a>                       | Tuteur académique,<br>Département Informatique |



# Avertissement

Ce document a été rédigé par Guillaume Chevallier susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Romain Raveaux, Vincent T'kindt et Nicolas Ragot susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



## Pour citer ce document

Guillaume Chevallier, *Recherche Opérationnelle et Machine Learning: Le cas d'une matheuristique pour un problème de flowshop*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2019-2020.

```
@mastersthesis{
  author={Chevallier, Guillaume},
  title={Recherche Opérationnelle et Machine Learning: Le cas d'une matheuristique pour
    un problème de flowshop},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2019-2020}
}
```

# Table des matières

|  |          |
|--|----------|
| Liste des intervenants                                   | a        |
| Avertissement  | b        |
| Pour citer ce document                                   | c        |
| Table des matières                                       | i        |
| Table des figures  | vi       |
| Liste des Algorithmes                                    | viii     |
| <b>1 Introduction</b>                                    | <b>1</b> |
| 1 Contexte.....  | 1        |
| 2 Objectifs.....   | 1        |
| 3 Bases méthodologiques .....                            | 2        |
| <b>2 Description générale</b>                            | <b>3</b> |
| 1 Environnement du projet .....                          | 3        |
| 2 Caractéristiques des utilisateurs .....                | 3        |
| 3 Fonctionnalités et structure générale du système ..... | 3        |
| 3.1 Générer une base d'apprentissage.....                | 3        |
| 3.2 Apprendre un modèle de prédiction.....               | 4        |
| 3.3 Lancer la matheuristique.....                        | 4        |
| <b>3 État de l'art</b>                                   | <b>6</b> |
| 1 Le problème du flowshop.....                           | 6        |
| 2 Matheuristique .....                                   | 9        |

|       |  |    |
|-------|--|----|
| 3     | Méthode Lasso .....  | 10 |
| 3.1   | Principe de la méthode.....  | 10 |
| 3.2   | Algorithmes de résolution .....  | 12 |
| 3.3   | Comparaison avec une régression linéaire classique .....   | 12 |
| 4     | <b>Analyse et conception</b> .....   | 14 |
| 1     | Extraction des caractéristiques.....   | 14 |
| 2     | Génération et structure de la base d'apprentissage .....   | 15 |
| 3     | Apprentissage du modèle de prédiction .....  | 16 |
| 4     | Amélioration de la matheuristique .....  | 16 |
| 5     | <b>Mise en oeuvre</b> .....  | 18 |
| 1     | Caractéristiques choisies pour la base d'apprentissage.....  | 18 |
| 1.1   | Famille 1 : Données temporelles sur la séquence .....  | 19 |
| 1.2   | Famille 2 : Distribution des travaux n'étant pas dans l'ordre croissant des durées opératoires sur la seconde machine dans la fenêtre d'optimisation avec 3 quantiles..... | 19 |
| 1.3   | Famille 3 : Distribution des temps morts sur la machine 2 dans la fenêtre d'optimisation avec 3 quantiles.....   | 20 |
| 1.4   | Famille 4 : Distribution des ratios des durées opératoires pour chaque travail avec 3 quantiles .....  | 20 |
| 1.5   | Famille 5 : Borne inférieure .....   | 20 |
| 1.6   | Famille 6 : Distribution des temps morts sur la machine 2 à droite de la fenêtre d'optimisation avec 3 quantiles .....   | 21 |
| 1.7   | Famille 7 : Déciles de la famille 2.....   | 21 |
| 1.8   | Famille 8 : Déciles de la famille 3.....   | 21 |
| 1.9   | Famille 9 : Déciles de la famille 4.....   | 21 |
| 1.10  | Famille 10 : Déciles de la famille 6.....  | 21 |
| 1.11  | Famille 11-1 : Critères min, max et écarts types pour les familles 2, 3, 4 et 6 .....  | 22 |
| 1.12  | Famille 11-2 : Durées opératoires des travaux dans la fenêtre .....  | 22 |
| 1.13  | Famille 12 : Caractéristiques SRPT .....   | 22 |
| 2     | Développement.....   | 23 |
| 2.1   | Générateur d'instances .....   | 23 |
| 2.2   | Générateur de bases de test et d'apprentissage.....  | 23 |
| 2.3   | Générateur de modèle de prédiction.....  | 24 |
| 2.4   | Matheuristiques .....  | 24 |
| 3     | Analyses .....   | 24 |
| 3.1   | Analyse de l'impact des caractéristiques sur l'apprentissage.....  | 24 |
| 3.1.1 | Apprentissage 1 : 100 jobs, 2465 individus, RBS.....   | 25 |
| 3.1.2 | Apprentissage 2 : 100 jobs, 12600 individus, RBS.....  | 26 |
| 3.1.3 | Apprentissage 3 : 20 jobs, 4860 individus, sans RBS.....   | 27 |

|          |  |           |
|----------|--|-----------|
| 3.1.4    | Apprentissage 4 : 20 et 50 jobs, 104440 individus, sans RBS.....   | 28        |
| 3.1.5    | Apprentissage 5 : 20 et 50 jobs, 104440 individus, sans RBS, variable cible delta.....                             | 29        |
| 3.1.6    | Apprentissage 6 : 20 et 50 jobs, 9680 individus, sans RBS, variable cible delta, caractéristiques famille 12 ..... | 29        |
| 3.2      | Comparaison avec la matheuristique originale .....   | 30        |
| <b>6</b> | <b>Bilan et conclusion</b>   | <b>32</b> |
| 1        | Bilan du semestre 10 .....   | 32        |
| 1.1      | Fait.....  | 32        |
| 1.2      | Reste à faire.....   | 32        |
| 2        | Bilan sur la qualité .....   | 33        |
| 3        | Bilan auto-critique sur la gestion du projet .....   | 33        |
|          | <b>Annexes</b>   | <b>34</b> |
| <b>A</b> | <b>Spécifications fonctionnelles</b>   | <b>35</b> |
| 1        | Programme d'apprentissage (langage Python) .....   | 35        |
| 1.1      | Fonction 1.1 : diviserBaseApprentissage.....   | 35        |
| 1.2      | Fonction 1.2 : générerModele .....   | 35        |
| 1.3      | Fonction 1.3 : testerModele .....  | 36        |
| 1.4      | Fonction 1.4 : exporterModele.....   | 36        |
| 2        | Génération de la base d'apprentissage (langage C++).....   | 36        |
| 2.1      | Fonction 2.1 : obtenirVecteurCaracteristiques.....   | 36        |
| 2.2      | Fonction 2.2 : exporterBaseApprentissage.....  | 37        |
| 3        | Amélioration de la matheuristique (langage C++) .....  | 37        |
| 3.1      | Fonction 3.1 : chargerModele .....   | 37        |
| 3.2      | Fonction 3.2 : predireAmelioration.....  | 37        |
| <b>B</b> | <b>Spécifications non fonctionnelles</b>   | <b>38</b> |
| 1        | Contraintes de développement et conception .....   | 38        |
| 1.1      | Matériel.....  | 38        |
| 1.2      | Langages.....  | 38        |
| 1.3      | Bibliothèques .....  | 38        |
| 2        | Contraintes de fonctionnement et d'exploitation .....  | 39        |
| 2.1      | Performances.....  | 39        |
| 2.2      | Capacité .....   | 39        |
| 2.3      | Contrôlabilité.....  | 39        |
| 2.4      | Sécurité .....   | 39        |
| 2.5      | Risques .....  | 39        |
| 2.6      | Rendu des livrables.....   | 39        |

|          |  |           |
|----------|--|-----------|
| <b>C</b> | <b>Plan de développement</b>   | <b>40</b> |
| 1        | Découpage du projet en tâches .....  | 40        |
| 1.1      | Tâche 1 : comprendre le problème du flowshop.....                                  | 40        |
| 1.2      | Tâche 2 : comprendre la matheuristique.....  | 40        |
| 1.3      | Tâche 3 : comprendre la construction de la base d'apprentissage.....               | 41        |
| 1.4      | Tâche 4 : comprendre les familles de vecteurs.....                                 | 41        |
| 1.5      | Tâche 5 : planification du projet .....  | 41        |
| 1.6      | Tâche 6 : rédaction des spécifications fonctionnelles .....                        | 41        |
| 1.7      | Tâche 7 : rédaction de l'état de l'art .....                                       | 42        |
| 1.8      | Tâche 8 : formation Python .....   | 42        |
| 1.9      | Tâche 9 : formation méthode de régression LASSO.....                               | 42        |
| 1.10     | Tâche 10 : formation Scikit-Learn .....  | 42        |
| 1.11     | Tâche 11 : préparation de la soutenance S9 .....                                   | 43        |
| 1.12     | Tâche 12 : définition des vecteurs de caractéristiques.....                        | 43        |
| 1.13     | Tâche 13 : génération d'une base de vecteurs de caractéristiques.....              | 43        |
| 1.14     | Tâche 14 : apprentissage du modèle de prédiction.....                              | 43        |
| 1.15     | Tâche 15 : intégration du programme dans la matheuristique .....                   | 44        |
| 1.16     | Tâche 16 : tests.....  | 44        |
| 1.17     | Tâche 17 : rédaction du rapport final.....   | 44        |
| 1.18     | Tâche 18 : préparation de la soutenance S10 .....                                  | 44        |
| 2        | Planification à la fin du S9 .....   | 45        |
| 3        | Planification à la fin du S10 .....  | 46        |
| <b>D</b> | <b>Description des interfaces externes du logiciel</b>                             | <b>47</b> |
| 1        | Interfaces matériel/logiciel .....   | 47        |
| 2        | Interfaces homme/machine .....   | 47        |
| 3        | Interfaces logiciel/logiciel .....   | 47        |
| <b>E</b> | <b>Statistiques de comparaison entre les matheuristiques</b>                       | <b>48</b> |
| <b>F</b> | <b>Documentation d'installation</b>  | <b>49</b> |
| 1        | Générateur d'instances, constructeur de base d'apprentissage, matheuristiques .... | 49        |
| 2        | Constructeur de modèle de prédiction .....   | 49        |
| 3        | Documentation développeurs .....   | 49        |
| <b>G</b> | <b>Documentation d'utilisation</b>   | <b>50</b> |
| 1        | Génération d'instances de flowshop .....   | 50        |
| 2        | Génération d'une base d'apprentissage avec descripteurs .....                      | 50        |
| 3        | Construction d'un modèle de prédiction.....  | 51        |
| 4        | Utilisation du modèle de prédiction dans la matheuristique.....                    | 51        |



|               |    |
|---------------|----|
| Webographie   | 52 |
| Bibliographie | 53 |

# Table des figures

## 2 Description générale

|   |   |   |
|---|---|---|
| 1 | Diagramme de cas d'utilisation "Générer la base d'apprentissage" .....                            | 4 |
| 2 | Diagramme de cas d'utilisation "Apprendre un modèle de prédiction" .....                          | 4 |
| 3 | Diagramme de cas d'utilisation "Chercher des solutions optimales à un problème du flowshop" ..... | 5 |

## 3 État de l'art

|   |   |    |
|---|---|----|
| 1 | Exemple de flowshop à 5 travaux sur m machines (source Wikipedia).....  | 6  |
| 2 | Exemple d'une séquence correcte de 10 travaux .....   | 6  |
| 3 | Exemple d'une résolution du problème de flowshop à 2 machines.....  | 7  |
| 4 | Exemple d'optimisation locale d'une séquence .....  | 9  |
| 5 | Lasso path (17 variables) .....   | 11 |
| 6 | Erreur de prédiction en fonction de lambda .....  | 11 |
| 7 | Itérations de l'algorithme de descente par coordonnée sur une fonction définie dans $\mathbb{R}^2$ (source Wikipédia) ..... | 12 |
| 8 | Comparaison régression classique et Lasso [ <a href="#">WWW4</a> ].....   | 13 |

## 4 Analyse et conception

|   |   |    |
|---|---|----|
| 1 | Séquences non caractérisée (permutation de travaux) ..... | 14 |
| 2 | Séquences caractérisée (vecteur de caractéristiques)..... | 14 |
| 3 | Structure de la base d'apprentissage.....                 | 15 |

## 5 Mise en oeuvre

|   |                        |    |
|---|------------------------|----|
| 1 | Schéma explicatif..... | 18 |
|---|------------------------|----|

|  |   |    |
|--|---|----|
| 2  | Exemple de calcul de la famille 2.....                        | 19 |
| 3  | Exemple de calcul de la famille 3.....                        | 20 |
| 4  | Interactions entre les différents composants .....            | 23 |
| 5  | Format du fichier du modèle de prédiction.....                | 24 |
| 6  | Apprentissage N°1 .....                                       | 25 |
| 7  | Apprentissage N°2.....  | 26 |
| 8  | Apprentissage N°3.....  | 27 |
| 9  | Apprentissage N°4.....  | 28 |
| 10   | Apprentissage N°5.....  | 29 |
| 11   | Apprentissage N°6.....  | 30 |
| 12   | Test diagnostique du modèle (sur 3 instances de 50 jobs)..... | 31 |
| <br><b>C Plan de développement</b>                                 |   |    |
| 1  | Diagramme de Gantt à la fin du S9.....                        | 45 |
| 2  | Diagramme de Gantt à la fin du S10.....                       | 46 |
| <br><b>E Statistiques de comparaison entre les matheuristiques</b> |   |    |
| 1  | Comparaison entre les différentes matheuristiques.....        | 48 |



# Liste des Algorithmes

|   |  |    |
|---|--|----|
| 1 | Matheuristique existante.....                              | 16 |
| 2 | Matheuristique hybridée avec le modèle de prédiction ..... | 17 |

# 1

## Introduction

### 1 Contexte

Ce document présente le Projet de Recherche et Développement sur le sujet "Recherche Opérationnelle et Machine Learning : le cas d'une matheuristique pour un problème de flowshop". Il définira le contexte et les objectifs du projet, les fonctionnalités du système à développer, l'état de l'art sur le problème du flowshop, la conception du système et enfin le bilan sur la première partie du développement du projet.

Ce projet est proposé par l'équipe ROOT (Recherche Opérationnelle : Ordonnancement, Transport) du Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT). Il est encadré par M. Romain Raveaux, M. Vincent T'Kindt et M. Nicolas Ragot qui représentent la maîtrise d'ouvrage. La maîtrise d'oeuvre est réalisée par l'étudiant Guillaume Chevallier.

La résolution du problème du flowshop consiste à trouver l'ordre optimal d'un ensemble de travaux à assigner à un ensemble de machines disposées en série pour que ces travaux soient traités en un temps minimal. La résolution d'une instance de ce problème est connue pour être très coûteuse en temps de calcul : en effet, plus on cherche à résoudre le problème pour un grand nombre de travaux, plus le temps de résolution tend vers l'infini de manière exponentielle (cf **état de l'art**).

Il s'agit pourtant d'un problème d'ordonnancement qui s'applique à de nombreux domaines, notamment dans l'industrie pour les chaînes de production, d'où la nécessité de chercher des méthodes alternatives de résolution moins coûteuses en temps de calcul.

### 2 Objectifs

Nous nous intéresserons dans ce PRD à l'optimisation d'un problème de flowshop à deux machines. L'objectif de ce projet est d'étudier l'apport du machine learning pour la résolution du problème du flowshop. Pour cela, nous partirons d'un algorithme de résolution de type matheuristique (cf **état de l'art**) déjà existant, puis nous intégrerons une méthode de machine learning qui permettra d'économiser du temps de calcul. Cette méthode consistera à prédire,

pour une fenêtre d'optimisation donnée, si l'optimisation de la séquence dans cette fenêtre permet de réduire la somme des temps de calcul, si c'est le cas on lancera la méthode de résolution mathématique dessus, sinon on passera à la fenêtre suivante (cf **analyse et conception**).

Nous étudierons ensuite l'hybridation obtenue pour voir si elle est plus performante que la matheuristique initiale. Le but sera donc de faire en sorte que la matheuristique converge plus rapidement vers des bonnes solutions.

### 3 Bases méthodologiques

La partie développement logiciel du projet sera développée à travers une méthode agile SCRUM avec des réunions hebdomadaires pour discuter de l'avancement du projet. L'avantage de la méthode SCRUM sera de pouvoir construire les différents programmes par itération pour s'assurer que ce qui est développé correspond bien aux attentes du client.

Au cours du projet différents outils seront utilisés :

- Gitlab : pour la gestion de version du code.
- Trello : pour le suivi des tâches à réaliser.
- Microsoft Project : pour la planification du projet.

Le détail de la planification est présent à la fin de ce document (cf. annexe **C**).

# 2

## Description générale

### 1 Environnement du projet

Ce projet s'appuie sur trois parties logicielles déjà existantes :

- La structure d'une base d'apprentissage qui fait suite à un PRD réalisé pendant l'année 2018-2019 durant lequel l'étudiant avait travaillé sur la résolution du problème de flowshop à l'aide de méthodes d'apprentissage par réseaux de neurones.
- Une matheuristique déjà programmée en C++ par le LIFAT et fonctionnelle qui permet de résoudre des instances du problème du flowshop. Cette matheuristique utilise en interne l'API du logiciel CPLEX et nécessite donc que le logiciel CPLEX soit installé sur la machine hôte.
- Une série d'algorithmes qui permettent de calculer des caractéristiques pour les séquences. Ces algorithmes sont développés mais n'ont pas été testés.

Il n'y a pas de dépendances liées au matériel. Le développement se fera sous Windows en Python pour la partie apprentissage, et C++ pour l'amélioration de la matheuristique et la génération de la base d'apprentissage.

### 2 Caractéristiques des utilisateurs

Les utilisateurs du système seront des enseignants chercheurs de l'équipe ROOT travaillant au LIFAT. Ces utilisateurs ont donc de bonnes connaissances en informatique et en ordonnancement.

