Titre: Title:	Aide au choix d'une solution optimale sur un front Pareto à l'aide de méthodes de groupement
Auteur: Author:	Ilyas Rahhali
Date:	2020
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Rahhali, I. (2020). Aide au choix d'une solution optimale sur un front Pareto à l'aide de méthodes de groupement [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/5407/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/5407/
Directeurs de recherche: Advisors:	Charles Audet, & Jonathan Jalbert
Programme: Program:	Maîtrise recherche en mathématiques appliquées

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Aide au choix d'une solution optimale sur un front Pareto à l'aide de méthodes de groupement

ILYAS RAHHALI
Département de mathématiques et de génie industriel

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

Août 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Aide au choix d'une solution optimale sur un front Pareto à l'aide de méthodes de groupement

présenté par **Ilyas RAHHALI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Nadia LAHRICHI, présidente
Charles AUDET, membre et directeur de recherche
Jonathan JALBERT, membre et codirecteur de recherche
Daniel ALOISE, membre

REMERCIEMENTS

Je tiens à remercier chaleureusement mes directeurs de recherche Charles Audet et Jonathan Jalbert pour leur précieux support, présence et encadrement au cours de la réalisation de ce travail. Je tiens à remercier également Aimen Gheribi pour sa collaboration pour trouver un problème industriel réel permettant de valider et démontrer l'intérêt pratique de ce travail. Je remercie enfin Nadia Lahrichi et Daniel Aloise membres de jury pour leurs commentaires pertinants améliorant la qualité du présent mémoire.

RÉSUMÉ

Ce travail propose une méthodologie qui permet à des experts résolvant des problèmes biobjectifs et qui obtiennent un ensemble de solutions optimales sur un front Pareto de faire un choix informé de l'une ou plusieurs de ces solutions suivant des critères ne figurant pas forcément parmi les variables du problème. La méthodologie décrite dans ce travail se base sur des méthodes de segmentation permettant de regrouper les solutions partageant des caractéristiques communes. Ceci permet à l'expert de découvrir comment sont distribuées les solutions dans l'espace des paramètres. Cette approche est différente des approches de segmentation regroupant les points selon selon les valeurs des fonctions objectif, car cela nous permet d'obtenir des groupements distincts de points partageant des caractéristiques de variables similaires entre eux, qui constitue une information très intéressante pour l'expert. Cette approche répond également au besoin des experts de potentiellement faire des compromis d'optimalité et choisir des points répondant mieux à leurs critères, quitte à s'éloigner légérement du front Pareto. Différents types d'algorithmes de segmentation sont présentés et comparés. On propose ensuite deux heuristiques pour choisir des solutions parmi chaque groupement de solutions. La première étant une heuristique rapide ne requérant pas de critères spéciaux et proposant une solution automatiquement. La deuxième retourne les meilleures solution suivant une fonction score que l'expert définira suivant ses critères de choix de solutions. Une illustration de l'utilisation pratique de la méthodologie sera présentée enfin sur un projet d'industrie réel de synthétisation d'un fluide à partir de sels fondus pour une centrale solaire ainsi qu'un exemple de définition de fonction score et d'implémentation de la deuxième heuristique.

ABSTRACT

The purpose of this work is to define a novel methodology for indentifying interesting biobjective's problem solutions on or close to the Pareto front that responds to the expert's criteria. The methodology defined in this work is based on clustering algorithms to identify eventual clusters of solutions sharing similar variable characteristics and then identifying solutions of interests within each cluster. The experts often want to explore the near optimality region for solutions that can be more practical, cheaper or easier to implement than optimal ones. Different types of clustering algorithms will be discussed and compared, and two heuristics for choosing best points are defined. The first is an easy to use, automatic method that selects a solution within the densest region of each cluster and doesn't require the input of any criteria. The second heuristic implements a scoring system to rank solutions based on the experts requirements. An industry problem will be used to illustrate the use of the methodology and the heuristics. The novelty of this work is illustrated in clustering over the variable space rather than objective space, providing clusters with solutions with similar variable properties. It is also illustrated in providing a method for exploring promising solutions near the Pareto set.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1 INTRODUCTION

L'opimisation multi-objectif, et particulièrement bi-objectif, est au coeur des problèmes d'optimisation en milieu industriel. En effet, en industrie on a souvent besoin d'optimiser non une, mais deux ou plusieurs fonctions objectif en même temps [Kipouros et al.2008]. Par exemple pour des problèmes de conception d'une aile d'avion, il faut trouver un matériau qui soit à la fois léger et résistant aux contraintes mécaniques, ou encore pour concevoir un fluide caloporteur en énergie solaire, il faut que ce dernier aie une température de fusion minimale et une conductivité thermique maximale [Kenisarin2014]. Contrairement aux problèmes n'ayant qu'un objectif à optimiser, les solutions des problèmes multi-objectifs ne sont pas nécessairement uniques, mais sont constituées d'un ensemble de solutions optimales et non comparables appelées ensemble Pareto [Miettinen1999]. En pratique, il y a différentes philosophies de choix d'une solution, allant de celles où on choisit aléatoirement, jusqu'à appeler un expert pour en suggérer une souvent de manière intuitive [Masud1979].

Souvent, l'expert résolvant un problème biobjectif souhaite valoriser son savoir-faire et expérience pour guider son choix d'une solution optimale parmi les nombreuses que le solveur a fourni. À cette fin, et d'après des communications avec un tel expert, il lui est avantageux d'avoir accès à certaines informations sur les solutions trouvées tel que savoir si certaines sont agglomérées dans certaines régions de l'espace des variables (ce qu'on appellera au cours du document des groupements) et la distrubution et caractéristiques de tels groupements s'ils existent, et enfin pouvoir éventuellement pouvoir identifier des solutions répondant à certains critères qui ne font pas partie des fonctions objectif du problème d'optimisation. À moins de vérifier les solutions une par une, il lui est difficile d'accéder à ces informations. Les critères cités ci-dessus ne figurent pas parmi les fonctions objectif car lors de la définition d'un pro-

blème d'optimisation, on ne peut pas se permettre de traduire tous les critères en fonctions objectifs. En pratique, on en choisit un ou deux les plus importants et contraignants en tant que fonctions objectifs et idéalement utilise les autres critères pour choisir une solution intéressante. De plus, l'expert souhaite également très souvent explorer des solutions dans des régions proches des solutions optimales. Il n'est en effet pas rare de trouver des solutions plus intéressantes (moins coûteuses, plus faciles à fabriquer, etc..) dans ces régions bien qu'on doive faire un compromis d'optimalité. C'est pour répondre à ces deux problématiques que ce travail a été développé.