### 3 Fonctionnalités et structure générale du système

Le système sera basé sur trois fonctionnalités logicielles :

#### 3.1 Générer une base d'apprentissage

Cette première application permettra à l'utilisateur de lancer la génération de sa propre base d'apprentissage en paramétrant le nombre de travaux et la taille de sa base. (cf [analyse et](#)

**conception**). Cette base d'apprentissage sera exportée dans un fichier pour que l'utilisateur puisse la réutiliser.

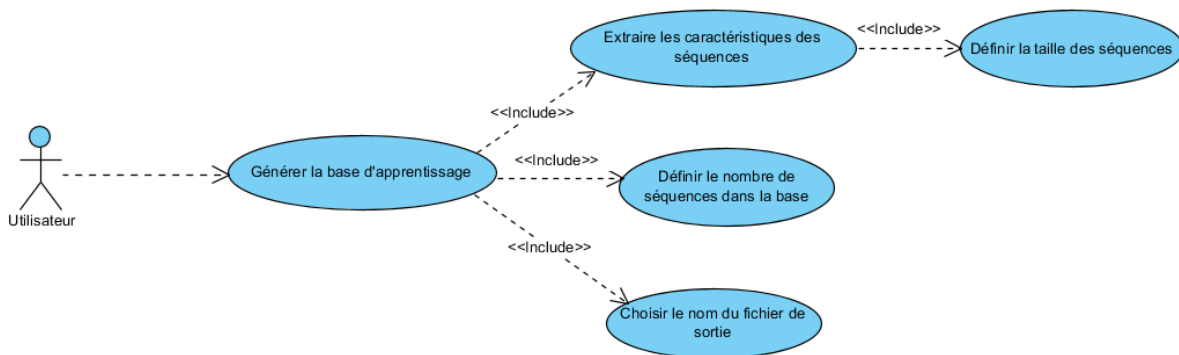


Figure 1 – Diagramme de cas d'utilisation "Générer la base d'apprentissage"

### 3.2 Apprendre un modèle de prédiction

Cette deuxième application permettra à l'utilisateur de construire un modèle de prédiction à partir de la base d'apprentissage exportée par l'application précédente. L'apprentissage se fera avec une méthode de cross-validation à partir de la base d'apprentissage initiale et l'utilisateur pourra paramétrer cette cross-validation en définissant le pourcentage de la base de données qui sera utilisée pour l'apprentissage du modèle (cf **analyse et conception**). L'utilisateur aura également la possibilité d'afficher des statistiques simples sur le modèle créé :

- Qualité de prédiction du modèle.
- Liste des variables utilisées pour l'apprentissage avec leur pourcentage d'influence.

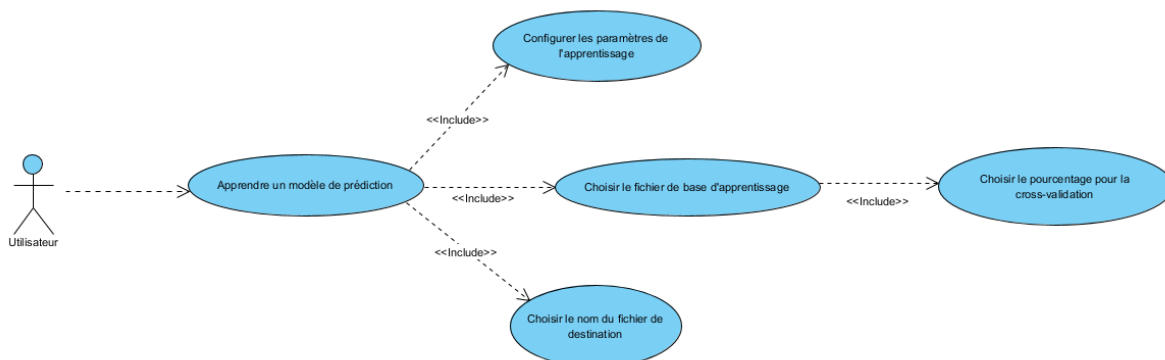
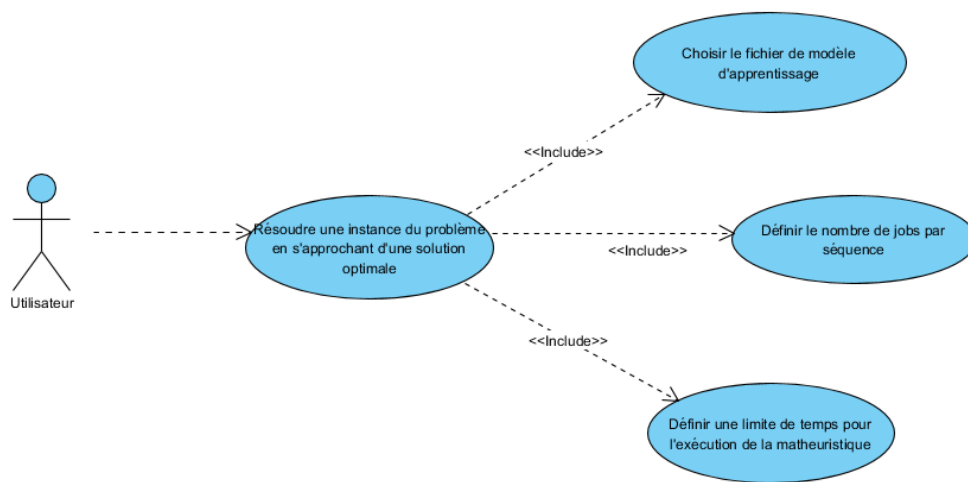


Figure 2 – Diagramme de cas d'utilisation "Apprendre un modèle de prédiction"

### 3.3 Lancer la matheuristique

Cette dernière application permettra à l'utilisateur de lancer la matheuristique hybridée pour résoudre des instances du problème du flowshop en utilisant le modèle de prédiction sur ses propres données. (cf **analyse et conception**)





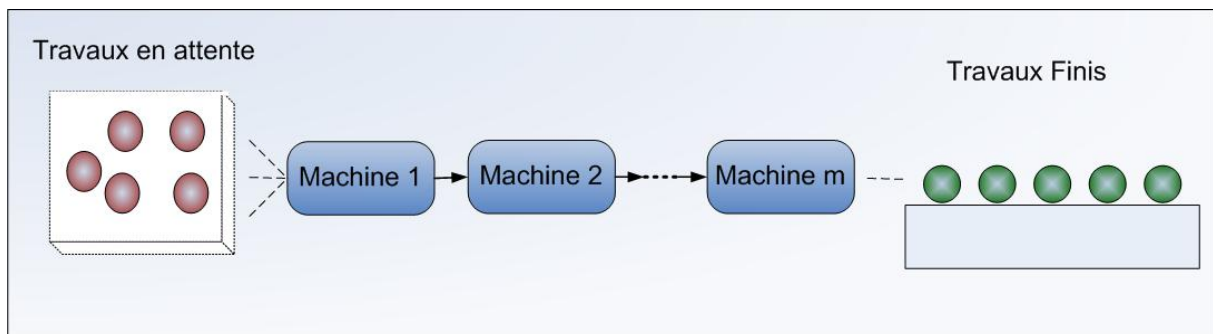
**Figure 3** – Diagramme de cas d'utilisation "Chercher des solutions optimales à un problème du flowshop"

# 3

## État de l'art

### 1 Le problème du flowshop

Le problème du flowshop consiste à trouver, pour un ensemble  $M = \{1, \dots, m\}$  de  $m$  machines disposées en série et un ensemble  $N = \{1, \dots, n\}$  de  $n$  travaux à assigner aux machines, l'ordre optimal des travaux pour minimiser le temps de traitement de tous les travaux par les machines.[2]



**Figure 1** – Exemple de flowshop à 5 travaux sur  $m$  machines (source Wikipedia)

On peut représenter l'ordre des travaux à faire passer dans les machines par un tableau que l'on appelle séquence, ou plus précisément "permutation de travaux" (figure 2). Une séquence de travaux est correcte si elle ne contient pas deux fois le même job et si elle ne comporte pas de cases vides, d'où le terme "permutation".

|    |   |   |   |   |   |   |   |   |   |
|----|---|---|---|---|---|---|---|---|---|
| 10 | 4 | 5 | 3 | 2 | 6 | 9 | 8 | 7 | 1 |
|----|---|---|---|---|---|---|---|---|---|

**Figure 2** – Exemple d'une séquence correcte de 10 travaux

Sur cette séquence, le job 10 sera le premier à rentrer dans la machine 1, puis quand il passera à la machine 2, le job 4 rentrera dans la machine 1, etc. Chaque job contenu dans une séquence de travaux passe un temps bien défini sur chaque machine et dispose ainsi d'une série de coefficients  $P_{ki}$  qui correspondent au temps pour traiter le job  $i$  sur la machine  $k$ .

Le problème est soumis à trois types de contraintes :

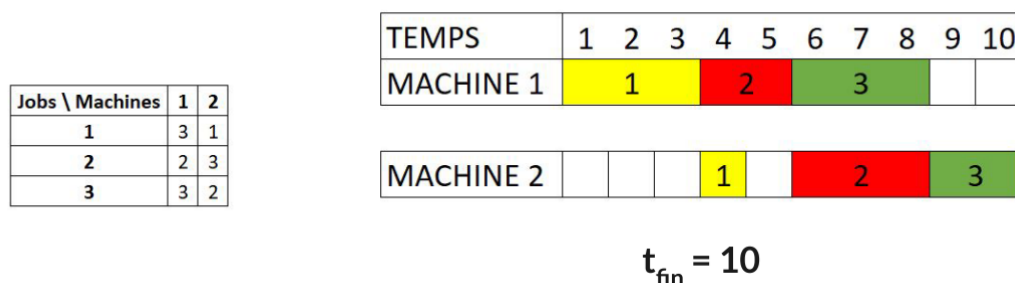
- Contrainte d'itinéraire : les travaux doivent tous suivre le même chemin en suivant un ordre de traitement. Dans le cas du flowshop à deux machines, les travaux doivent ainsi passer par la machine 1 avant de passer dans la machine 2. Dans le cas du flowshop à plus de deux machines, les travaux ne peuvent pas sauter de machines.
- Contrainte de ressource : les machines ne peuvent traiter qu'un job à la fois.
- Contrainte de permutation : chaque job n'est traité qu'une fois, ainsi il n'est présent qu'une fois dans la séquence de travaux.

Pour résoudre un problème d'ordonnancement, on cherche à optimiser une fonction appelée "fonction objectif". Suivant le problème, cette fonction doit être soit minimisée, soit maximisée pour parvenir à la solution optimale. Dans le cadre du problème du flowshop, cette fonction correspond à la minimisation de la somme des dates de fin de traitements des travaux sur la machine  $m$  notée  $\sum_{j=1}^n C_{mj}$  avec  $C_{ki}$  correspondant à la date de fin de traitement du job  $i$  sur la machine  $k$ . Le but est donc de trouver la séquence de travaux qui minimise le temps de traitement de chaque job sur l'ensemble des  $m$  machines.

Lorsqu'on tente de résoudre le problème sur plus de 2 machines, ce problème fait partie de la catégorie des problèmes NP-complet, qui est une catégorie de problèmes qu'on ne peut pas résoudre avec les méthodes de programmation mathématiques actuelles, en temps polynomial. La solution optimale n'est donc trouvable qu'avec un algorithme en temps exponentiel, par exemple avec une modélisation en programmation quadratique. Cette complexité de calcul en temps exponentiel donne des temps de résolution de l'ordre de :

- 10 secondes pour 10 travaux
- 2 minutes pour 100 travaux
- 3 semaines pour 1000 travaux
- ...

On tend donc très rapidement vers des temps de résolution infinis pour des grandes séquences de travaux.



**Figure 3** – Exemple d'une résolution du problème de flowshop à 2 machines

Pour les besoins du PRD, nous nous limiterons à un problème de flowshop à deux machines, on peut donc fixer  $m = 2$ . Prenons une variable de décision  $x_{ij}$  qui prend la valeur 1 si le job  $i$  est dans la position  $j$ , et la valeur 0 sinon. Nous pouvons ensuite formuler mathématiquement le

problème de la manière suivante avec la fonction objectif :

$$f_{objectif} = \min \sum_{j=1}^n C_{2j} \quad (1)$$

soumise aux contraintes :

$$\min \sum_{i=1}^n x_{ij} = 1 \quad \forall j \in \{1, \dots, n\} \quad (2)$$

$$\min \sum_{j=1}^n x_{ij} = 1 \quad \forall i \in \{1, \dots, n\} \quad (3)$$

$$C_{11} = \sum_{i=1}^n p_{1i} x_{i1} \quad (4)$$

$$C_{21} = C_{11} + \sum_{i=1}^n p_{2i} x_{i1} \quad (5)$$

$$C_{1j} = C_{1,j-1} + \sum_{i=1}^n p_{1i} x_{ij} \quad \forall j \in \{2, \dots, n\} \quad (6)$$

$$C_{2j} \geq C_{1j} + \sum_{i=1}^n p_{2i} x_{ij} \quad \forall j \in \{2, \dots, n\} \quad (7)$$

$$C_{2j} \geq C_{2,j-1} + \sum_{i=1}^n p_{2i} x_{ij} \quad \forall j \in \{2, \dots, n\} \quad (8)$$

$$x_{ij} \in \{0, 1\} \quad (9)$$

Explication des contraintes :

- Les contraintes (2) et (3) définissent que chaque position de la séquence est attribuée à un job et qu'un job ne peut être présent qu'une seule fois dans la séquence.
- Les contraintes (4) et (5) définissent la date de fin de traitement du premier travail pour les deux machines.
- La contrainte (6) indique que chaque job peut passer dans la machine 1 uniquement après que l'opération sur le job qui le précédait sur la même machine soit terminée.
- La contrainte (7) interdit pour chaque job le passage de celui-ci dans la machine 2 avant que l'opération précédente du job sur la machine 1 ne soit terminée.
- La contrainte (8) indique que chaque job peut passer dans la machine 2 uniquement après que l'opération sur le job qui le précédait sur la même machine soit terminée.

## 2 Matheuristique

Pour résoudre le problème du flowshop, on peut citer deux grandes catégories de méthodes :

- La programmation mathématique : efficace pour trouver une solution optimale mais lente, notamment pour les grandes instances.
- Les heuristiques : plus rapides, elles permettent de se déplacer intelligemment dans l'espace des solutions. Il en existe deux types : les heuristiques de construction, qui construisent itérativement une solution, et les heuristiques d'exploration, qui, à partir d'une solution donnée, cherchent un optimum local en explorant l'espace des solutions.

Dans cette partie, je vais présenter une méthode alternative pour résoudre le problème du flowshop proposée par Federico Della Croce, Andrea Grosso & Fabio Salassa [3]. Cette méthode est nommée matheuristique car elle repose sur l'utilisation d'une méthode mathématique et l'utilisation d'une heuristique dans un même algorithme.

Le but de cette matheuristique est de combiner l'efficacité d'une résolution exacte et la rapidité des heuristiques. L'idée de cet algorithme est donc de générer une séquence de départ avec une heuristique, ici l'algorithme RBS, puis d'améliorer cette séquence avec des optimisations locales itératives. Pour effectuer les optimisations locales, l'algorithme choisit une fenêtre à optimiser dans la séquence puis lance une méthode de résolution mathématique pour tenter d'optimiser cette fenêtre.

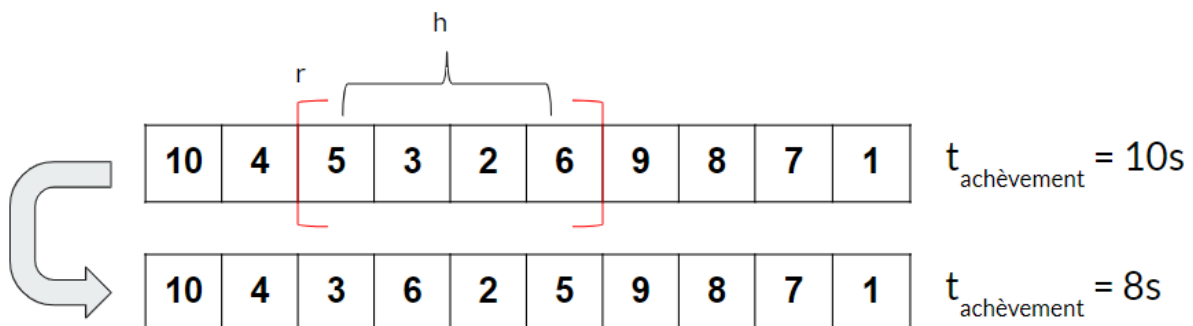


Figure 4 – Exemple d'optimisation locale d'une séquence

avec :

- $r$  : La position de la fenêtre à optimiser
- $h$  : La taille de la fenêtre à optimiser

Dans cet exemple, la taille de la fenêtre choisie est à la position 3 de la séquence ( $r = 3$ ) et a une taille de 4 ( $h = 4$ ). Le paramètre  $h$  est fixé au début de la matheuristique tandis que tous les  $r$  possibles sont testés, la matheuristique tente donc de minimiser la fonction objectif en faisant "glisser" la fenêtre de gauche à droite sur la séquence. La taille de la fenêtre est donc le paramètre clé à déterminer pour que la matheuristique soit efficace : une taille de fenêtre trop petite conduit à une mauvaise exploration de l'espace des solutions et une taille de fenêtre trop grande rend l'exploration très lente.

En générant de multiples petits sous-problèmes à résoudre avec des méthodes mathématiques, on s'affranchit de la complexité à effectuer une méthode mathématique sur une grande séquence. Dans le domaine de l'algorithmique, cette technique est appelée diviser pour régner.