Au cours de ce travail, une approche basée sur l'apprentissage automatique est adoptée pour concevoir une méthodologie générale permettant à l'utilisateur d'identifier d'éventuels groupements de solutions selon les variables du problème et faire un choix mieux informé et justifié selon ses critères d'une ou plusieurs des solutions de l'ensemble Pareto ou proches de ce dernier qui lui conviennent le mieux et qui répondent à d'éventuelles caractéristiques souhaitées. Le décideur aura ainsi une meilleure base, et de l'information a priori pour faire un tel choix.

L'idée générale est de segmenter les points non dominés d'un problème bi-objectif dans l'espace des variables. Ceci permet de créer des groupements qui partagent des caractéristiques similaires. Par la suite, on visualise ces groupements dans l'espace objectif. L'expert pourra identifier les caractéristiques partagées par les groupements à l'aide d'un tableau récapitulatif de ces derniers et pourra sélectionner une solution dans la zone appartenant à un groupement d'intérêt en utilisant les heuristiques développées au cours de ce travail.

Ce travail se concentre sur l'application de cette méthodologie dans un contexte d'optimisation en boîte noires pour deux raisons. La première étant la nature du problème pratique sur lequel sera testée la méthode proposée. La deuxième est que l'une des limitations de cette méthode basée sur la segmentation est qu'elle requiert de traiter un nombre de petit ordre de grandeur de points.

Certaines méthodes de résolution de problèmes d'optimisation peuvent générer un très grand nombre de solutions telles que les méthodes génétiques ou des cas d'optimisation combinatoire, on se situe alors dans un contexte de boîtes noires et utilisant l'algorithme BIMADS comme solveur garantissant un nombre adéquat de solutions visitées.

Ce mémoire est présenté de la manière suivante. Le chapitre 2 commence par une brève explication du contexte d'optimisation en boîtes noires ainsi que des algorithmes MADS et BIMADS. Ensuite au même chapitre, on donne une revue de la littérature d'algorithmes de classification pertinents à ce travail en présentant leurs caractéristiques ainsi que leur fonctionnement.

Le chapitre 3 est le cœur du document. Il présente en détail la démarche suivie et propose une structure pour la segmentation des solutions non dominées de tout type de problème bi-objectif en boîtes noires, en expliquant le choix des algorithmes ainsi que les différentes manières dont ils peuvent être appliqués pour des problèmes réels. Ensuite, un algorithme de sélection de points proches du front Pareto est introduit sur lesquels sera appliquée la segmentation. Des figures comparant les quatres différentes méthodes de segmentation des points Pareto sont également fournies.

Le chapitre 4 présente une application de la démarche du chapitre précédent pour un problème d'industrie réel, en fournissant une manière d'interpréter les résultats obtenus et prendre de meilleurs choix ainsi que des figures illustrant chaque étape de la démarche. Ce problème a été fourni par Aimen Gheribi, un expert qui a permis de valider la pertinence et utilité de la méthodologie.

où $x = (x_1, x_2, .., x_n)$ représente les variables du problème, $F(x) = (f_1(x), f_2(x))$ représente les deux fonctions du problème bi-objectif à optimiser et $\Omega \subseteq \mathcal{R}^n$ représente l'ensemble réalisable. On note aussi $\mathcal{F}$ l'espace objectif des solutions dans le plan (f_1, f_2) inclu dans $\mathcal{R}^2$.

CHAPITRE 2 DÉFINITIONS ET REVUE DE LITTÉRATURE

La première partie de ce chapitre présente le cadre d'optimisation bi-objectif et le contexte d'optimisation en boîtes noires [Audet and Hare2017] pour préparer une présentation de l'algorithme BIMADS qui sera utilisé dans la partie pratique pour résoudre les problèmes bi-objectifs. Ensuite, le concept de segmentation en apprentissage automatique et certains des algorithmes les plus pertinents pour ce travail seront introduits. Et finalement, une revue de littérature de l'optimisation multi-critère, et de méthodes existantes proches à celles de ce travail sera abordée.

2.1 Optimisation bi-objectif et BIMADS

2.1.1 Définition d'un problème d'optimisation

L'optimisation est un domaine des mathématiques qui modélise des problèmes concrets sous forme de problèmes analytiques (cette branche est appelée programmation mathématique) ainsi que conçoit et analyse différents algorithmes et méthodes pour résoudre ces problèmes. En pratique, on définit une (ou des) fonction(s) objectif pour lesquelles on essaie de trouver un extrémum global ou le meilleur extremum local possible. On dispose donc d'une fonction $f : \mathcal{R}^n \longrightarrow \mathcal{R}$ pour laquelle on essaie de trouver un minimum (ou maximum) dans $\mathcal{R}$ donné par le point $x_{opt} \in \Omega$, où $\Omega \subseteq \mathcal{R}^n$.

Il existe donc deux types de problèmes en optimisation ; des problèmes de maximisation et des problèmes de minimisation. Ces deux problèmes sont équivalents car maximiser une fonction f revient à minimiser son opposée $-f$. Par convention, on utilise donc le standard de minimisation quand on parle d'un problème d'optimisation.

On recourt à l'optimisation dans de nombreux domaines, notamment en recherche opérationnelle, en ingénierie, biologie, économie, finance et de nom-

breux autres domaines traitant des problèmes quantitatifs.

Les variables du problème d'optimisation sont habituellement sujettes à certaines contraintes limitant les domaines de ces dernières. Ces contraintes traduisent des conditions du problème qui doivent être respectées. Une solution respectant les contraintes du problème est appelée réalisable. L'ensemble des solutions respectant toutes les contraintes est appelé ensemble admissible ou ensemble de solutions réalisables et est noté $\Omega \subseteq \mathcal{R}^n$. Les contraintes peuvent être séparées en deux types, d'égalité ou d'inégalité. Par convention pour une formulation d'un problème d'optimisation standard, on n'utilise que des contraintes d'inégalité. Une contrainte d'égalité peut être décomposée en deux contraintes d'inégalité comme suit :

$$c(x) = 0 \quad \Longleftrightarrow \quad c(x) \leq 0 \ \ \& \ \ -c(x) \leq 0 \tag{2.1}$$

ou peut parfois être utilisée pour éliminer une variable en l'isolant par rapport aux autres.

Il existe plusieurs types de problèmes d'optimisation selon la nature de la fonction objectif et de l'ensemble des contraintes. On peut trouver parmi ces types :

- l'optimisation convexe où f est une fonction convexe et les contraintes constituent un ensemble convexe [Boyd et al.2004] ;
- l'optimisation en nombre entiers où certaines variables sont entières [Wolsey and Nemhauser1999] ;
- l'opimisation non linéaire où f et/ou les contraintes sont non linéarires [Bazaraa et al.2013] ;
- l'optimisation stochastique où f ou les contraintes contiennent une composante aléatoire [Spall2005] .

2.1.2 Définition de l'optimisation bi-objectif

Dans certains problèmes industriels, nous avons besoin d'optimiser non pas une, mais deux ou plusieurs fonctions objectif qui souvent sont incompatibles ou conflictuelles. Par exemple, on souhaite concevoir un matériau pour la fabrication d'une aile d'avion, et nous avons besoin d'un matériau qui soit à la fois solide et léger.

Un problème biobjectif est noté de la manière suivante :

$$\min_{x\in\Omega} F(x) \quad = \quad (f_1(x), f_2(x)). \tag{2.2}$$

Contrairement à l'optimisation mono-objectif, il n'existe pas nécessairement de solution optimale qui minimise les deux fonctions objectif à la fois. Les solutions optimales du problème sont constituées d'un ensemble de points qui représentent des compromis entre les valeurs des deux fonctions objectif. Ces points ne sont pas comparables entre eux et constituent ce qu'on appelle un ensemble Pareto [Pareto1896]. L'image de l'ensemble Pareto dans l'espace (f_1, f_2) est appelé front Pareto. Ces points non comparables entre eux sont appelés points non dominés [Yu1974] et sont définis comme suit :

Définition 1

Soient $x, y \in \Omega$. On dit que x domine y si et seulement si $f_i(x) \leq f_i(y)$ pour $i \in \{1, 2\}$ et où au moins l'une des inégalités est stricte.
On écrit alors $x \prec y$.

Définition 2

Soit $x \in \Omega$. On dit que x est non dominé, ou Pareto optimal, si et seulement si $\nexists\ y \in \Omega$ tel que $y \prec x$.
L'ensemble des points non dominés est noté $\mathcal{P}$.

La démarche de ce travail fonctionne également en contexte multi-objectif avec plus de deux fonctions objectif car elle se concentre principalement sur une analyse de l'espace des variables. Cependant, on choisit de se concentrer

sur une approche bi-objectif pour trois raisons. La première est une facilité de visualisation des solutions sur un plan objectif en deux dimensions. En effet, il est difficile de visualiser de manière précise et fidèle un espace de dimension de plus de 3 sans passer par des méthodes de réduction de dimensionnalité. L'utilisation de ces méthodes introduit une perte d'information, et les relations spaciales entre les points en dimension réduite ne sont pas fidèles aux distances dans l'espace multidimentionnel d'origine. La deuxième raison est qu'il est possible d'ordonner des solutions de dimension 2 car l'ensemble $\mathcal{P}$ peut être ordonné par valeur décroissante de l'une des deux fonctions objectif qui est équivalent à un ordre croissant de l'autre fonction objectif. Cette notion d'ordre est perdue à partir de trois dimensions. Et la troisième raison est que les problèmes en boîtes noires ne sont que rarement compatibles avec l'optimisation ayant un grand nombre d'objectifs en raison de l'effort numérique requis [Zghal et al.2011, Audet and Hare2017]. En effet, en optimisation mono-objectif, on cherche à trouver un point optimal, en optimisation bi-ojectif on cherche le front Pareto qui s'agit d'une courbe dans un espace à deux dimensions, et avec plusieurs objectifs p, on cherche une surface à $p-1$ dimensions.

Du fait que le problème pratique du chapitre 4 se situe dans un contexte de boîtes noires, on procède à introduire ce dernier.

2.1.3 L'optimisation en boîtes noires :

Souvent dans des problèmes d'optimisation en industrie, les fonctions objectif peuvent ne pas être dérivables, leurs dérivées inaccessbiles ou encore la valeur des de ces dernières ainsi que des contraintes peuvent provenir de boîtes noires de fomulation analytique inconnue. Or les algorithmes d'optimisation classiques se basent sur ces dérivées pour identifier les directions à suivre pour trouver les solutions optimales. L'étude de ces problèmes fait l'objet de l'optimisation en boîte noires et l'optimisation sans dérivées [Audet and Hare2017, Conn et al.2009, Larson et al.2019].

Cependant il ne faut pas confondre les deux. En effet, l'optimisation sans dérivées est l'étude mathématique d'algorithmes d'optimisation qui n'utilisent pas de dérivées de la fonction objectif. L'optimisation de boîtes noires quant à elle est l'étude de la conception et analyse d'algorithmes d'optimisation qui assument que la fonction objectif et/ou les contraintes sont données par des boîtes noires. Pour la première, les dérivées peuvent exister mais ne sont pas disponibles. Pour l'optimisation en boîtes noires, les dérivées peuvent en effet exister et être disponibles [Audet and Hare2017].

Parmi des problématiques réelles qu'il n'est pas possible de résoudre avec des techniques d'optimisation classiques et où l'optimisation en boîtes noires a prouvé son utilité et efficacité on trouve le projet d'estimation de quantité d'eau mené par HydroQuébec [Alarie et al.2013]. Cette mesure se fait traditionnellement par des techniciens se déplaçant sur place pour étudier des carottes de glace. Cette méthode est très limitante par le fait que le territoire couvert est large (plus de 555000 km), que ça prend beaucoup de temps et que certaines zones inatteignables ne permettent pas d'y prendre des mesures. Ces difficultés ont été surmontées en développant un appareil appelé GMON qui permet de prendre des mesures quotidiennes automatiquement sans intervention humaine. Le problème est donc de placer ces appareils de manière stratégique sur le territoire pour minimiser l'incertitude des estimations. La fonction objectif provient d'un résultat de simulation et est considéré comme une boîte noire.