### 3 Méthode Lasso

#### 3.1 Principe de la méthode

La méthode du Lasso, acronyme anglais de *Least Absolute Shrinkage and Selection Operator* est une méthode de régression linéaire qui permet, à partir d'un tableau "individus x variables" en entrée, de construire une équation correspondant à ces données en sélectionnant les variables les plus pertinentes. Cette équation construite correspond au modèle de prédiction. La construction de ce modèle se fait généralement en divisant la base d'apprentissage en trois échantillons :

- L'échantillon d'apprentissage : échantillon sur lequel sera effectué la régression. Cet échantillon servira à construire le modèle de prédiction.
- L'échantillon de validation : échantillon sur lequel seront ajusté les paramètres d'apprentissage du modèle construit avec l'échantillon d'apprentissage.
- L'échantillon de test : échantillon qui servira à évaluer la qualité du modèle de prédiction.

Cette méthode qui consiste à échantillonner la base d'apprentissage est aussi connue sous le nom de cross-validation. La méthode Lasso utilise la méthode des moindres carrés qui consiste à minimiser la distance au carré entre la courbe de prédiction et les points qui représentent les données. À cette minimisation s'ajoute une fonction de pénalité spécifique à la méthode Lasso, appelée dans la littérature scientifique "contrainte L1" [WWW3] :

$$f_{penalite} = \lambda \sum_{j=1}^p |\beta_j| \quad (10)$$

avec :

- $p$  : Le nombre de variables.
- $\beta_j$  : Le coefficient de la variable  $j$ .  $\beta_j \in \mathbb{R}$

Lors de la minimisation, cette fonction (10) permet à la courbe de ne pas trop suivre les données d'apprentissage et ainsi éviter le sur-apprentissage. La particularité de la méthode Lasso est qu'elle permet de faire de la sélection de variables, et peut ainsi entièrement exclure une variable dans la modélisation si celle-ci n'est pas cohérente par rapport aux autres. C'est cette fonction de pénalité, spécifique au Lasso, qui permet de sélectionner les variables les plus pertinentes pour l'apprentissage. Son impact est contrôlé par le coefficient  $\lambda$ , paramétrable par l'utilisateur.

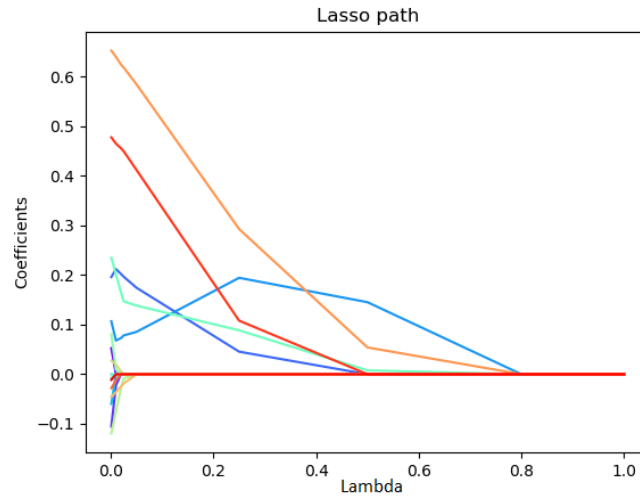
La sortie de l'algorithme Lasso est donc un vecteur  $\hat{\beta}$  contenant la liste des coefficients qui définissent l'équation linéaire du modèle de prédiction. Mathématiquement, on peut formuler cette méthode ainsi :

$$\hat{\beta} = \min_{\beta_1, \dots, \beta_p} \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \left( \sum_{j=1}^p |\beta_j| + \beta_0 \right) \quad (11)$$

avec :

- $n$  : Le nombre d'individus.
- $p$  : Le nombre de variables.
- $\hat{\beta}$  : Le vecteur contenant les coefficients du modèle de prédiction.  $\hat{\beta} \in \mathbb{R}^{p+1}$
- $\beta_j$  : Le coefficient de la variable  $j$ .  $\beta_j \in \mathbb{R}$
- $y_i$  : La variable cible correspondant à l'individu  $i$ .  $y_i \in \mathbb{R}$
- $x_{ij}$  : La variable de type  $j$  correspondant à l'individu  $i$ .  $x_{ij} \in \mathbb{R}$

Il est possible de visualiser l'impact du  $\lambda$  en traçant un graphique nommé « Lasso path », qui représente les valeurs des coefficients de variables en fonction de la valeur de  $\lambda$  : [WWW2]

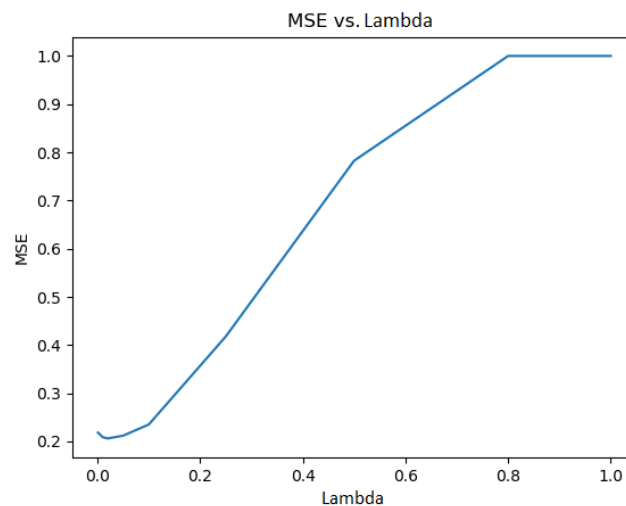


**Figure 5** – Lasso path (17 variables)

On remarque alors que plus  $\lambda$  est élevé, plus les coefficients tendent vers 0. Une valeur de  $\lambda$  trop élevée contraint donc trop le modèle et celui-ci ne peut plus faire de prédictions. Au contraire, lorsque  $\lambda = 0$ , les coefficients sont ceux d'une régression linéaire classique : ces coefficients sont donc adaptés à l'échantillon d'apprentissage mais pas forcément à l'échantillon de test. [WWW1] Il est donc nécessaire de tester plusieurs valeurs de  $\lambda$  jusqu'à trouver celle qui permet de minimiser un critère d'erreur entre les prédictions du modèle et l'échantillon de test. On peut définir le critère d'erreur en utilisant la fonction erreur quadratique moyenne (MSE) :

$$\text{MSE}_\lambda = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (12)$$

Le calcul du MSE pour chaque valeur de  $\lambda$  permet ensuite de tracer un graphique de l'erreur de prédiction en fonction de  $\lambda$  : [WWW2]

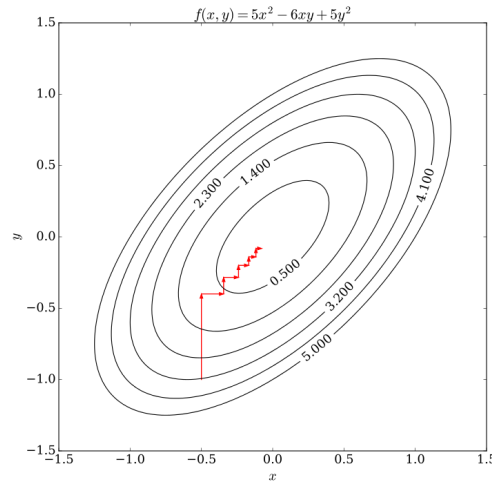


**Figure 6** – Erreur de prédiction en fonction de lambda

Il suffit ensuite de prendre  $\lambda$  correspondant à la valeur minimale du MSE pour générer le modèle de prédiction optimal (ici  $\lambda \approx 0.01$ ).

### 3.2 Algorithmes de résolution

Il existe plusieurs algorithmes qui permettent de minimiser la fonction (11) afin de trouver le vecteur  $\hat{\beta}$  contenant les coefficients du modèle de prédiction. Cette fonction est cependant complexe à minimiser car la fonction de pénalité (10) n'est pas différentiable à cause de la nature de la fonction  $f(x) = |x|$  qui n'est pas dérivable en  $x = 0$ . Ce caractère non différentiable de la fonction de pénalité fait que la fonction (11) ne peut pas être minimisée avec des méthodes de programmation linéaire et demande donc plus de puissance de calcul. [4]



**Figure 7** – Itérations de l'algorithme de descente par coordonnée sur une fonction définie dans  $\mathbb{R}^2$   
(source Wikipédia)

Un des algorithmes les plus couramment utilisés pour effectuer une régression Lasso est l'algorithme de descente par coordonnée (Figure 7). Le principe de cet algorithme consiste à se déplacer dans l'espace des solutions  $\mathbb{R}^n$  en minimisant la fonction sur une dimension à la fois de manière itérative jusqu'à converger vers un extremum. Si on applique l'algorithme à la fonction Lasso (11), cela revient à modifier un à un chaque coefficient  $\beta_j$  en cherchant à chaque fois à minimiser le résultat de la fonction. L'algorithme de descente par coordonnées est aujourd'hui l'algorithme utilisé par défaut dans la bibliothèque Scikit-Learn pour générer un modèle de prédiction avec la méthode Lasso. [WWW5]

D'autres algorithmes existent pour minimiser la fonction Lasso comme l'algorithme LARS, acronyme de *Least Angle Regression* qui est le deuxième algorithme implémenté dans Scikit-Learn pour générer un modèle de prédiction Lasso.

### 3.3 Comparaison avec une régression linéaire classique

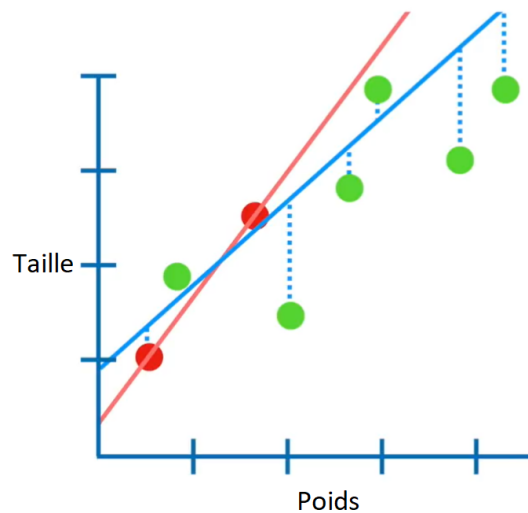
Pour comparer l'intérêt d'une régression Lasso par rapport à une régression linéaire classique, nous utiliserons le schéma suivant qui représente un modèle de prédiction permettant de prédire la taille d'une personne en fonction de son poids.

Sur ce schéma :

- Les points rouges correspondent à l'échantillon d'apprentissage.
- Les points verts correspondent à l'échantillon de test.
- La droite rouge correspond à la régression linéaire sur l'ensemble des données d'apprentissage.



— La droite bleue correspondent à la droite de régression Lasso.



**Figure 8** – Comparaison régression classique et Lasso [WWW4]

On peut constater sur le schéma que la régression linéaire classique a tendance à très bien suivre les données d'apprentissage car la seule contrainte est la minimisation des distances au carré entre la droite de prédiction et les données d'apprentissage. Cependant elle est très éloignée des données de test et elle est donc peu efficace pour prédire celles-ci, car les données ont une grande variance.

La régression Lasso, grâce à l'ajout de la fonction de pénalité  $f_{penalite}$ , minimise la pente de la courbe en plus de la distance au carré entre la droite et les données : cette caractéristique permet à la méthode Lasso de créer une droite moins verticale et donc un modèle de prédiction moins contraint qui s'adapte mieux aux données de test.

# 4

## Analyse et conception

### 1 Extraction des caractéristiques

Afin de construire un modèle de prédiction, il est nécessaire de caractériser chaque séquence de travaux avant de la passer en paramètre de l'algorithme d'apprentissage. Pour caractériser les séquences, il faut définir et extraire des caractéristiques de celles-ci de manière à construire des variables qui serviront à l'apprentissage du modèle de prédiction. L'extraction des caractéristiques va donc permettre de passer d'une séquence de travaux à un vecteur de caractéristiques correspondant à la séquence.

En entrée de l'algorithme, on aura une séquence de travaux :

|            | Permutation de $n$ travaux |   |    |     |     |
|------------|----------------------------|---|----|-----|-----|
| Séquence 1 | 3                          | 2 | 10 | ... | $n$ |

**Figure 1** – Séquences non caractérisée (permutation de travaux)

En sortie, nous aurons la liste des caractéristiques de la séquence.

|            | Caractéristique $c_1$ | ... | Caractéristique $c_k$ | Variable cible (% amélioration) |
|------------|-----------------------|-----|-----------------------|---------------------------------|
| Séquence 1 | 8.6                   | ... | $k_{seq_1}$           | 0.2                             |

**Figure 2** – Séquences caractérisée (vecteur de caractéristiques)

Ces caractéristiques seront organisées en plusieurs familles :

- Famille 1 : Données temporelles sur la séquence.
- Famille 2 : Distribution des travaux n'étant pas dans l'ordre croissant des durées opératoires sur la seconde machine dans la fenêtre d'optimisation avec 3 quantiles.
- Famille 3 : Distribution des temps morts sur la machine 2 dans la fenêtre d'optimisation avec 3 quantiles.
- Famille 4 : Distribution des ratios des durées opératoires pour chaque job avec 3 quantiles.

— ....

Pour chaque caractéristique de chaque famille [5], un algorithme permettra de calculer les caractéristiques correspondantes. Il faudra ensuite étudier l'apport de chaque famille sur la qualité du modèle de prédiction.

## 2 Génération et structure de la base d'apprentissage

La base d'apprentissage déjà existante a été construite sous cette forme : [1]

| n   | r   | h   | %Imp | %ImpSolInit | BestS | S   |
|-----|-----|-----|------|-------------|-------|-----|
| ... | ... | ... | ...  | ...         | ...   | ... |

Le fichier de cette base est composée de blocs de 6 lignes et chaque bloc correspond à une amélioration itérative d'une solution initiale  $S_0$  différente.

Une ligne correspond à une itération, donc à une amélioration de la solution  $S_i$ , et le nombre d'itérations a été fixé à 6 par solution donc  $0 < i < 6$ .

**n** : Nombre de travaux de l'instance.

**r** : Position de départ de la fenêtre.

**h** : Taille de la fenêtre (comprise en 4 et 16 dans la base).

**%Imp** : Pourcentage d'amélioration sur l'itération.

**%ImpSolInit** : Pourcentage d'amélioration par rapport à la solution initiale  $S_0$ .

**BestS** : Solution optimisée avec la fenêtre optimale sur  $S_i$ . Correspond à  $S_{i+1}$ .

**S** : Solution à optimiser. Correspond à  $S_i$ .

Cette base a été construite pour un apprentissage orienté deep-learning, et comme elle est composée uniquement de solutions améliorantes, celle-ci manque de diversité. On ne pourra donc pas réutiliser son contenu, notamment les séquences déjà calculées, mais on pourra s'inspirer de sa structure pour créer celle de la future base d'apprentissage.

Pour la future base d'apprentissage, nous aurons besoin de stocker une liste de vecteurs de caractéristiques correspondants à chaque séquence, ainsi on aura une structure de la forme :

| n   | r   | h   | %Imp | %ImpSolInit | S   | Caractéristique $c_1$ | ... | Caractéristique $c_k$ |
|-----|-----|-----|------|-------------|-----|-----------------------|-----|-----------------------|
| ... | ... | ... | ...  | ...         | ... | ...                   | ... | ...                   |

Figure 3 – Structure de la base d'apprentissage

**n** : Nombre de travaux de l'instance.

**r** : Position de départ de la fenêtre.

**h** : Taille de la fenêtre (comprise en 4 et 16 dans la base).

**S** : Séquence de travaux.

**%Imp** : Pourcentage d'amélioration sur l'itération.

**%ImpSolInit** : Pourcentage d'amélioration par rapport à la solution initiale  $S_0$ .

**Caractéristiques  $c_1 \dots c_k$**  : Vecteur de caractéristiques correspondant à la solution S.

Cette base d'apprentissage sera construite en testant toutes les fenêtres d'optimisation possibles avec une taille comprise entre 4 et 16 sur une séquence générée par une heuristique (ici l'algorithme RBS). On retiendra dans la base d'apprentissage les 5 solutions les plus amélioratrices au

cours de l'optimisation, et pour ajouter de la diversité, 15 autres solutions (amélioratrices ou non) seront ajoutées aléatoirement dans la base d'apprentissage.

Cette base d'apprentissage sera ensuite exportée sous la forme d'un fichier CSV qui contiendra la liste des caractéristiques ainsi que le pourcentage d'amélioration.

### 3 Apprentissage du modèle de prédiction

Afin d'apprendre un modèle de prédiction, on utilisera la régression Lasso (cf **état de l'art**). Nous pourrons ainsi identifier les caractéristiques des séquences qui sont les plus impliquées dans la création du modèle.

Pour identifier les caractéristiques qui seront impliquées dans la création du modèle de prédiction, il faudra prendre dans la liste des coefficients tous les coefficients qui n'ont pas une valeur de 0. Si un coefficient d'une variable vaut 0, cela veut dire que cette variable n'a pas été utilisée dans la construction du modèle de prédiction.

### 4 Amélioration de la matheuristique

Il faudra également apporter des modifications à la matheuristique déjà existante pour y ajouter la fonctionnalité de prédiction. Cette matheuristique suit le fonctionnement suivant :

---

#### Algorithme 1 : Matheuristique existante

---

```

 $\bar{x} \leftarrow$  solution to minimize ;
repeat
   $improved \leftarrow false$ ;
   $h \leftarrow 12$ ;
   $n \leftarrow$  number of jobs;
  repeat
    Pick  $r \in \{1, \dots, n - h + 1\}$  randomly;
     $\hat{x} \leftarrow$  minimize  $\bar{x}$  in  $(r; h)$ ;
    if  $f(\bar{x}) > f(\hat{x})$  then
       $\bar{x} \leftarrow \hat{x}$ ;
       $improved \leftarrow true$ ;
  until  $improved$  or all  $r$  values have been tried;
until not improved or time limit expired;

```

---

Cette approche est très coûteuse en temps puisqu'elle tente d'optimiser une solution en appelant une méthode de résolution exacte sur un grand nombre de fenêtres. De plus, avec cette méthode, l'espace de recherche est grandement limité car la taille de la fenêtre est fixée à  $h = 12$ .