Un autre projet qui n'a pu être résolu que grâce à l'opimisation en boîtes noires est un problème de décollage d'avion [Torres et al.2011]. Il s'agit d'un problème biobjectif où il faut minimiser le bruit et la consommation de carburant. Les fonctions objectif sont les résultats d'une simulation par ordinateur pour différentes variables qui est considérée comme une boîte noire.

2.1.4 Les algorithmes MADS et BIMADS

MADS est l'acronyme pour *Mesh Adaptative Direct Search* [Audet and Dennis, Jr.2006, Audet and Hare2017]. MADS fait partie des algorithmes de recherche directe dont le premier a été le CS ou *Coordinate Search* [Fermi and Metropolis1952]. Ces algorithmes appliquent un treillis sur l'ensemble des paramètres, et l'algorithme n'a le droit de se déplacer que sur les points de ce treillis. Le treillis cependant peut changer de taille. Un algorithme de recherche directe est caractérisé par deux composantes, une phase de recherche globale et une phase de sonde locale. La phase de recherche est optionelle et permet de sortir de régions d'optima locaux en explorant des points plus loin dans l'espace de solutions que la zone de sonde. La phase de sonde quant à elle, permet de chercher dans une zone particulière de ce treillis autour d'un point appelé point de sonde, des points meilleurs que ceux explorés auparavant. Le treillis se resserre à chaque échec jusqu'à convergence. Les algorithmes de recherche directe se différencient entre eux suivant deux paramètres : la manière de sélectionner des points dans la phase de sonde, et la manière de mettre à jour la taille du treillis.

L'algorithme MADS est une généralisation et amélioration de l'algorithme *Generalized Pattern Search* ou GPS qui possède un meilleur résultat de convergence. MADS assure en effet, sous de légères hypothèses, la convergence vers un point stationnaire au sens du calcul non lisse de Clarke [Clarke1983]. Il permet aussi un plus large choix de directions pour la phase de sonde.

Il existe trois versions de l'algorithme MADS, à savoir LT-MADS [Audet and Dennis, Jr.2006] qui se base sur des matrices triangulaires inférieures aléatoires, QR-MADS [Van Dyke and Asaki2013] qui se base sur la décomposition QR et sur des directions normalement distribuées, et Ortho-MADS [Abramson et al.2009, Audet et al.2014] qui utilise des directions quasi-aléatoires déterministes et orthogonales.

L'algorithme BIMADS [Audet et al.2008] est une adaptation de l'algorithme

MADS au cas bi-objectif. BIMADS se base sur la décomposition du problème bi-objectif en un ensemble de sous-problèmes mono-objectifs que l'on résoud en utilisant MADS.

Il est important de noter que ces sous-problèmes ne sont pas combinés de manière pondérée. Au contraire, chaque solution d'un sous-problème nous fournit une approximation locale de l'ensemble Pareto que l'on combine. On sélectionne ensuite les points non dominés de cet ensemble combiné qui constituent les réelles solutions du problème et approximation de l'ensemble Pareto global.

Il existe également d'autres algorithmes se basant sur MADS pour la résolution de problèmes multi-objectifs [Audet et al.2010].

2.1.5 Définition de la segmentation :

L'apprentissage automatique est la science de la conception d'algorithmes capables d'effectuer des tâches n'ayant pas d'instructions explicites pour le faire [Bishop2006], ou qui améliorent leur performance au fur et à mesure qu'ils sont exécutés [Mitchell et al.1997, Cherkassky and Mulier2007]. Pour ce faire, l'algorithme apprend à partir de données fournies des représentations de l'espace de paramètres lui permettant d'inférer une solution. Il existe trois principaux types d'algorithmes d'apprentissage automatique [Ayodele2010] selon le type de données que l'on a et la tâche à effectuer :

- L'apprentissage supervisé : nous disposons de données sous forme de variables ainsi que du résultat attendu à partir de ces données. L'algorithme est entraîné à partir de ces données et leur résultats, puis construit un modèle mathématique qui fait des prédictions sur de nouvelles données dont il ne connaît pas le résultat [Russell and Norvig2002]. Des exemples d'utilisation de ces algorithmes incluent la reconnaissance d'images [Carneiro and Vasconcelos2005] ou de sons [Roman et al.2003] ou encore la prédiction de conditions climatographiques

[Hsieh2009].

- L'apprentissage non supervisé : l'algorithme a pour but d'identifier des structures à partir d'un jeu de données non labellisées, découvrant ainsi des relations entre les données permettant de regrouper celles partageant des caractéristiques similaires [Hinton et al.1999] . Ce type d'algorithmes est souvent utilisé pour créer des compagnes de marketing en identifiant les différentes classes de clients en les segmentant par leurs habitudes et pouvoir d'achat [Russell and Lodwick1999, Rajagopal et al.2011].

L'apprentissage automatique est utilisé dans tout type d'industrie et pour différents objectifs. Par exemple, en production pour la maintenance préventive ou le contrôle opérationnel, en marketing pour segmenter la clientèle et fournir à chacune les produits qui lui correspondent, en médecine pour la détection de maladies, en finance pour l'analyse de risques, ou encore pour optimiser l'offre à la demande et éviter la surproduction dans le domaine de l'énergie [Langley and Simon1995].

2.2 L'apprentissage non supervisé

Parmi les trois catégories de l'apprentissage automatique énoncées précédemment, l'apprentissage non supervisé nous intéresse en particulier car il tente de découvrir une structure inconnue inhérente aux données, nous permettant ainsi d'identifier des groupements de points partageant des caractéristiques similaires à partir des données appelés classes. En effet, à partir des points de l'ensemble Pareto, nous pouvons ainsi les séparer en classes et faire un choix d'une solution appartenant à l'une des classes qui nous intéresse. Cette séparation des points se fait de manière à ce que chaque classe regroupe des points similaires.

On procède à présent à définir les algorithmes de segmentation qui sont utilisés dans ce travail.

L'algorithme Kmeans

Le premier est Kmeans, qui est un algorithme itératif nommé ainsi par MacQueen en 1967 [MacQueen et al.1967]. Cet algorithme présuppose un nombre de classes prédéfini représenté par le 'K' du nom 'Kmeans' ou K-moyennes. Cet algorithme est souvent utilisé comme référence de performance car il est rapide à exécuter et n'a besoin que d'un paramètre : le nombre de classes attendu. Cependant, Kmeans fait des hypothèses strictes sur les données :

- la variance de chaque variable est sphérique,
- toutes les variables ont la même variance.

Ceci cause un risque échec de l'algorithme si la distribution des points n'est pas sphérique ou que la densité des classes est différente. Kmeans est formulé dans l'algorithme 1.

On dispose de N points $\{x_1, .., x_N\} \subset \mathcal{R}^n$ que l'on essaie de séparer en classes.

Algorithme 1 Kmeans

Entrée: *cache* : C

1: *Considérer un nombre de classes* K
2: *Initialiser* K *centroïdes* $\{c_1, .., c_K\}$
3: **faire**
4: $\quad s = \sum_{k=1}^{K} \sum_{i=1}^{N} (x_i - c_k)^2$
5: $\quad$ *Assigner chaque point* x_i *au centroïde* c_k *le plus proche*
6: $\quad$ *mettre à jour les centroïdes*
7: **tant que** *changement de centroïdes*

Le but de la méthode est de minimiser la variance au sein de chaque classe qui n'est autre que la somme des distances au carré de chaque point au centroïde de la classe y étant associé.

Pour l'algorithme Kmeans, il est essentiel de déterminer le nombre de classes à l'avance. Ce choix est critique pour la performance de l'algorithme.

La phase d'initialisation de centroïdes est importante. Une technique naïve

pour le faire est de les placer aléatoirement. Cette approche a toutefois certains défauts. Puisque que Kmeans est une heuristique, le défaut majeur est un risque d'être coincé sur un mauvais minimum local donnant une mauvaise segmentation dépendamment du placement initial des centroïdes. En effet, la segmentation finale peut être différente dépendamment du placement initial des centroïdes et de la distribution des données. Un exemple pourraît être des centroïdes arbitrairement proches dans une région dense de l'espace et aucun centroïde dans une autre région dense de cet espace. La méthode d'initialisation Kmeans++ [Arthur2006] présentée par Arthur résout ce problème et a été adoptée pour l'initialisation dans ce travail. Cette méthode consiste à garantir des points espacés dans l'espace de paramètres. Le déroulement de cette méthode est présenté dans l'algorithme 2.

Algorithme 2 initialisation Kmeans++

1: *Choisir un centroïde initial parmi les données suivant une distribution uniforme*
2: **pour** $K-1$ *itérations* **faire**
3: *calculer la distance* d_x *entre chaque point* x *des données et le centroïde le plus proche*
4: *Choisir un nouveau point des données suivant une distribution pondérée par* d_x^2 *pour chaque point* x
5: **fin pour**

L'algorithme Meanshift

Le deuxième algorithme utilisé est Meanshift qui a été introduit en 1975 par Fukunaga et Hostetler [K. Fukunaga1975]. Cet algorithme est principalement utilisé en vision par ordinateur pour des tâches de segmentation, mais est également très utilisé pour la segmentation. Meanshift se base sur le principe d'estimation de densité par noyau qui est une méthode non paramétrique pour estimer des densités de probabilité. En effet, Meanshift considère l'espace de paramètres comme une fonction de densité de probabilité dont les points qui

lui sont fournis sont considérés comme un échantillon. L'algorithme considère que s'il existe des groupements dans les données, ils correspondent aux modes ou maxima locaux de cette fonction de densité. Le déroulement de l'algorithme consiste à déplacer les points itérativement vers le mode ou la région de plus grande densité la plus proche.

À l'opposé de Kmeans, Meanshift ne requiert pas de spécifier un nombre de classes à l'avance, il ramène chaque point vers le mode le plus proche. On obtient ainsi une classe pour chaque mode trouvé par l'algorithme. Cependant, Meanshift requiert comme paramètre le rayon h de la fenêtre à l'intérieur de laquelle on calcule une moyenne pondérée qui correspond à la variance du noyau gaussien que l'algorithme utilise. Le déroulement de Meanshift est relativement simple et est précisé dans l'algorithme 3.

Les avantages de Meanshift autres que le fait qu'il ne requière pas de nombre de classes prédéfini sont qu'il est robuste aux données aberrantes et ne comporte qu'un hyperparamètre h. Ses inconvénients sont que les résultats varient beaucoup dépendamment du paramètre h, a une grande complexité de calcul et sa performance diminue quand le nombre de dimensions grandit.

Algorithme 3 Meanshift

Sélectionner une fenêtre F
pour chaque point p du jeu de données **faire**
    tant que pas encore convergé **faire**
        calculer la moyenne pondérée des points dans une fenêtre F
        de rayon h autour du point p à l'aide d'un noyau gaussien
        de variance h.
        déplacer la fenêtre F pour la centrer sur la nouvelle moyenne
    fin tant que
fin pour

L'algorithme HDBscan

Le troisième algorithme qui est utilisé dans ce travail est HDBscan [Campello2013]. Cet algorithme est une version améliorée de l'algorithme DBscan qui est un algorithme de segmentation basé sur la densité. DBscan est un sigle de "Density-Based Spatial Clustering of Applications with Noise" et a été proposé par Ester, Kriegel, Sander et Xu en 1996 [Ester et al.1996]. DBscan fait partie des algorithmes qui segmentent en se basant sur la densité. Il est approprié pour identifier les régions de faible densité, et les assigner comme bruit, c'est-à-dire, points pas assez nombreux pour créer une classe. Cette proptiété le rend plus adapté pour les données de nature bruitée ou contenant des points aberrants dans le sens où l'espace est constitué de classes mais aussi beaucoup de points dispersés qui ne font partie d'aucune classe. Ces points interfèrent avec les autres algorithmes de segmentation qui tentent de segmenter tous les points de l'espace baissant ainsi leur performance. Cet algorithme a aussi l'avantage de ne faire que peu d'hypothèses sur la structure des données comparé à Kmeans par exemple dont la qualité de performance dépend de la validité de ces hypothèses. HDBscan [Campello2013] est une extension de DBscan ajoutant une nouvelle couche de segmentation hiérarchique améliorant les performances de l'algorithme qui permet de le rendre plus stable. Le désavantage de HDbscan comme son prédécesseur DBscan est qu'il est très sensible à ses hyperparamètres. Une légère modification de ceux-ci peut donner des résultats différents ou même d'échouer (assigner tous les points à une classe, ou considérer tous les points comme étant du bruit). Il en possède cinq, dont les deux plus importants sont :

- la taille minimale d'une classe qui dicte le nombre minimal de points requis pour former une classe,
- le nombre d'échantillons permet de réguler à quel la segmentation est conservative. Une grande valeur pour ce paramètre se traduit par une plus grande rigueur à considérer un point comme bruit,

Les étapes d'exécution de HDBscan sont exposés dans l'algorithme 4.