Pour réduire le temps d'exécution, nous allons donc utiliser le modèle de prédiction en ajoutant un test dans la matheuristique qui vérifie si l'optimisation sur la fenêtre choisie permet d'obtenir une amélioration sur la fonction objectif de la solution. La matheuristique hybridée aura donc le fonctionnement suivant :

---

**Algorithme 2 : Matheuristique hybridée avec le modèle de prédiction**

---

```

 $\bar{x} \leftarrow$  solution to minimize ;
repeat
   $improved \leftarrow false$ ;
   $n \leftarrow$  number of jobs;
   $threshold \leftarrow$  minimum improvement percentage to minimize the solution;
  repeat
    Pick  $h \in \{4, \dots, 16\}$  randomly;
    Pick  $r \in \{1, \dots, n - h + 1\}$  randomly;
    if predict improvement of  $\bar{x}$  in  $(r;h) > threshold$  then
       $\hat{x} \leftarrow$  minimize  $\bar{x}$  in  $(r;h)$ ;
      if  $f(\bar{x}) > f(\hat{x})$  then
         $\bar{x} \leftarrow \hat{x}$ ;
         $improved \leftarrow true$ ;
  until improved or all  $(r;h)$  pairs have been tried;
until not improved or time limit expired;

```

---

# 5

## Mise en oeuvre

### 1 Caractéristiques choisies pour la base d'apprentissage

Cette partie détaille toutes les caractéristiques qui ont été choisies pour construire la base d'apprentissage. Les notations qui seront utilisées pour décrire les caractéristiques sont les suivantes :

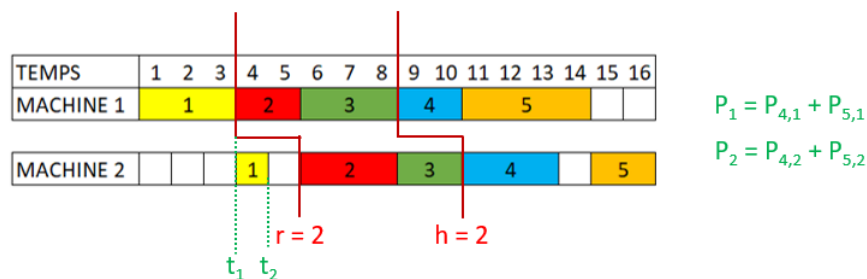


Figure 1 – Schéma explicatif

$n$  = nombre de jobs dans la séquence (1)

$r$  = position de début de la fenêtre (2)

$h$  = taille de la fenêtre (nombre de jobs dans la fenêtre) (3)

$t_1$  = date de début au plus tôt des travaux dans  $[r;r+h]$  sur la machine 1 (4)

$$t_2 = \text{date de début au plus tôt des travaux dans } [r;r+h] \text{ sur la machine 2} \quad (5)$$

$$\mathcal{P}_1 = \sum_{l=r+h+1}^n \mathcal{P}_{l,1} \quad (6)$$

$$\mathcal{P}_2 = \sum_{l=r+h+1}^n \mathcal{P}_{l,2} \quad (7)$$

### 1.1 Famille 1 : Données temporelles sur la séquence

- $\mathcal{C}_1 = t_1$
- $\mathcal{C}_2 = t_2$
- $\mathcal{C}_3 = \mathcal{P}_1$
- $\mathcal{C}_4 = \mathcal{P}_2$

Ces caractéristiques sont normées avec la taille de la séquence.

### 1.2 Famille 2 : Distribution des travaux n'étant pas dans l'ordre croissant des durées opératoires sur la seconde machine dans la fenêtre d'optimisation avec 3 quantiles

Les 3 quantiles sont calculés de la façon suivante :

- $x$  : nombre de travaux consécutifs ne respectant pas l'ordonnancement SPT sur la machine 2 dans  $[r;r+h]$ .
- $f(x)$  : nombre de fois où on a un bloc de  $x$  travaux ne respectant pas l'ordonnancement SPT sur la machine 2 dans  $[r;r+h]$ .
- $\mathcal{P}(X \leq x)$  : probabilité d'avoir des blocs d'une taille au plus  $x$ .
- $x_1$  : valeur de  $x$  telle que  $\mathcal{P}(X \leq x_1) \leq \frac{1}{3}$  et  $x_1$  est maximum.
- $x_2$  : valeur de  $x$  telle que  $\mathcal{P}(X \leq x_2) \leq \frac{2}{3}$  et  $x_2$  est maximum.

Voici un exemple de calcul :

|                         |               |               |               |               |               |               |
|-------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| $x$                     | 0             | 1             | 2             | 3             | 4             | $5=(h-r)$     |
| $f(x)$                  | 0             | 2             | 0             | 1             | 2             | 3             |
| $\mathcal{P}(X \leq x)$ | $\frac{0}{N}$ | $\frac{2}{N}$ | $\frac{2}{N}$ | $\frac{3}{8}$ | $\frac{5}{8}$ | $\frac{8}{8}$ |

Figure 2 – Exemple de calcul de la famille 2

Les caractéristiques retenues pour cette famille sont les suivantes :

- $\mathcal{C}_5 = x_1 = 2$
- $\mathcal{C}_6 = x_2 = 4$

### 1.3 Famille 3 : Distribution des temps morts sur la machine 2 dans la fenêtre d'optimisation avec 3 quantiles

On recense les durées d'inactivité sur la machine 2 entre deux travaux et on calcule les 3 quantiles sur leur distribution. Ces quantiles sont calculés après normalisation par rapport au temps de traitement maximum dans la fenêtre sur la machine 1,  $P_{max} = \max_{i \in s^{r,h}}(P_{i,1})$ .

Les 3 quantiles sont calculés de la façon suivante :

- $x$  : liste de taille de temps mort.
- $f(x)$  : nombre de temps morts de taille  $x$ .
- $\mathcal{P}(X \leq x)$  : probabilité d'avoir un temps mort d'une taille au plus  $x$ .
- $x_1$  : valeur de  $x$  telle que  $\mathcal{P}(X \leq x_1) \leq \frac{1}{3}$  et  $x_1$  est maximum.
- $x_2$  : valeur de  $x$  telle que  $\mathcal{P}(X \leq x_2) \leq \frac{2}{3}$  et  $x_2$  est maximum.

Voici un exemple de calcul :

|                         |               |               |               |      |
|-------------------------|---------------|---------------|---------------|------|
| $x$                     | 0.55          | 0.63          | 0.87          | 0.85 |
| $f(x)$                  | 1             | 1             | 1             | 1    |
| $\mathcal{P}(X \leq x)$ | $\frac{1}{4}$ | $\frac{1}{2}$ | $\frac{3}{4}$ | 1    |

Figure 3 – Exemple de calcul de la famille 3

Les caractéristiques retenues pour cette famille sont les suivantes :

- $\mathcal{C}_7 = x_1 = 0.55$
- $\mathcal{C}_8 = x_2 = 0.63$

### 1.4 Famille 4 : Distribution des ratios des durées opératoires pour chaque travail avec 3 quantiles

Pour chaque travail présent dans la fenêtre, on calcule le ratio entre son temps de traitement sur la machine 1 et son temps de traitement sur la machine 2. Cela revient à calculer pour chaque travail,  $r_i = \frac{P_{i,1}}{P_{i,2}}$ .

Les 3 quantiles sont calculés de la façon suivante :

- $x$  : valeur d'un ratio  $r_i$ .
- $f(x)$  : nombre de fois où le ratio  $x$  apparaît.
- $\mathcal{P}(X \leq x)$  : probabilité d'avoir un ratio au plus égal  $x$ .
- $x_1$  : valeur de  $x$  telle que  $\mathcal{P}(X \leq x_1) \leq \frac{1}{3}$  et  $x_1$  est maximum.
- $x_2$  : valeur de  $x$  telle que  $\mathcal{P}(X \leq x_2) \leq \frac{2}{3}$  et  $x_2$  est maximum.

Les caractéristiques retenues pour cette famille sont les suivantes :

- $\mathcal{C}_9 = x_1$
- $\mathcal{C}_{10} = x_2$

### 1.5 Famille 5 : Borne inférieure

On optimise la séquence dans la fenêtre  $r,h$  et on récupère la borne inférieure donnée par CPLEX. La caractéristique correspond au ratio entre la borne inférieure trouvée et la fonction objectif de la séquence à optimiser.



$$— C_{11} = \frac{LowerBound}{\sum_{j=1}^n C_j}$$

### 1.6 Famille 6 : Distribution des temps morts sur la machine 2 à droite de la fenêtre d'optimisation avec 3 quantiles

Ces trois quantiles sont calculés de la même manière que dans la famille 3, mais en considérant l'intervalle  $h+1, n$  au lieu de  $r, h$ . Ils sont donc calculés après normalisation par rapport au temps de traitement maximum à droite de la fenêtre sur la machine 1,  $P_{max} = \max_{i \in S^{h+1, n}}(P_{i,1})$ .

$$— C_{121} = x_1$$

$$— C_{121} = x_2$$

### 1.7 Famille 7 : Déciles de la famille 2

Calculs identiques à la famille 2, mais calcul des déciles au lieu des 3 quantiles. Neuf caractéristiques (les neuf déciles) peuvent donc être extraites.

$$— C_{13} = x_1$$

$$— C_{14} = x_2$$

$$— ...$$

$$— C_{21} = x_9$$

### 1.8 Famille 8 : Déciles de la famille 3

Calculs identiques à la famille 3, mais calcul des déciles au lieu des 3 quantiles. Neuf caractéristiques (les neuf déciles) peuvent donc être extraites.

$$— C_{22} = x_1$$

$$— C_{23} = x_2$$

$$— ...$$

$$— C_{30} = x_9$$

### 1.9 Famille 9 : Déciles de la famille 4

Calculs identiques à la famille 4, mais calcul des déciles au lieu des 3 quantiles. Neuf caractéristiques (les neuf déciles) peuvent donc être extraites.

$$— C_{31} = x_1$$

$$— C_{32} = x_2$$

$$— ...$$

$$— C_{39} = x_9$$

### 1.10 Famille 10 : Déciles de la famille 6

Calculs identiques à la famille 6, mais calcul des déciles au lieu des 3 quantiles. Neuf caractéristiques (les neuf déciles) peuvent donc être extraites.

$$— C_{40} = x_1$$

$$— C_{41} = x_2$$

$$— ...$$

$$— C_{48} = x_9$$

### 1.11 Famille 11-1 : Critères min, max et écarts types pour les familles 2, 3, 4 et 6

Pour les familles 2, 3, 4 et 6, on calcule le minimum, le maximum et l'écart type des valeurs de  $x$ .

- $C_{49} = \max(x_{F2})$
- $C_{50} = \min(x_{F2})$
- $C_{51} = \text{ecarttype}(x_{F2})$
- ...
- $C_{58} = \max(x_{F6})$
- $C_{59} = \min(x_{F6})$
- $C_{60} = \text{ecarttype}(x_{F6})$

### 1.12 Famille 11-2 : Durées opératoires des travaux dans la fenêtre

Cette famille regroupe différentes caractéristiques concernant les durées opératoires des travaux sur chaque machine dans la fenêtre d'optimisation.

- $C_{61} = \min_{i \in s^{r,h}}(P_{i,1})$
- $C_{62} = \max_{i \in s^{r,h}}(P_{i,1})$
- $C_{63} = \text{ecarttype}_{i \in s^{r,h}}(P_{i,1})$
- $C_{64} = \min_{i \in s^{r,h}}(P_{i,2})$
- $C_{65} = \max_{i \in s^{r,h}}(P_{i,2})$
- $C_{66} = \text{ecarttype}_{i \in s^{r,h}}(P_{i,2})$

### 1.13 Famille 12 : Caractéristiques SRPT

Ces caractéristiques permettent de quantifier la relation entre l'ordonnancement de la séquence et la règle d'ordonnancement SRPT (Shortest Remaining Processing Time, Plus Petit Temps de Traitement en Premier). Si on considère, pour chaque travail  $i$ ,  $D_i$  la différence entre le temps d'achèvement sur la machine 2 et le temps d'achèvement issu de l'ordonnancement SRPT, tel que  $D_i = C_s(i, 2) - C_{\text{SRPT}}(i)$ , on peut calculer les caractéristiques suivantes :

- $C_{67} = \frac{\sum_{i=r}^{r+h} D_i}{h}$ , moyenne des  $D_i$  sur la fenêtre d'optimisation.
- $C_{68} = \frac{\max_{i \in s^{r,h}} D_i}{h}$ , maximum des  $D_i$  sur la fenêtre d'optimisation.
- $C_{69} = \frac{\sum_{i=0}^n D_i}{n}$ , moyenne des  $D_i$  sur toute la séquence.
- $C_{70} = \frac{\max_{i \in 0,n} D_i}{n}$ , maximum des  $D_i$  sur toute la séquence.
- $C_{71} = \frac{\sum_{i=0}^{n/5} D_i}{n/5}$ , moyenne des  $D_i$  sur la fenêtre  $[0, \frac{n}{5}]$ .
- $C_{72} = \frac{\max_{i \in 0,n/5} D_i}{n/5}$ , maximum des  $D_i$  sur la fenêtre  $[0, \frac{n}{5}]$ .
- $C_{73} = \frac{\sum_{i=n/5+1}^{2n/5} D_i}{n/5}$ , moyenne des  $D_i$  sur la fenêtre  $[\frac{n}{5} + 1, \frac{2n}{5}]$ .
- $C_{74} = \frac{\max_{i \in n/5+1,2n/5} D_i}{n/5}$ , maximum des  $D_i$  sur la fenêtre  $[\frac{n}{5} + 1, \frac{2n}{5}]$ .
- $C_{75} = \frac{\sum_{i=2n/5+1}^{3n/5} D_i}{n/5}$ , moyenne des  $D_i$  sur la fenêtre  $[\frac{2n}{5} + 1, \frac{3n}{5}]$ .

- $C_{76} = \frac{\max_{i \in [2n/5+1, 3n/5]} D_i}{n/5}$ , maximum des  $D_i$  sur la fenêtre  $[\frac{2n}{5} + 1, \frac{3n}{5}]$ .
- $C_{77} = \frac{\sum_{i=3n/5+1}^{4n/5} D_i}{n/5}$ , moyenne des  $D_i$  sur la fenêtre  $[\frac{3n}{5} + 1, \frac{4n}{5}]$ .
- $C_{78} = \frac{\max_{i \in [3n/5+1, 4n/5]} D_i}{n/5}$ , maximum des  $D_i$  sur la fenêtre  $[\frac{3n}{5} + 1, \frac{4n}{5}]$ .
- $C_{79} = \frac{\sum_{i=4n/5+1}^n D_i}{n/5}$ , moyenne des  $D_i$  sur la fenêtre  $[\frac{4n}{5} + 1, n]$ .
- $C_{80} = \frac{\max_{i \in [4n/5+1, n]} D_i}{n/5}$ , maximum des  $D_i$  sur la fenêtre  $[\frac{4n}{5} + 1, n]$ .

## 2 Développement

Afin de répondre au cahier des charges, quatre composants logiciels ont été créés. Ces composants ont été conçus de manière indépendante pour pouvoir être réutilisables dans d'autres projets. Ces quatre composants sont :

- Générateur d'instance.
- Générateur de bases de test et d'apprentissage.
- Générateur de modèle de prédiction.
- Matheuristiques hybrides : ce composant permet de tester le modèle de prédiction sur différentes matheuristiques.

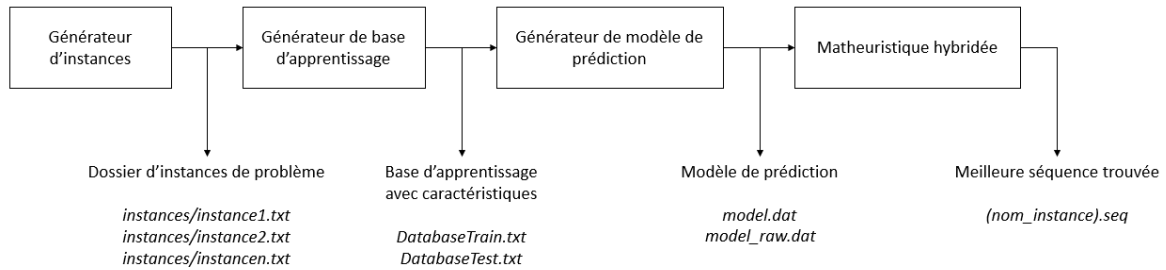


Figure 4 – Interactions entre les différents composants

### 2.1 Générateur d'instances

Ce composant génère des instances de problème de flowshop. Il n'était pas prévu dans les spécifications mais s'est avéré utile pour automatiser la génération d'un grand nombre d'instances.

L'utilisateur passe en entrée le nombre d'instances à générer, ainsi que le nombre de travaux et le nombre de machines dans ces instances. Le composant génère ensuite des temps de traitement pour chaque travail sur chaque machine d'une durée comprise entre 0 et 100.

La sortie est un dossier de fichiers d'instances.

### 2.2 Générateur de bases de test et d'apprentissage

La recherche des différentes caractéristiques à programmer a été une tâche importante dans le développement du projet. L'intégralité des caractéristiques énoncées dans la partie 5.1 a ensuite été programmée dans le générateur de bases de test et d'apprentissage.