Algorithme 4 HDBscan

1: *Transformer l'espace suivant la densité.*
2: *Construire un arbre couvrant minimal à partir du graphe pondéré.*
3: *Construire une hiérarchie de segmentation à partir des composants connectés.*
4: *Condenser cette hiérarchie en se basant sur la taille de classe minimale.*
5: *Extraire les classes stables de l'arbre condensé.*

L'algorithme spectral

Le dernier algorithme de segmentation utilisé dans ce travail est l'algorithme de segmentation spectrale. Il apparaît pour la première fois entre les années 1969 et 1973 [Cheeger1969, Fiedler1973]. Cependant deux travaux qui ont popularisé cet algorithme comme méthode de segmentation utilisée en intelligence artificielle ont été publiés par Shi et Malik [Shi and Malik2000] en 2000 et Ng, Jordan et Weiss [Ng et al.] en 2002. C'est un algorithme qui a gagné en popularité pour des tâches de segmentation par sa simple implémentation et ses hautes performances. L'algorithme spectralutilise des principes de théorie de graphes spectraux [Chung and Graham1997] pour effectuer la segmentation en construisant un graphe de similarité sur les données et segmentant sur les valeurs propres de la matrice laplacienne de ce graphe. Il existe différentes implémentations de cet algorithme chacune utilisant une manière différente de construire la matrice laplacienne du graphe de similarité. L'avantage de cet algorithme est aussi sa robustesse quand le nombre de dimensions est grand car, en construisant le graphe de similarité, on réduit la dimensionnalité de manière non-linéaire. Il permet ainsi de capturer même les géométries non linéaires de l'espace, lui permettant d'avoir une meilleure performance sur une large variété de problèmes que les algorithmes de segmentation qui segmentent de manière linéaire comme Kmeans. Un inconvénient considérable de cet algorithme est la perte de données que la construction du graphe de similarité introduit. Une comparaison entre les algorithmes sera discutée et

démontrée au chapitre 3. Une description des étapes de l'algorithme spectral est présentée dans l'algorithme 5.

Algorithme 5 Segmentation spectrale [Von Luxburg2007]

1: *Cinsidérer un nombre de classes k*
2: *Construire le graphe de similarité. Soit W sa matrice d'adjascence pondérée*
3: *Calculer la matrice laplacienne normalisée L à partir de W*
4: *Calculer les k premiers vecteurs propres* $u_1, ..u_k$ *de L*
5: *Construire la matrice T à partir des* u_i *comme colonnes puis normaliser ses lignes suivant la norme 1*
6: *Effectuer une segmentation sur les lignes de la matrice T (par Kmeans par exemple)*

2.3 L'optimisation multi-critère

La procédure développée dans ce travail touche au domaine de l'optimisation multi-critère qui est très liée à l'optimisation multi-objectif. En effet, comme mentionné précedemment les algorithmes d'optimisation multi-objectifs donnent un ensemble de solutions optimales qui offrent un compromis entre les fonctions objectif, et il est donc difficile d'en choisir une. L'optimisation multi-critère [Keeney et al.1993, Belton and Stewart2002] est la science dédiée à proposer des méthodes de choix entre ces solutions de la manière qui correspond au mieux aux attentes des décideurs. Ces méthodes sont utilisées pour des problèmes de décision dans plusieurs domaines comme le génie civil [Seo and Sakawa2012], en médecine [Marsh et al.2016] ou encore en socio-économie [Zhang et al.2014] entre autres.

Certains auteurs comme Zimmermann [Zimmermann2012] suggèrent de décomposer cette analyse multi-critère en deux catégories, l'analyse de décision multi-objectif et l'analyse de décision multi-attributs se concentrant sur des espaces de décision continus et discrets respectivement. Notre étude se place

dans la première catégorie.

Il existe beaucoup de méthodes pour effectuer ce choix de solution parmi les solutions Pareto. Certaines méthodes proposent de réduire le nombre de solutions non dominées au cours du processus d'optimisation multi-objectif comme les travaux de Rosenman [Rosenman and Gero1985] qui étend les travaux de Morse [Morse1980], Steuer [Steuer and Harris1980] et Torn [Törn1980].

D'autres proposent une analyse post-optimisation avec pour objectif analyser le front Pareto pour en sélectionner un point. Parmi ces méthodes d'analyse technique de la courbe Pareto on trouve les travaux de Yu [Yu and Leitmann1974] et Zeleny [Zeleny1973] qui proposent de sélectionner le point sur la courbe formée par Front Pareto le plus proche à un point de référence. Ce point de référence peut être défini comme le point théorique ayant pour coordonnées les valeurs minimales des fonctions objectifs $(\min(f_1), \min(f_2), .., \min(f_n))$ pour un problème à p objectifs. Une autre méthode proposée par Miettinen en 1999 [Miettinen2012] consiste à calculer l'amélioration maginale d'une fonction objectif en réduisant d'une unité une des autres fonctions objectifs. Le point retenu est celui ayant une amélioration marginale maximale.