### 2.3 Générateur de modèle de prédiction

Le générateur de modèle de prédiction a été programmé, comme prévu, à l'aide de la bibliothèque Scikit-Learn. Cette bibliothèque contient différents modules permettant d'effectuer différents types de régression linéaire. Le module LassoCV a été choisi car il permet de mettre en oeuvre rapidement une méthode Lasso avec cross-validation. Deux fichiers de modèle sont sauvegardés : "model\_raw.dat" qui contient le modèle brut et "model.dat" qui est le modèle appris en enlevant les caractéristiques les moins pertinentes.

|                         | Variables (caractéristiques) | Coefficients | Variances   | Moyenne    |
|-------------------------|------------------------------|--------------|-------------|------------|
| 1                       | C1                           | 0.325141     | 111.992862  | 12.125807  |
| ..                      | ..                           | ..           | ..          | ..         |
| n                       | C80                          | 0.066866     | 3227.472902 | 125.465300 |
| Moyenne variable cible  |                              |              |             |            |
| Variance variable cible |                              |              |             |            |

Figure 5 – Format du fichier du modèle de prédiction

### 2.4 Matheuristiques

Plusieurs matheuristiques ont été développées de façon à tester le modèle de prédiction de plusieurs manières :

- Matheuristique originale hybridée : Il s'agit de la matheuristique originale ( $h=12$  et  $r$  aléatoire) guidée par le modèle de prédiction. Si le modèle prédit une amélioration supérieure à 0, on lance CPLEX, sinon on continue à chercher.
- Matheuristique force brute hybridée : Cette matheuristique teste toutes les valeurs de  $r$  ainsi que des valeurs de  $h$  allant de 4 à 16. Pour chaque couple  $(r,h)$ , le modèle de prédiction est lancé. Si le modèle prédit une amélioration supérieure à 0, on lance CPLEX, sinon on continue à chercher.
- Matheuristique avec prédictions ordonnées (arrêt à 70% de la liste) : À chaque itération de la matheuristique, on prédit les améliorations pour chaque couple  $(r,h)$  et on trie les prédictions dans l'ordre décroissant. On lance ensuite CPLEX sur chaque couple  $(r,h)$  jusqu'à ce que la solution soit améliorée. Lorsque les 70% meilleures prédictions ont été testées et que la solution n'a pas été améliorée, la matheuristique s'arrête.
- Matheuristique avec prédictions ordonnées (arrêt à 10% de la pire valeur de prédiction) : Même principe que la matheuristique précédente. Lorsque la valeur de prédiction correspond à 10% de la pire valeur de prédiction et que la solution n'a pas été améliorée, la matheuristique s'arrête.

## 3 Analyses

### 3.1 Analyse de l'impact des caractéristiques sur l'apprentissage

L'objectif de cette première analyse était de quantifier l'impact des caractéristiques sur l'apprentissage. Le but est de voir, à l'issue de l'apprentissage d'un modèle, quelle est sa qualité et

quelles sont les caractéristiques qui ont le plus contribuées à cette qualité.

Dans un premier temps, il a été choisi d'effectuer l'apprentissage en utilisant l'algorithme RBS, donc de construire une base d'apprentissage en recherchant des améliorations sur des solutions initiales générées par l'algorithme RBS.

Sur les apprentissages 1 à 4, la variable cible est le pourcentage d'amélioration défini par :

$$\%_{imp} = \frac{S - S'}{S} \quad (8)$$

avec :

- $S$  : la fonction objectif de la séquence initiale.
- $S'$  : la fonction objectif de la séquence optimisée.

Pour correspondre à la notation de la bibliothèque Scikit-Learn, la lettre  $\alpha$  sera utilisée à la place de la lettre  $\lambda$  pour les statistiques.

### 3.1.1 Apprentissage 1 : 100 jobs, 2465 individus, RBS

Cette première base a été construite à partir d'instances de 100 jobs et comporte 2465 individus.

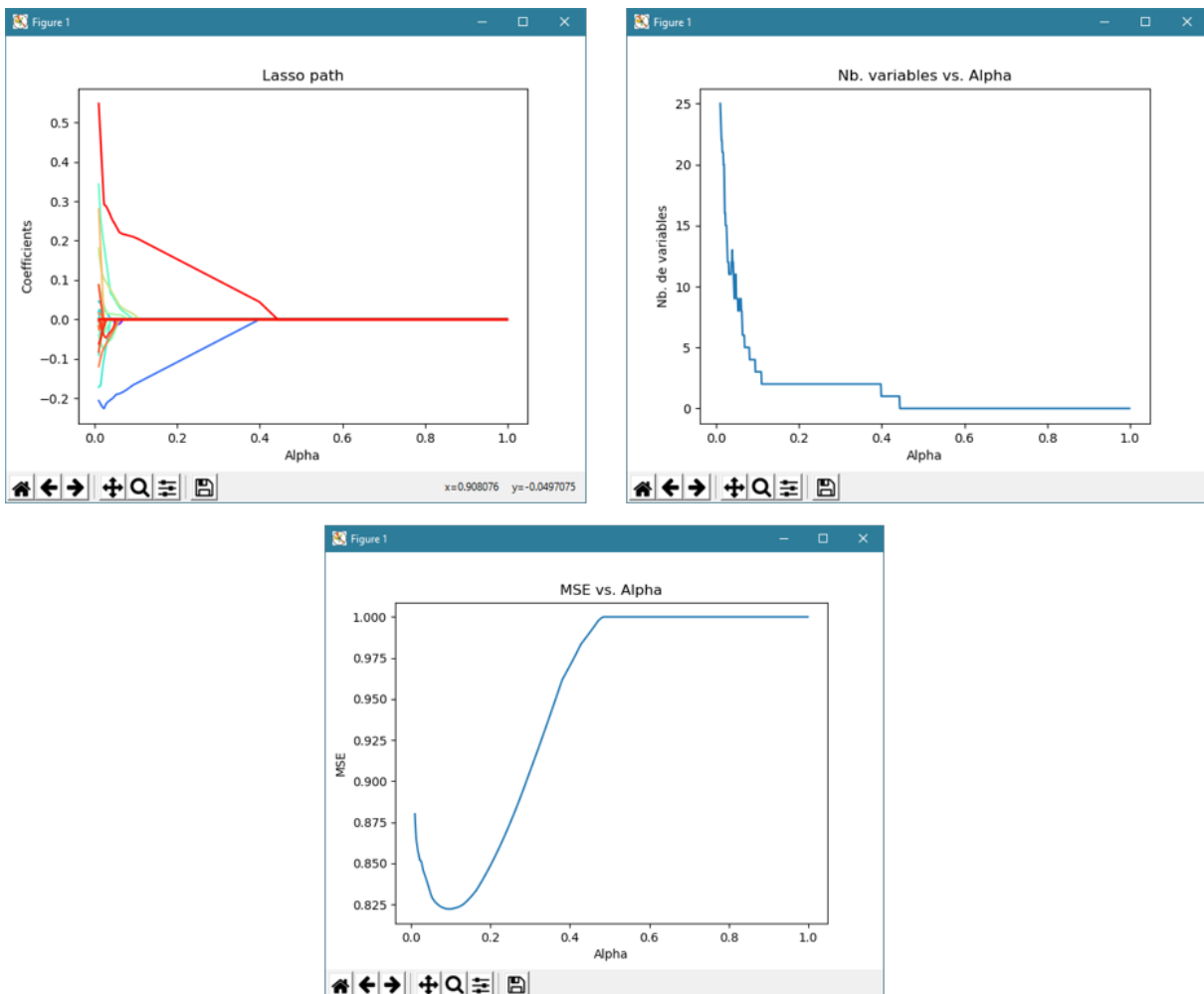


Figure 6 – Apprentissage N°1

On peut voir sur le graphique que l'erreur quadratique moyenne du modèle est très élevée, même pour la valeur de  $\alpha$  optimale (ici l'erreur quadratique moyenne vaut 82.5% pour  $\alpha=0.096$ ). Il ressort également de l'apprentissage que seulement 3 caractéristiques sont utilisées : H, C11 et C44 (respectivement : la largeur de la fenêtre, le ratio sur la borne inférieure, le 5ème décile des temps morts sur la machine 2 à la droite de la fenêtre). Sur le graphique du Lasso path, on peut aussi remarquer que deux caractéristiques dominent largement : H et C11.

Il a été conclu que la base d'apprentissage n'était pas suffisamment grande pour que chaque caractéristique puisse avoir une contribution au modèle.

### 3.1.2 Apprentissage 2 : 100 jobs, 12600 individus, RBS

Ce deuxième apprentissage, en utilisant une base d'apprentissage 5 fois plus grande, a permis de déterminer si la taille de la base permettait d'améliorer significativement l'erreur quadratique moyenne du modèle en utilisant les autres caractéristiques.

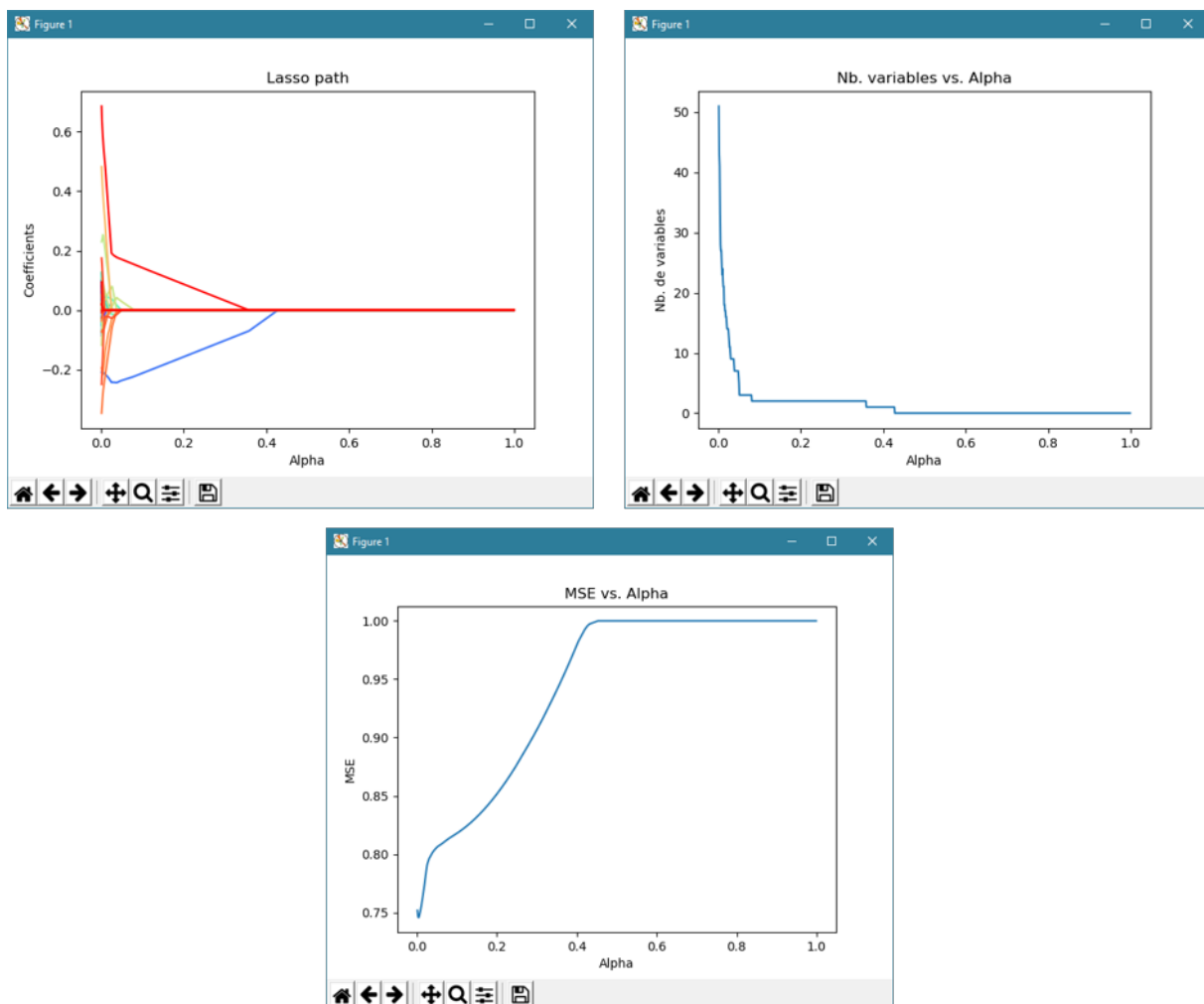


Figure 7 – Apprentissage N°2

On remarque que l'erreur quadratique moyenne pour la valeur de  $\alpha$  optimale est en baisse d'environ 8% (ici l'erreur quadratique moyenne vaut 74.5% pour  $\alpha=0.003$ ). C'est une baisse significative, mais au vu des temps de calculs nécessaires pour générer une telle base, environ 4 jours sur un processeur avec 12 threads, il n'était pas envisageable de multiplier encore la taille de la base d'apprentissage.

35 caractéristiques ont contribué à l'apprentissage de ce modèle (les 5 premières étant H, C51, C58, C43 et C11). Les autres caractéristiques de la base d'apprentissage avaient majoritairement des valeurs nulles ou constantes pour chaque individu et ne permettaient donc pas d'améliorer le modèle.

Pour donner de l'information à ces caractéristiques, il a donc été choisi de supprimer l'algorithme RBS lors de la génération de la base et de le remplacer par une séquence générée aléatoirement dans le but de mieux explorer les phases d'amélioration.

### 3.1.3 Apprentissage 3 : 20 jobs, 4860 individus, sans RBS

Une petite base d'apprentissage de 4860 individus à partir d'instances de 20 jobs a été générée afin de rapidement tester la méthode.

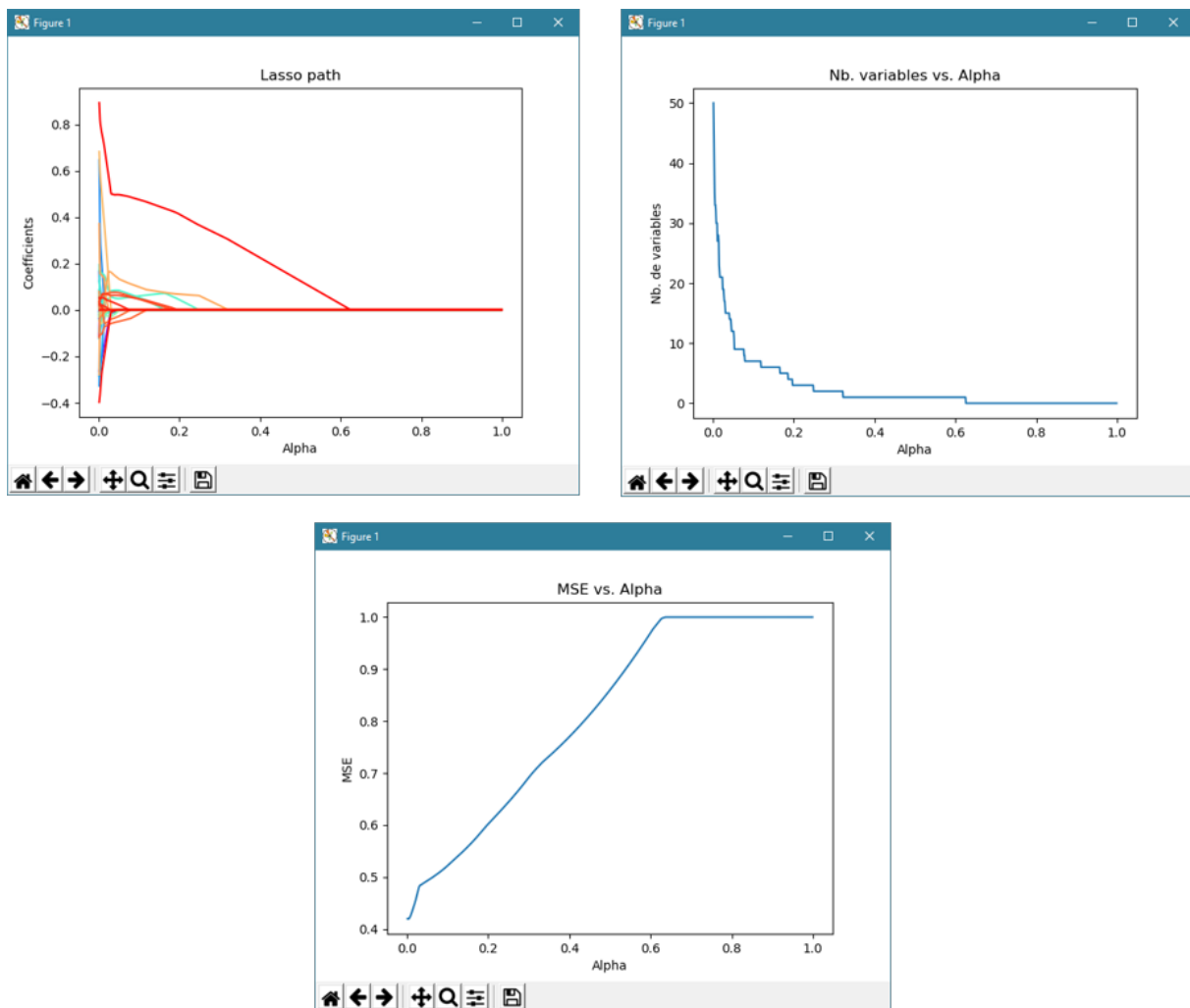


Figure 8 – Apprentissage N°3

L'erreur quadratique moyenne est en nette baisse sur ce modèle (ici la MSE vaut 41.9% pour  $\alpha=0.002$ ). La partie gauche du Lasso path est également plus étalée, ce qui indique que les caractéristiques contiennent plus d'informations : en effet 45 variables ont contribué à l'apprentissage du modèle. Les 5 premières caractéristiques sont : H, C51, C13, R, C4.