Finalement, une dernière catégorie proposent d'utiliser des méthodes d'intelligence artificielle pour choisir des solutions Pareto. Taboada [Taboada and Coit2006] propose une méthode de segmentation des points non-dominés que ce travail tente d'améliorer en l'adaptant au contexte industriel et en résolvants certaines de ses limitations. La méthode proposée par Taboada consiste à utiliser l'algorithme Kmeans sur les solutions non-dominées trouvées par un algorithme d'optimisation multi objectif évolutionnaire (MOEA) [Zitzler and Thiele1999]. D'autres travaux tels que ceux de Maulik [Maulik et al.2009] proposent également d'utiliser une segmentation pour identifier des solutions qui les intéressent. Les travaux de Maulik cependant ont été développés spécifiquement pour résoudre des problèmes d'optimisation dont l'objectif est de

segmenter des gènes. La méthode propose toutefois d'utiliser l'information contenue dans les valeurs des variables pour segmenter les solutions Pareto. Rosemann [Rosenman and Gero1985] propose également d'utiliser des méthodes de segmentation à des fins d'analyse de l'ensemble Pareto. Sa méthode propose de réduire le nombre de solutions Pareto en utilisant des méthodes de segmentation hiérarchique tout en restant fidèle à la forme du front Pareto. La méthode utilise également des méthodes de programmation dynamique pour inclure cette segmentation à l'algorithme d'optimisation et ainsi filtrer les solutions Pareto au cours de l'exécution de l'algorithme d'optimisation. Agrawal, Panigrahi et Tiwari [Agrawal, Shubham and Panigrahi, Bijaya Ketan and Tiwari2008] proposent une méthode utilisant des méthodes de segmentation d'une manière différente des méthodes précédentes. Ils utilisent la segmentation floue pour guider l'algorithme de segmentation à chercher des solutions dans des zones non explorées du front Pareto tentant de remplir les trous.

CHAPITRE 3 MÉTHODOLOGIE

Ce chapitre propose une méthodologie pour analyser un front Pareto et comment justifier le choix de certaines solutions par rapport à d'autres. Il est possible de penser à deux manières pour filtrer les solutions Pareto et n'en garder que quelques unes qui soient intéressantes : réduire leur nombre durant l'exécution de l'optimisation (comme il en est question dans [Rosemann and GERO1985]), ou utiliser les résultats de l'optimisation afin de sélectionner parmi les points trouvés ceux qui sont intéressants.

Ce travail se situe dans ce deuxième contexte.Taboada adopte une appoche similaire à celle proposée [Taboada and Coit2006]. Ce travail tente de proposer des solutions aux limitations de cette méthode tout en améliorant les étapes de cette dernière. Plus spécifiquement, il est utilisé dans cette méthodologie des algorithmes d'optimisation bi-objectif avec de meilleures performances [Fowler et al.2008] que NSGA-II et autres algorithmes évolutionnaires conseillés par [Taboada and Coit2006]. On propose également l'utilisation de quatre algorithmes de segmentation différents dont le choix dépendra de la nature du problème plutôt que toujours utiliser Kmeans qui risque de donner de mauvais résultats pour des problèmes ne satisfaisant pas ses hypothèses, permettant à la méthodologie d'être applicable à un plus large éventail de problèmes.

De plus, la manière dont cette approche est innovante par rapport à celle de Taboada et beaucoup d'autres méthodes existantes est qu'elle ne fait pas la segmentation sur les valeurs des fonctions objectif car il existe de nombreuses manières de le faire, mais se base plutôt sur les valeurs des variables pour séparer les solutions en classes. L'intuition derrière notre approche est qu'elle permet ainsi au décideur de percevoir des solutions dans différentes régions du front Pareto qui ont des caractéristiques de variables similaires si elles existent ou encore d'identifier instantanément sur le front Pareto toutes les solutions partageant des caractéristiques de variables communes si ces caractéristiques

sont présentes pour un nombre suffisant de solutions.

Cette information peut être d'une grande utilité dans certains types de problèmes. Par exemple, si le décideur cherche des solutions stables par rapport aux variations des valeurs des variables, il choisira plutôt parmi les classes qui n'ont pas d'éléments dans différentes régions de l'espace des fonctions (f_1, f_2). Si au contraire il est intéressé par des solutions répondant à des caractéristiques précises, il pourra choisir parmi les différentes régions où une classe répondant à ses critères est présente si ces régions existent. Une autre manière dont cette approche est innovante est qu'elle permet de choisir non seulement parmi les solutions non-dominées, mais potentiellement aussi des solutions proches de celles-ci qui pourront avoir des critères plus intéressants pour le décideur que les solutions Pareto. En effet les solutions Pareto peuvent être difficiles à réaliser, très coûteuses ou peu stables (une légère variation d'une ou plusieurs variables causant une grande baisse ou augmentation des fonctions objectif), ou encore trop peu nombreuses pour que les algorithmes de segmentation puissent produire de bons résultats (le nombre de points requis pour avoir une bonne segmentation augmente exponentiellement avec la dimensionnalité du problème).

La méthodologie proposée permet de pallier à ces problèmes en relaxant l'ensemble de points non-dominés permettant ainsi de maîtriser le nombre de points disponibles pour à la fois en avoir assez pour une bonne performance des algorithmes de segmentation, mais aussi pas trop nombreux causant un temps d'exécution trop grand pour ces derniers. Cette méthode permet également d'identifier des points ne faisant pas partie des solutions Pareto mais répondant mieux aux critères des décideurs que ces dernières tout en étant proches de l'optimalité. La méthodologie abordée est divisée en quatres étapes :

- génération et préparation des données ;

- choix d'un algorithme de segmentation ;

- présentation des classes et de leurs caractéristiques ;

- heuristiques de choix de représentants par classe.

La première partie décrit la manière et la nature des données traitées ainsi que les phases de préparation nécessaires pour les étapes suivantes. La deuxième partie compare et discute le choix d'un algorithme de segmentation

3.1 Génération et préparation de données

On commence par la cache d'un algorithme d'optimisation bi-objectif. Une cache est définie comme l'ensemble des solutions réalisables que l'algorithme a parcouru durant son exécution. Elle inclut ainsi la valeur des variables ainsi que celle des fonctions objectif et des contraintes. Dans ce qui suit, on utilisera la définition mathématique d'une cache suivante :

Définition 3
La cache associé à l'application d'un algorithme à un problème d'optimisation est l'ensemble :

$$C = \{(x, f(x), c(x)) \in \mathcal{R}^n \times \mathcal{R}^2 \times \mathcal{R}^m : x \in \{x^1, x^2, \ldots, x^p\}\}$$

où $x^1, \ldots, x^p$ sont les points visités par l'algorithme.