## 3.1.4 Apprentissage 4 : 20 et 50 jobs, 104440 individus, sans RBS

L'apprentissage précédent constituait un cas idéal d'apprentissage avec une taille d'instance fixe. Afin de permettre au modèle d'être plus polyvalent pour être utilisé sur des tailles d'instances différentes, une base d'apprentissage mixant des individus tirés d'instances de 50 jobs et d'instances de 20 jobs a été créé.

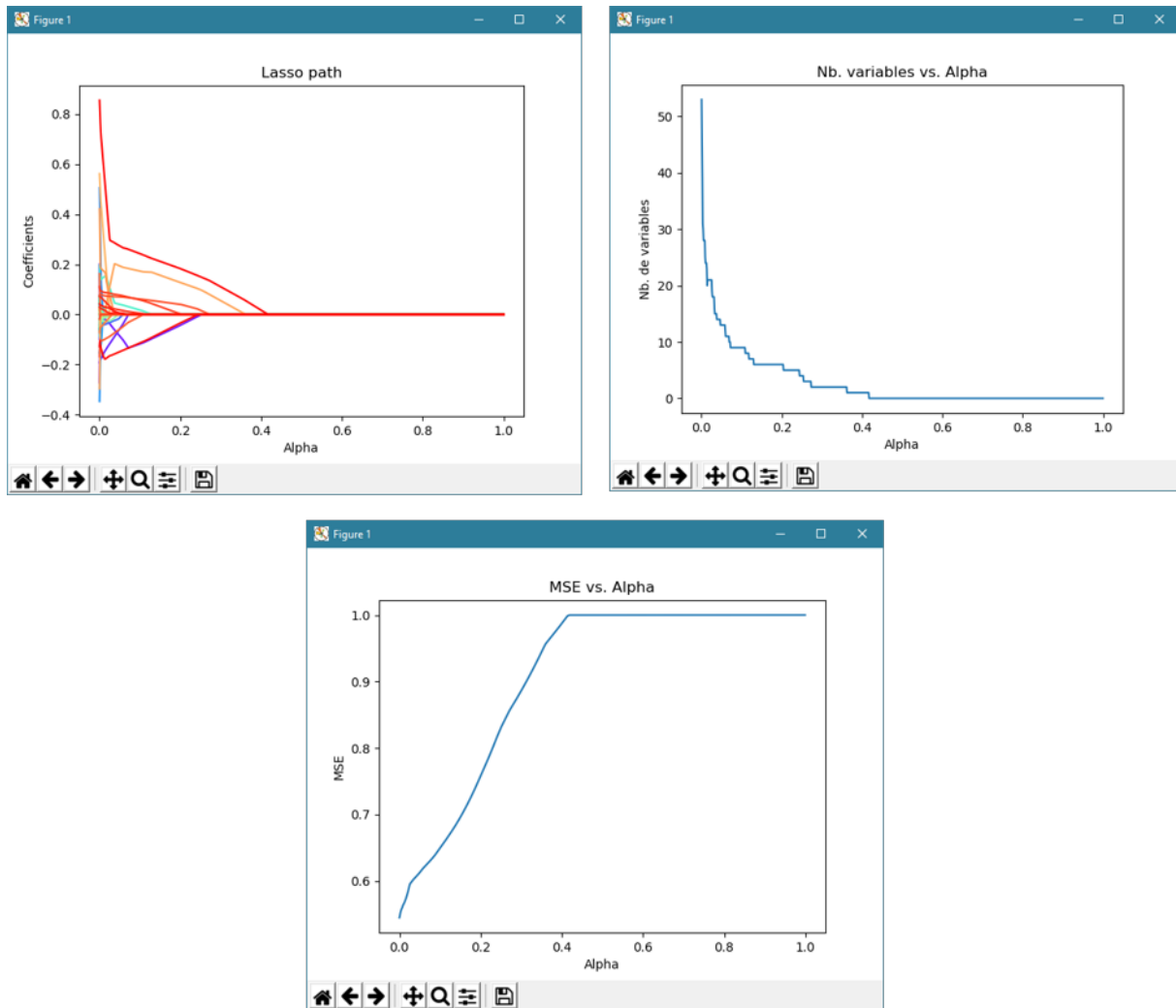


Figure 9 – Apprentissage N°4

L'erreur quadratique moyenne augmente fortement par rapport aux apprentissages précédent (ici 54.4% pour  $\alpha=0$ ). Le terme  $\alpha=0$  indique que chaque caractéristique a été utilisée dans l'apprentissage et qu'il n'y a pas eu de sélection. Cette grande augmentation du MSE vient du fait que la variable cible, le pourcentage d'amélioration (8), n'est pas adapté pour une base d'apprentissage qui contient des individus issus de tailles d'instances différentes.

Une autre variable cible a donc été choisie : la variable delta définie par :

$$\text{delta} = \frac{S - S'}{n - r + 1} \quad (9)$$

Cette variable prend en compte l'impact potentiel de l'amélioration dans la fenêtre sur les jobs situés après la fenêtre. Elle représente donc mieux de manière générale la qualité d'une solution.



## 3.1.5 Apprentissage 5 : 20 et 50 jobs, 104440 individus, sans RBS, variable cible delta

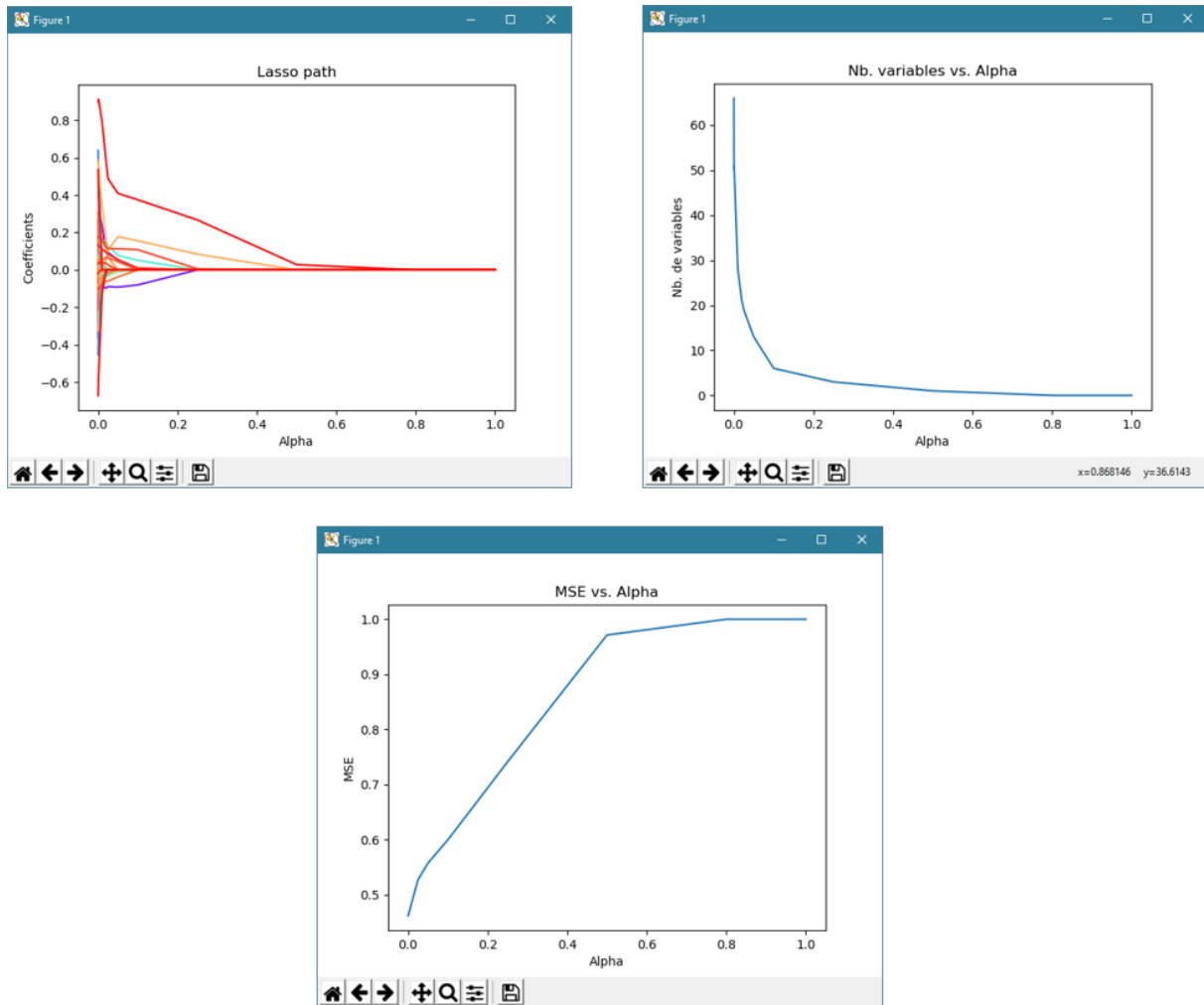


Figure 10 – Apprentissage N°5

L'erreur quadratique moyenne baisse en utilisant le critère « delta » (ici 46.2% pour  $\alpha=0.0001$ ). Les 5 caractéristiques les plus utilisées sont : H, R, C13, C51, C22. Ce critère permet donc de construire des modèles de prédiction plus polyvalents.

Face à la valeur de MSE qui restait élevée malgré les grandes bases d'apprentissage, il a été décidé d'ajouter une nouvelle famille de caractéristiques. Il s'agit de 13 caractéristiques liées à l'ordonnancement SRPT, présentées dans la partie 1.13. Les variables R, H et Size ont également été retirées de la base d'apprentissage afin de construire un modèle uniquement basé sur les familles de caractéristiques.

## 3.1.6 Apprentissage 6 : 20 et 50 jobs, 9680 individus, sans RBS, variable cible delta, caractéristiques famille 12

À l'issue de cet apprentissage, le Lasso Path est moins tassé sur la gauche par rapport aux apprentissages précédents. Il semble donc que ces nouvelles caractéristiques apportent de l'information au modèle de prédiction. La MSE est en baisse (41.9% pour  $\alpha=0.0001$ ), il s'agit de

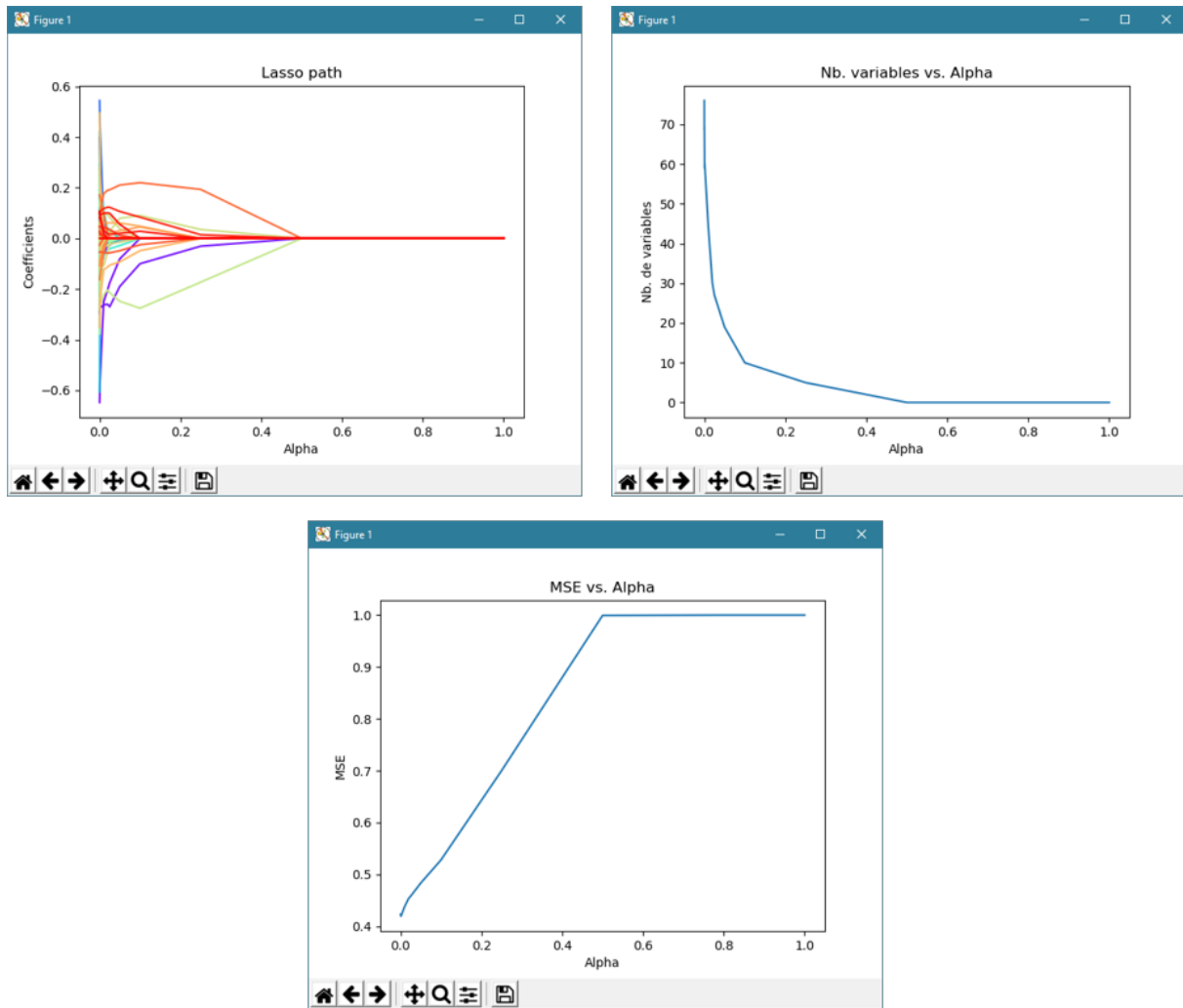


Figure 11 – Apprentissage N°6

la MSE la plus faible obtenue lors des tests d'apprentissage pour un modèle construit à partir d'une base d'apprentissage composée d'instances de tailles multiples.

L'ajout de ces caractéristiques a demandé la régénération entière de la base d'apprentissage. La génération est une opération très coûteuse en temps de calcul, et par manque de temps, il n'a pas été possible de construire une plus grande base d'apprentissage avec ces nouvelles caractéristiques.

Le modèle issu de cet apprentissage sera utilisé pour toutes les expériences suivantes.

### 3.2 Comparaison avec la matheuristique originale

Dans cette partie de l'étude, le but était de comparer les performances du modèle de prédiction sur différentes matheuristiques par rapport à la matheuristique initiale.

Le tableau des statistiques est disponible en annexe E.

On peut constater que le modèle de prédiction est plutôt efficace sur les instances de 50 jobs car on arrive aux mêmes solutions qu'avec la matheuristique d'origine en moins de temps. En revanche, sur les instances de 100 et 200 jobs, les matheuristiques qui utilisent le modèle de prédiction ne permettent pas d'avoir de meilleures solutions qu'avec la matheuristique originale.

Ces mauvaises performances s'expliquent par le fait que le modèle effectue des mauvaises prédictions : il prédit des améliorations lorsque ce n'est pas le cas en réalité, et inversement.

Pour mieux mesurer la qualité des prédictions du modèle, un test de spécificité et de sensibilité a été effectué : le modèle a été testé sur plusieurs instances à l'aide de la matheuristique bruteforce, et à chaque itération la prédiction du modèle ainsi que la vérité terrain ont été enregistrées.

La condition du test était : si le modèle prédit une valeur delta strictement supérieure à 0 à partir des caractéristiques de la séquence, alors il prédit une amélioration, sinon il ne prédit pas d'amélioration.

Les résultats ont été mis sous la forme d'un tableau :

|                     | Amélioration | Pas d'amélioration |      |
|---------------------|--------------|--------------------|------|
| Prédiction positive | 4            | 520                | 524  |
| Prédiction négative | 17           | 3619               | 3636 |
|                     | 21           | 4139               | 4160 |

Figure 12 – Test diagnostique du modèle (sur 3 instances de 50 jobs)

Plusieurs indicateurs statistiques peuvent être extraits de ce tableau :

- Précision = Probabilité qu'une prédiction soit correcte =  $\frac{4+3619}{4160} = 87.1\%$
- Sensibilité = Probabilité qu'une prédiction soit positive sachant qu'il y a une amélioration =  $\frac{4}{21} = 19\%$
- Spécificité = Probabilité qu'une prédiction soit négative sachant qu'il n'y a pas d'amélioration =  $\frac{3619}{4139} = 87.4\%$
- Valeur prédictive positive = Probabilité qu'il y ait une amélioration sachant que la prédiction est positive =  $\frac{4}{524} = 0.8\%$
- Valeur prédictive négative = Probabilité qu'il n'y ait pas d'amélioration sachant que la prédiction est négative =  $\frac{3619}{3636} = 99.5\%$
- F-mesure =  $2 * \frac{\text{sensibilité} * \text{valeur prédictive positive}}{\text{sensibilité} + \text{valeur prédictive positive}} = 2 * \frac{0.19 * 0.008}{0.19 + 0.008} = 0.015$

Ces indicateurs montrent que le modèle de prédiction, malgré sa bonne précision, a une sensibilité très faible et donc n'arrive pas à déterminer si une fenêtre peut mener à une amélioration. Concrètement, cela veut dire que le modèle de prédiction ignore 81% (1 - sensibilité) des possibilités d'améliorations.

Le modèle de prédiction est en revanche efficace pour déterminer si une fenêtre ne mène pas à une amélioration car la spécificité est plutôt élevée, mais ce n'est pas suffisant pour que le modèle de prédiction soit utilisable dans une matheuristique. La valeur de F-mesure proche de 0 indique d'ailleurs que les performances du modèle en classification sont mauvaises.

L'idéal serait d'avoir un modèle de prédiction avec à la fois une très forte sensibilité et une très forte spécificité. De cette façon, la classification serait très efficace et le modèle de prédiction pourrait faire gagner énormément de temps par rapport à la matheuristique originale en améliorant uniquement sur les bonnes fenêtres.

Pour arriver à cette qualité de prédiction, il faudrait construire un modèle de prédiction à partir d'une base d'apprentissage issue d'un très grand nombre d'instances et probablement utiliser des caractéristiques supplémentaires.

# 6

## Bilan et conclusion

### 1 Bilan du semestre 10

#### 1.1 Fait

À la date du rendu du rapport final de PRD, les tâches suivantes ont été effectuées :

- Compréhension du problème du flowshop.
- Compréhension de la construction de la base d'apprentissage.
- Compréhension de la matheuristique.
- Compréhension de la méthode Lasso.
- Planification du projet.
- Rédaction du rapport du semestre 9, incluant le cahier des spécifications et l'état de l'art.
- Formation sur Python et Scikit-Learn.
- Programmation du générateur d'instances.
- Programmation du constructeur de base d'apprentissage et test.
- Programmation du constructeur de modèle de prédiction.
- Programmation des différentes matheuristiques.
- Documentation du code.
- Guide d'installation et d'utilisation des différents outils.
- Cahier de recette contenant différents tests fonctionnels pour vérifier que les outils fonctionnent correctement.
- Tests d'apprentissage pour construire le modèle de prédiction avec un taux d'erreur le plus faible possible.
- Comparaison du modèle sur les différentes matheuristiques.
- Finalisation du rapport, incluant la partie mise en oeuvre.

#### 1.2 Reste à faire

- Génération d'une grande base d'apprentissage (supérieure à 100000 individus) en utilisant l'intégralité des caractéristiques pour vérifier si on peut améliorer les statistiques du modèle.

## 2 Bilan sur la qualité

La qualité de code est assurée pour la partie Python grâce à l'utilisation des outils Flake8 et Pylint qui vérifient la syntaxe et la sémantique du code. Ces outils ont été exécutés à chaque commit sur le Git pour vérifier que le code respectait les normes.

Pour la partie C++, les normes utilisées sont celles qui étaient déjà présentes dans le code original.

Un cahier de recette, disponible sur le git, a été créé et liste une suite de différents tests à effectuer pour s'assurer que les outils développés fonctionnent correctement.

Ce rapport sera évalué et contrôlé par les encadrants de ce PRD.

## 3 Bilan auto-critique sur la gestion du projet

Le développement a pris plus de temps que prévu, notamment la partie "génération du modèle de prédiction" à cause des nombreux essais qui ont été nécessaires pour obtenir un modèle de prédiction utilisable. Les événements qui sont survenus à la fin du semestre ne m'ont pas permis de faire des essais d'apprentissage sur des bases d'apprentissage plus volumineuses car je n'avais plus accès aux machines de Polytech pour générer les bases d'apprentissages.

Cependant, tous les outils demandés dans le cahier des charges ont été programmés et les expérimentations qui n'ont pas pu être faites durant le PRD pourront être faites plus tard grâce à ces outils.

## Annexes

# A

## Spécifications fonctionnelles

### 1 Programme d'apprentissage (langage Python)

#### 1.1 Fonction 1.1 : diviserBaseApprentissage

##### Description

Divise la base d'apprentissage en deux sous-ensembles (méthode de cross-validation) : un ensemble servira pour la construction du modèle et un ensemble servira pour la fonction de test du modèle. Les deux sous-ensembles seront construits de manière aléatoire, en respectant le pourcentage passé en paramètre.

##### Entrées

- Base d'apprentissage entière : liste de vecteurs caractéristiques avec leurs pourcentages d'amélioration.
- Pourcentage de la base d'apprentissage à utiliser pour l'apprentissage du modèle : entier  $x$ .

##### Sorties

- Base pour l'apprentissage du modèle :  $x\%$  de la base d'apprentissage initiale.
- Base pour l'évaluation du modèle :  $(1-x)\%$  de la base d'apprentissage initiale.

#### 1.2 Fonction 1.2 : générerModele

##### Description

Génère un modèle de prédiction à partir de la base d'apprentissage.

##### Entrées

- Base pour l'apprentissage du modèle : liste de vecteurs caractéristiques avec leurs pourcentages d'amélioration. Cette base est retournée par la fonction 1.1.

**Sorties**

- Modèle de prédiction au format Scikit-Learn.

**1.3 Fonction 1.3 : testerModele****Description**

Teste la qualité du modèle généré par la fonction 1.4.

**Entrées**

- Base pour l'évaluation du modèle : liste de vecteurs caractéristiques avec leurs pourcentages d'amélioration. Cette base est retournée par la fonction 1.1.

**Sorties**

- Qualité du modèle (MSE, % d'écart entre les solutions réelles et les solutions prédites).
- Variables sélectionnées pour l'apprentissage du modèle par LASSO.

**1.4 Fonction 1.4 : exporterModele****Description**

Exporte le modèle construit par Scikit-Learn pour le réutiliser dans la matheuristique. Le modèle de prédiction étant basé sur une régression Lasso, le format de sortie sera la liste des coefficients du modèle.

**Entrées**

- Modèle de prédiction au format Scikit-Learn.

**Sorties**

- Fichier CSV contenant les coefficients du modèle.

**2 Génération de la base d'apprentissage (langage C++)****2.1 Fonction 2.1 : obtenirVecteurCaracteristiques****Description**

Elle retourne le vecteur de caractéristiques correspondant à la solution passée en paramètre.

**Entrées**

- Solution : permutation de travaux.

**Sorties**

- Vecteur de caractéristiques de la solution passée en paramètre.



## 2.2 Fonction 2.2 : exporterBaseApprentissage

### Description

Cette fonction fait appel à la fonction 2.1 pour obtenir les vecteurs caractéristiques des solutions puis retourne une base d'apprentissage formatée en CSV.

### Entrées

- Séquence de travaux : séquence générée par l'algorithme RBS.

### Sorties

- Base d'apprentissage formatée en CSV.

## 3 Amélioration de la matheuristique (langage C++)

### 3.1 Fonction 3.1 : chargerModele

#### Description

Charge le modèle de prédiction depuis un chemin de fichier passé en paramètre et retourne un tableau contenant les coefficients du modèle de prédiction.

#### Entrées

- Chemin de fichier du modèle de prédiction : chaîne de caractères.

#### Sorties

- Modèle de prédiction : liste de nombres réels.

### 3.2 Fonction 3.2 : predireAmelioration

#### Description

Cette fonction prédit l'amélioration possible d'une solution à partir de son vecteur de caractéristiques.

#### Entrées

- Coefficients du modèle : liste de nombres réels.
- Vecteur de caractéristiques de la solution passée en paramètre.

#### Sorties

- Pourcentage d'amélioration : nombre réel.

# B

## Spécifications non fonctionnelles

### 1 Contraintes de développement et conception

#### 1.1 Matériel

Il n'y a pas de contraintes de développement liées au matériel. Une machine plus puissante qu'une autre effectuera simplement les calculs plus rapidement.

#### 1.2 Langages

Il y aura deux langages de programmation seront utilisés au cours du développement du projet :

- Python : Python sera utilisé pour développer le programme qui génère le modèle de prédiction. Ce langage est en effet très adapté pour ce cas car il dispose de nombreuses bibliothèques dont Scikit-Learn permettant de mettre en place du machine learning. On pourra également tracer différents graphiques facilement grâce à des bibliothèques comme Matplotlib pour analyser les données issues de l'apprentissage.
- C++ : Nous utiliserons le langage C++ pour intégrer la partie machine learning à la matheuristique originale ainsi que pour générer la base d'apprentissage. Ce langage est utilisé car la matheuristique est déjà codée en C++. On utilisera également ce langage pour effectuer l'extraction des caractéristiques et générer la base d'apprentissage car c'est une opération qui demande un grand temps de calcul, il est donc préférable d'utiliser un langage compilé.

#### 1.3 Bibliothèques

Nous utiliserons la bibliothèque Scikit-Learn qui permet de mettre en place des méthodes de machine learning en utilisant la régression Lasso. Cette bibliothèque est bien documentée, est stable et de nombreux tutoriels sont disponibles pour comprendre son fonctionnement.

L'API du logiciel CPLEX sera également utilisée afin de lancer une résolution exacte sur la fenêtre d'une séquence.

Concernant les IDE, Visual Studio Code sera utilisé pour la programmation Python et Visual Studio pour la programmation en C++.

## 2 Contraintes de fonctionnement et d'exploitation

### 2.1 Performances

Il n'y a pas de contraintes de performances pour ce projet. Il faudra cependant que la matheuristique modifiée avec la méthode de prédiction tende plus rapidement vers des bonnes solutions que la matheuristique initiale.

### 2.2 Capacité

Il n'y a pas de limite de capacité, que ce soit sur la génération de la base d'apprentissage ou sur l'apprentissage du modèle. Plus la base d'apprentissage sera complète, plus elle prendra du temps à être générée mais plus le modèle de prédiction sera précis.

### 2.3 Contrôlabilité

Pour suivre la construction du modèle de prédiction, des logs seront affichés dans la console.

### 2.4 Sécurité

Il n'y a pas eu de demandes spécifiques concernant la sécurité. Le programme ne stockera pas de données personnelles et les utilisateurs seront des chercheurs travaillant au LIFAT donc il n'y a pas besoin d'un accès sécurisé au logiciel.

### 2.5 Risques

Le risque principal du projet est de ne pas trouver de caractéristiques qui représentent correctement une solution améliorante. Il serait alors impossible de construire un modèle de prédiction cohérent.

### 2.6 Rendu des livrables

Ce rapport devra être rendu avant le 20 décembre 2019. Le rapport final de PRD sera à rendre pour le 1<sup>er</sup> avril 2020.

# C

## Plan de développement

### 1 Découpage du projet en tâches

#### 1.1 Tâche 1 : comprendre le problème du flowshop

##### Description

Cette tâche consiste à comprendre le coeur du problème à résoudre : le problème du flowshop.

##### Livrables

- Présentation PowerPoint expliquant le problème du flowshop.

##### Estimation de charge

- Cette tâche est estimée à 1 jour/homme.

#### 1.2 Tâche 2 : comprendre la matheuristique

##### Description

Cette tâche consiste à comprendre le fonctionnement de la matheuristique existante pour pouvoir ensuite l'améliorer.

##### Livrables

- Présentation PowerPoint expliquant la matheuristique.

##### Estimation de charge

- Cette tâche est estimée à 7 jours/homme.

### 1.3 Tâche 3 : comprendre la construction de la base d'apprentissage

#### Description

Cette tâche consiste à comprendre la construction de la base d'apprentissage construite par l'étudiant qui a travaillé sur le sujet pendant l'année 2018-2019.

#### Livrables

- Présentation PowerPoint sur la construction de la base d'apprentissage.

#### Estimation de charge

- Cette tâche est estimée à 5 jours/homme.

### 1.4 Tâche 4 : comprendre les familles de vecteurs

#### Description

Cette tâche consiste à comprendre les features décrites dans le document NotesF2et ML du 4.4.19 v2.pdf.

#### Estimation de charge

- Cette tâche est estimée à 7 jours/homme.

### 1.5 Tâche 5 : planification du projet

#### Description

Cette tâche consiste à diviser le projet en tâches et à planifier les différentes tâches du projet.

#### Livrables

- Diagramme de Gantt du projet.

#### Estimation de charge

- Cette tâche est estimée à 2 jours/homme.

### 1.6 Tâche 6 : rédaction des spécifications fonctionnelles

#### Description

Cette tâche consiste à définir les spécifications des logiciels qui seront à développer ainsi qu'à lister leurs fonctionnalités.

#### Livrables

- Cahier des spécifications.

#### Estimation de charge

- Cette tâche est estimée à 5 jours/homme.

## 1.7 Tâche 7 : rédaction de l'état de l'art

### Description

Cette tâche consiste à rechercher des méthodes existantes pour décrire les solutions d'un problème de flowshop. Ces méthodes trouvées pourront être utilisées pour construire les vecteurs de caractéristiques des solutions.

### Livrables

- Document état de l'art.

### Estimation de charge

- Cette tâche est estimée à 7 jours/homme.

## 1.8 Tâche 8 : formation Python

### Description

Cette tâche consiste à se former sur le développement Python, et plus précisément sur la mise en place d'un environnement virtuel.

### Estimation de charge

- Cette tâche est estimée à 2 jours/homme.

## 1.9 Tâche 9 : formation méthode de régression LASSO

### Description

Cette tâche consiste à se former sur la régression LASSO pour comprendre comment un modèle de prédiction est généré.

### Livrables

- Document expliquant la méthode LASSO.

### Estimation de charge

- Cette tâche est estimée à 7 jours/homme.

## 1.10 Tâche 10 : formation Scikit-Learn

### Description

Cette tâche consiste à se former sur le fonctionnement de la bibliothèque Scikit-Learn. Le but sera de faire un code de test sur des données de test qui génère puis teste un modèle de prédiction.

### Estimation de charge

- Cette tâche est estimée à 10 jours/homme.

**1.11 Tâche 11 : préparation de la soutenance S9****Description**

Cette tâche consiste à préparer la soutenance de mi-parcours.

**Livrables**

- Soutenance avec présentation PowerPoint.

**Estimation de charge**

- Cette tâche est estimée à 4 jours/homme.

**1.12 Tâche 12 : définition des vecteurs de caractéristiques****Description**

Cette tâche consiste à définir quelles sont les features qui seront utilisées dans le code pour construire les vecteurs de caractéristiques.

**Estimation de charge**

- Cette tâche est estimée à 10 jours/homme.

**1.13 Tâche 13 : génération d'une base de vecteurs de caractéristiques****Description**

Cette tâche consiste à créer le programme de génération de la base d'apprentissage qui servira à créer le modèle de prédiction.

**Livrables**

- Programme de génération de base d'apprentissage.
- Documentation

**Estimation de charge**

- Cette tâche est estimée à 20 jours/homme.

**1.14 Tâche 14 : apprentissage du modèle de prédiction****Description**

Cette tâche consiste à créer le programme qui générera le modèle de prédiction.

**Livrables**

- Programme d'apprentissage du modèle de prédiction.
- Documentation

**Estimation de charge**

- Cette tâche est estimée à 10 jours/homme.

**1.15 Tâche 15 : intégration du programme dans la matheuristique****Description**

Cette tâche consiste à modifier la matheuristique pour y intégrer l'utilisation du modèle de prédiction.

**Livrables**

- Matheuristique améliorée.

**Estimation de charge**

- Cette tâche est estimée à 10 jours/homme.

**1.16 Tâche 16 : tests****Description**

Cette tâche consiste à tester les différentes parties de codes et à détecter et corriger d'éventuels défauts.

**Estimation de charge**

- Cette tâche est estimée à 10 jours/homme.

**1.17 Tâche 17 : rédaction du rapport final****Description**

Cette tâche consiste à rédiger le rapport de PRD final en y regroupant chaque partie rédigée (cahier des spécifications, état de l'art, ...).

**Livrables**

- Rapport de PRD.

**Estimation de charge**

- Cette tâche est estimée à 3 jours/homme.

**1.18 Tâche 18 : préparation de la soutenance S10****Description**

Cette tâche consiste à préparer la soutenance de fin de PRD.

**Livrables**

- Soutenance avec présentation PowerPoint.



Estimation de charge

— Cette tâche est estimée à 15 jours/homme.

2 Planification à la fin du S9

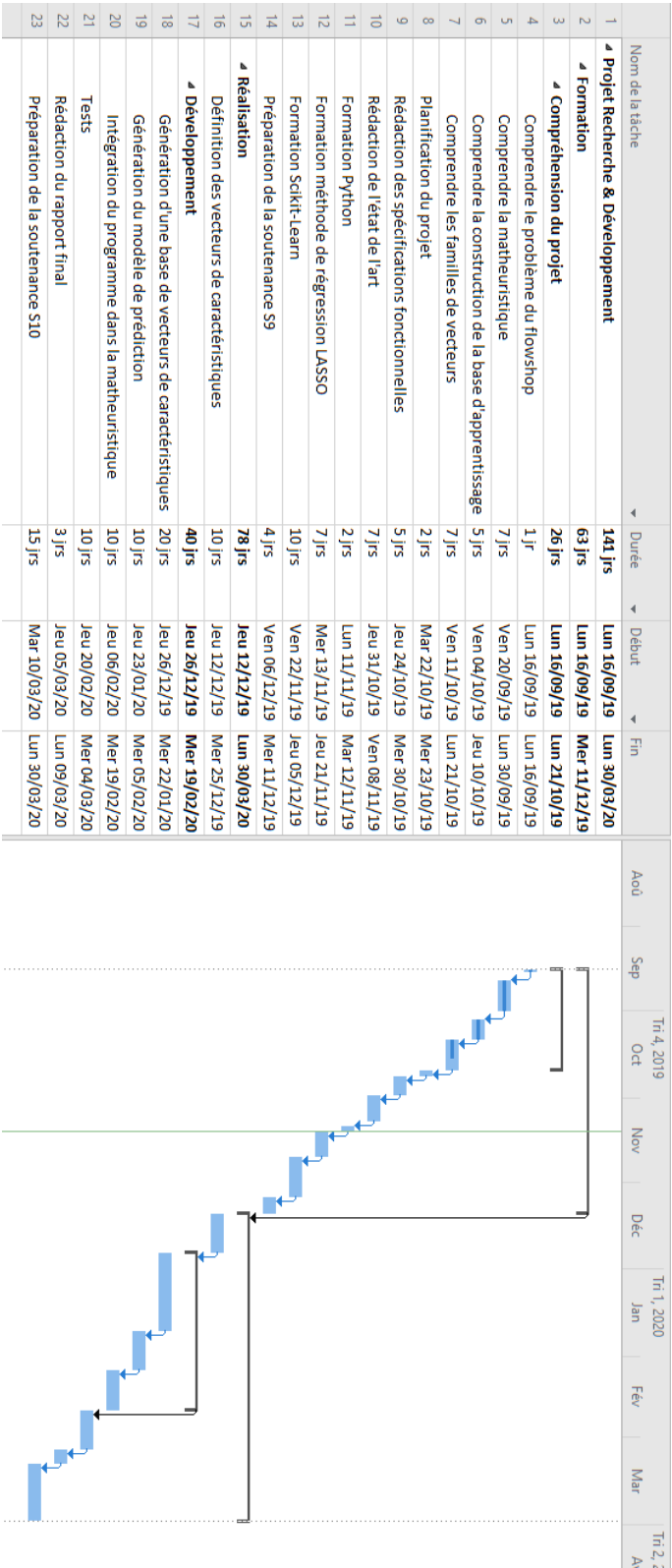


Figure 1 – Diagramme de Gantt à la fin du S9

3 Planification à la fin du S10



Figure 2 – Diagramme de Gantt à la fin du S10

# D

## Description des interfaces externes du logiciel

### 1 Interfaces matériel/logiciel

Ce projet ne dépend pas d'un matériel spécifique, il n'y aura donc pas d'interfaces entre le matériel et le logiciel.

### 2 Interfaces homme/machine

Pour l'apprentissage du modèle de prédiction, nous utiliserons une interface en ligne de commandes (CLI) sous forme de menus permettant d'ajuster les paramètres d'apprentissage et de choisir le fichier de base d'apprentissage à utiliser.

Pour le chargement du modèle de prédiction dans la matheuristique, il y aura également une interface CLI permettant d'entrer le nom du fichier du modèle de prédiction à charger.

### 3 Interfaces logiciel/logiciel

La matheuristique existante est codée en C++ et le programme d'apprentissage sera codé en Python, mais il n'y a pas d'interfaces à construire entre les logiciels : les modèles de prédictions seront exportés depuis Python sous une forme CSV puis seront parsés dans le code C++.

La matheuristique utilise cependant l'API du logiciel CPLEX qui devra donc être installé sur la machine du client.

# Statistiques de comparaison entre les matheuristiques

| Fichier          | Nb jobs | Tlim (secondes) | RBS    | r aléatoire, h=12 |        |            |        | tous les r, Acch=16      |       |                                       |        | tous les r, Acch=12, tri des deltas décroissants |        |  |  |
|------------------|---------|-----------------|--------|-------------------|--------|------------|--------|--------------------------|-------|---------------------------------------|--------|--|--------|--|--|
|                  |         |                 |        | Origine           |        | Prédicteur |        | Prédicteur (brute force) |       | Prédicteur (70% des meilleurs deltas) |        | Prédicteur (jusqu'à 10% du pire delta)           |        |  |  |
| instance01.txt   | 50      | 30              | 46317  | 10,83             | 46183  | 0,11       | 46288  | 16,56                    | 46184 | 7,14                                  | 46184  | 1,02   | 46251  |  |  |
| instance02.txt   | 50      | 30              | 55777  | 5,71              | 55790  | 2,32       | 55729  | 31,95                    | 55730 | 2,41                                  | 55729  | 2,45   | 55729  |  |  |
| instance03.txt   | 50      | 30              | 47656  | 9,797             | 47618  | 2,553      | 47618  | 14,02                    | 47618 | 6,034                                 | 47617  | 7,25   | 47617  |  |  |
| instance1001.txt | 100     | 500             | 184749 | 32,873            | 184438 | 66,254     | 184434 | trop long -              |       | 268,973                               | 184434 | 290,79   | 184434 |  |  |
| instance1002.txt | 100     | 500             | 209553 | 41,756            | 208254 | 18,217     | 208491 | trop long -              |       | 578,44                                | 208276 | 583,89   | 208276 |  |  |
| instance2001.txt | 200     | 1000            | 857016 | 551,51            | 855408 | 575,835    | 855428 | trop long -              |       | trop long -                           |        | trop long -                                      |        |  |  |

Figure 1 – Comparaison entre les différentes matheuristiques

# F

# Documentation d'installation

## 1 Générateur d'instances, constructeur de base d'apprentissage, matheuristiques

Ces outils se nomment respectivement "GenInstanceFlowshop", "GenBD Avec Descripteurs" et "Matheuristic" sur le Git. Ils ont été compilés avec Visual Studio 2017 et les exécutables sont disponibles dans les sous-dossiers "x64/Release" de chaque projet.

Le projet C++ Visual Studio de chaque outil est également fourni. Pour compiler ces différents outils avec les fichiers Visual Studio fournis, il est nécessaire d'installer :

- Microsoft Visual Studio 2017
- IBM CPLEX 12.9

Puis de compiler en 64 bits et en mode Release.

## 2 Constructeur de modèle de prédiction

Le constructeur de modèle de prédiction s'exécute dans un environnement Python. Pour l'utiliser, il est nécessaire d'installer :

- Python 3.7.6 (64 bits de préférence, surtout pour apprendre un modèle à partir de grandes bases d'apprentissage)
- Les dépendances Python du projet : elles sont listées dans le fichier requirements.txt à la racine du Git. Pour installer les dépendances, il suffit d'exécuter la commande "pip install -r requirements.txt"

## 3 Documentation développeurs

Les documentations des fonctions et classes de chaque outil sont disponibles dans le dossier html de chaque projet.



# Documentation d'utilisation

## 1 Génération d'instances de flowshop

Pour générer x instances de n jobs et m machines :

*GenInstanceFlowshop.exe n m x*

Les instances sont générées dans le sous-dossier /instances.

## 2 Génération d'une base d'apprentissage avec descripteurs

Pour générer une base d'apprentissage avec descripteurs à partir d'un dossier d'instances :

*GenBDavecDescripteurs.exe nom\_dossier\_instances numero\_instance\_min numero\_instance\_max  
proba\_cross\_validation\_base\_apprentissage RBS random\_seed(optionnel)*

ou en version multithread :

*GenBDavecDescripteursMultithread.bat nom\_dossier\_instances nb\_instances nb\_threads  
proba\_cross\_validation\_base\_appr RBS*

La sortie donne un fichier DatabaseTrain.txt et DatabaseTest.txt qui contiennent respectivement la base d'apprentissage et la base de test. Les deux bases ne contiennent pas d'instances communes.

Il est donc possible de lancer plusieurs fois le programme GenBDavecDescripteurs.exe afin de paralléliser la génération de la base d'apprentissage.

Exemple pour paralléliser le calcul de 20 instances sur 4 coeurs sans RBS :

*GenBDavecDescripteurs.exe instances 1 5 0.5 0*  
*GenBDavecDescripteurs.exe instances 6 10 0.5 0*  
*GenBDavecDescripteurs.exe instances 11 15 0.5 0*  
*GenBDavecDescripteurs.exe instances 16 20 0.5 0*

Ou directement avec le script multithread :

*GenBDavecDescripteursMultithread.bat instances 20 4 0.5 0*

### 3 Construction d'un modèle de prédiction

Exécuter :

```
python GenModPredictionPourcentImp.py chemin_base_apprentissage chemin_base_test silent_mode
```

Ce script construit le modèle de prédiction avec la méthode LASSO et affiche différentes statistiques sur la construction de celui-ci (LASSO Path, Nombre de variables en fonction de alpha, MSE en fonction de alpha).

Le paramètre `silent_mode` peut prendre plusieurs valeurs :

- "graphs", affiche les différents graphiques au cours de la construction du modèle.
- "silent", permet de lancer la construction du modèle en silence (sans les graphiques).

Le modèle est ensuite exporté dans un fichier `model.dat` .

### 4 Utilisation du modèle de prédiction dans la matheuristique

Pour lancer la résolution d'un problème de flowshop à l'aide du prédicteur :

```
Matheuristic.exe chemin_fichier_instance chemin_fichier_modele numero_matheuristique  
limite_temps(pour matheuristique 1 et 2) random_seed(optionnel)
```

`numero_matheuristique` peut prendre différentes valeurs :

- 1 : bruteforce avec prédicteur (tous les  $r$ ,  $4 \leq h \leq 16$ )
- 2 : matheuristique originale avec prédicteur (tous les  $r$ ,  $h=12$ )
- 3 : nouvelle matheuristique (deltas décroissant, stoppe à 70% à la liste des deltas) avec prédicteur (tous les  $r$ ,  $4 \leq h \leq 12$ )
- 4 : nouvelle matheuristique (deltas décroissant, stoppe à 10% du pire delta) avec prédicteur (tous les  $r$ ,  $4 \leq h \leq 12$ )

La meilleure séquence trouvée ainsi que sa fonction objectif est sauvegardée dans un fichier nommé `chemin_fichier_instance.seq` .



# Webographie

- [WWW1] Jain AARSHAY. *A Complete Tutorial on Ridge and Lasso Regression in Python*. URL : <https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/>.
- [WWW2] Université de LYON 2. *Régression Lasso sous Python*. URL : [http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/fr\\_Tanagra\\_Regression\\_Lasso\\_Python.pdf](http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/fr_Tanagra_Regression_Lasso_Python.pdf).
- [WWW3] Ricco RAKOTOMALALA. *Régression régularisée - Ridge, Lasso, Elasticnet*. URL : [http://eric.univ-lyon2.fr/~ricco/cours/slides/regularized\\_regression.pdf](http://eric.univ-lyon2.fr/~ricco/cours/slides/regularized_regression.pdf).
- [WWW4] Josh STARMER. *Regularization Part 2 : Lasso Regression*. URL : <https://www.youtube.com/watch?v=NGf0voTMlcs>.
- [WWW5] Réponse de XAVIER BOURRET SICOTTE. *Coordinate descent soft-thresholding update operator for LASSO*. URL : <https://stats.stackexchange.com/questions/123672/coordinate-descent-soft-thresholding-update-operator-for-lasso>.



# Bibliographie

- [1] Alafate ABULIMITI. « Recherche Opérationnelle et Apprentissage Automatique ». Projet Recherche & Développement. Tours, France : Ecole Polytechnique de l'Université François Rabelais de Tours, 2018-2019.
- [2] Sana El BAHOU. « Flow-shop à deux machines avec des temps de latence : approche exacte et heuristique ». Mém. de mast. Université du Québec à Chicoutimi, jan. 2008. URL : <https://archipel.uqam.ca/1410/1/M10431.pdf>.
- [3] Frederico Della CROCE, Andrea GROSSO et Fabio SALASSA. « A matheuristic approach for the two-machine total completion time flow shop problem ». In : *Annals of Operations Research* 213 (juil. 2014), p. 67-78.
- [4] Yongda KIM. « Gradient LASSO algorithm ». Juin 2006. URL : <http://www.cs.cmu.edu/afs/cs/project/link-3/lafferty/www/ml-stat2/talks/YondaiKimGLasso-SLIDE-YD.pdf>.
- [5] Romain RAVEAUX et Vincent T'KINDT. « Flowshop 2 machines et apprentissage (Notes suite à la réunion du 4 avril 2015) ». Avr. 2015.

# Recherche Opérationnelle et Machine Learning :

## Le cas d'une matheuristique pour un problème de flowshop

Guillaume Chevallier

Encadrement : Romain Raveaux, Vincent T'kindt et Nicolas Ragot

## Objectifs

Etudier l'apport du machine learning pour un problème de flowshop à deux machines.

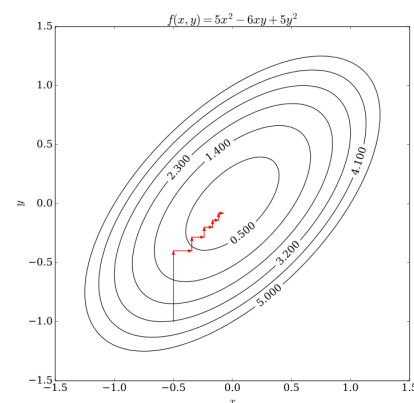
| TEMPS     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|-----------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| MACHINE 1 | 1 | 2 | 3 | 4 | 5 |   |   |   |   |    |    |    |    |    |    |    |

| MACHINE 2 |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |
|-----------|--|--|--|---|---|---|---|---|--|--|--|--|--|--|--|--|
|-----------|--|--|--|---|---|---|---|---|--|--|--|--|--|--|--|--|

Exemple d'ordonnancement d'un flowshop à 2 machines sur 5 jobs

## Mise en œuvre

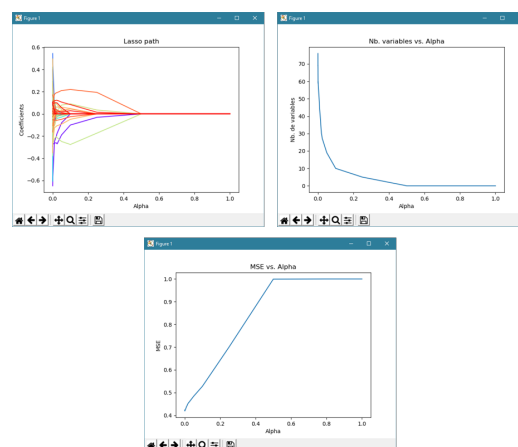
Génération d'une base d'apprentissage puis d'un modèle de prédiction pour accélérer et améliorer une matheuristique déjà existante.



Algorithme de descente de gradients utilisé pour l'apprentissage

## Résultats attendus

- Un outil permettant de générer des bases d'apprentissage.
- Un outil permettant de générer un modèle de prédiction.
- Une matheuristique hybridée plus performante.



Résultats d'un essai d'apprentissage

# Recherche Opérationnelle et Machine Learning : Le cas d'une matheuristique pour un problème de flowshop

## Guillaume Chevallier

Encadrement : Romain Raveaux, Vincent T'kindt et Nicolas Ragot

### Objectifs

Etudier l'apport du machine learning pour un problème de flowshop à deux machines.

### Mise en œuvre

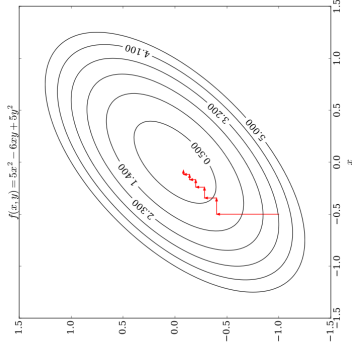
Génération d'une base d'apprentissage puis d'un modèle de prédiction pour accélérer et améliorer une matheuristique déjà existante.

### Résultats attendus

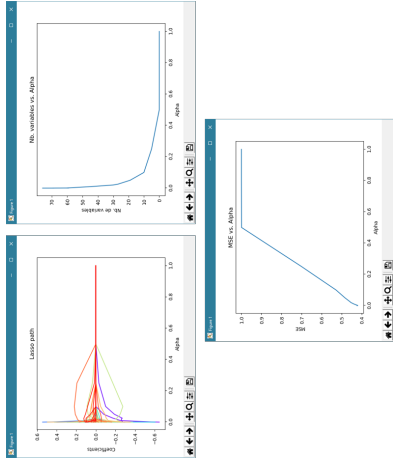
- Un outil permettant de générer des bases d'apprentissage.
- Un outil permettant de générer un modèle de prédiction.
- Une matheuristique hybridée plus performante.

| TEMPS     | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|-----------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| MACHINE 1 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5  |    |    |    |    |    |    |
| MACHINE 2 |   |   | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4  | 5  | 5  |    |    |    |    |

Exemple d'ordonnancement d'un flowshop à 2 machines sur 5 jobs



Algorithme de descente de gradients utilisé pour l'apprentissage



Résultats d'un essai d'apprentissage

# Recherche Opérationnelle et Machine Learning

## Le cas d'une matheuristique pour un problème de flowshop

### Résumé

L'objectif de ce projet est d'étudier l'apport du machine learning pour la résolution du problème du flowshop. Pour cela, nous partirons d'une matheuristique classique et déjà existante, puis nous mettrons au point un algorithme d'apprentissage à base de descripteurs qui devra permettre de guider plus rapidement cette matheuristique vers une bonne solution. Nous intégrerons ensuite cet algorithme dans la matheuristique et nous étudierons l'hybridation obtenue pour voir si elle est plus performante que la matheuristique initiale. Ce document définit le contexte et les objectifs du projet, les fonctionnalités du système à développer, l'état de l'art sur le problème du flowshop, la conception du système et enfin le bilan sur la première partie du développement du projet.

### Mots-clés

Lasso, régression linéaire, Python, machine learning, flowshop, matheuristique, recherche opérationnelle

### Abstract

The goal of this project is to study the contribution of machine learning for solving the flowshop problem. We will start from a classical and already existing matheuristic, then we will develop a features-based learning algorithm which will help to guide this matheuristic more quickly towards a good solution. We will then integrate this algorithm in the matheuristic, and we will study the hybridization to see if it is more powerful than the initial matheuristic. This document defines the context and objectives of the project, the functionalities of the system to be developed, the state of the art on the flowshop problem, the system design and finally the progress of the first part of the development of the project.

### Keywords

Lasso, linear regression, Python, machine learning, flowshop, matheuristic, operational research

### Tuteurs académiques

Romain RAVEAUX  
Vincent T'KINDT  
Nicolas RAGOT

### Étudiant

Guillaume CHEVALLIER (DI5)