Avant de manipuler les données, il faut au préalable les normaliser. En effet, souvent on se retrouve avec des variables et fonctions objectif d'ordres de grandeur très différents et les algorithmes de classification sont sensibles aux changement d'échelle Une méthode de mise à l'échelle des solutions s'impose donc. On a le choix entre effectuer une normalisation ou une standardisation. Une normalisation permet de rééchelonner les données dans l'intervalle $[0, 1]$. Alors qu'une standardisation encode les données suivant une loi normale centrée réduite en déduisant la moyenne et divisant par l'écart type des données.

Pour effectuer une normalisation, on utilise la relation suivante pour chacune des valeurs de chacune des variables puis pour les fonctions objectif du problème d'optimisation :

$$x_{i,j,norm} = \frac{x_{i,j} - x_{i,min}}{x_{i,max} - x_{i,min}}, ß \in \{1, .., p\}, j \in 1, .., n$$

avec $x_{i,j,norm}$ est la nouvelle valeur normalisée de la variable x_i au point j, $x_{i,min}$ est la valeur minimale de cette variable du jeu de données, et $x_{i,max}$ la valeur maximale. Cette transformation nous garantit que toutes les variables ainsi que les valeurs des fonctions objectif sont à la même échelle entre 0 et 1. Il faut cependant faire attention à ce que chaque variable contienne au moins deux valeurs différentes pour ne pas diviser par zéro. Toutefois, on ne risque pas de tomber dans ce cas car cela indiquerait une mauvaise définition du problème d'optimisation et que l'une des variable soit inutile.

Pour effectuer une standardisation, on utilise plutôt la relation suivante :

$$x_{i,j,std} = \frac{x_{i,j} - \mu_i}{\sigma_i}, i \in \{1, .., p\}, j \in 1, .., n$$

avec $x_{i,j,std}$ est la valeur standardisée de la variable x_i au point j, μ_i est la moyenne de la variable x_i et σ_i son écart type. Cette transformation nous garantit que les valeurs des variables et fonctions objectifs suivent une loi normale contrée réduite.

Le choix de la standardisation ou normalisation est guidé par la nature du problème. Une normalisation sera préférée quand la distribution des variables ne suit pas une loi gaussienne ou est de distribution inconnue ainsi que si les données ne contiennent pas des valeurs extrêmes (quelques valeurs d'une variable d'un ordre de grandeur différent des autres valeurs pour cette même variable), alors qu'une standardisation justement lorsque ces valeurs extrêmes

sont présentes ou si on sait les données suivent une loi guaussienne.

Losqu'on travaille avec une cache produite par un algorithme d'optimisation, cela présente un avantage considérable qui consiste à disposer dès le début de beaucoup de points prometteurs et intéressants s'il a fait suffisamment d'itérations, car la distribution des points sera possiblement proche de la région optimale vers la fin de son exécution. Cependant, il est également possible de commencer avec un ensemble aléatoire de points, mais il faudra en extraire un sous-ensemble proche de la région Pareto pour garantir la pertinence de l'étude de segmentation. En effet, le but étant d'ultimement segmenter les points Pareto en régions distinctes sur l'espace objetif suivant leurs caractéristiques dans l'espace des variable, il sera vain et inutile d'utiliser la totalité d'un ensemble aléatoire de points car on risque de biaiser la segmentation avec les caractéristiques de points loins de la région optimale. Pour obtenir un ensemble de points intéressants, on commence par extraire les points non-dominés correspondant aux solutions quasi-Pareto en utilisant l'algorithme 6.

La complexité en temps de calcul en pire cas de cet algorithme est $O(n^2)$ qui est assez élevée si on dispose d'un très grand nombre de données. Cependant, en contexte de boîtes noires, il est rare qu'on ait un nombre de points qui soit très grand car la génération des points prend un temps considérable, et le défi de ce type d'optimisation ainsi que des algorithme de résolutions des problèmes en boîtes noires tels que MADS est de trouver de bonnes solutions en peu d'itérations. Ce qui fait qu'on dispose souvent d'un nombre raisonnable de points pour que $O(n^2)$ ne soit pas un inconvénient. À titre d'exemple, pour le problème de chapitre suivant, la résolution du problème d'optimisation a pris une semaine pour générer 50 000 points, l'algorithme 6 a pris 15 minutes pour extraire les points non dominés.

Algorithme 6 Extraction des points non dominés

Entrée: *cache* : C

fonction ExtractPareto
  $P = \emptyset \leftarrow$ *Front Pareto*
  pour $x \in C$ **faire**
    si x est non dominé dans C et $x \in \Omega$ **alors**
      $P \leftarrow P \cup \{x\}$
    fin si
  fin pour
  retourne P
fin fonction

On étend ensuite cet ensemble qui peut être très petit en y ajoutant des points proches comme montré par l'algorithme 7. Cette étape répond à l'une des deux grandes problématiques qui motivent ce traavil. Le but d'une telle relaxation du front Pareto est de permettre à l'utilisateur d'explorer non seulement les solutions optimales, mais également les solutions se trouvant dans des régions proches du front Pareto. Cet algorithme prend en entrée la cache C et l'ensemble P extrait par l'algorithme 6. Une technique qui peut être utilisée est de prendre tous les points se trouvant à un certain rayon dans l'espace objectif $\mathcal{F} \subset \mathcal{R}^2$ des autour des points non dominés, et agrandir ce rayon jusqu'à avoir un nombre satisfaisant de points. L'algorithme 7 utilise la boule de rayon r centrée en $F(x)$ et on la note $\mathcal{B}_r(F(x))$. Cet algorithme a une complexité en temps de calcul linéaire de $O(n)$, qui est très rapide.

La figure 3.1 montre l'approximation obtenue par cette méthode. On voit à gauche les points du Front Pareto obtenu par BIMADS et à droite l'ensemble des points relaxés en incluant tous les points à un certain rayon à partir des points Pareto.

Algorithme 7 Relaxation du front Pareto

Entrée: cache C, Pareto P, rayon r

fonction RELAXPARETO
    $R = \emptyset \leftarrow$ *front relâché*
    pour x *dans* P **faire**
        $R \leftarrow R \cup (C \cap \mathcal{B}_r(F(x))$
    fin pour
    retourne R
fin fonction

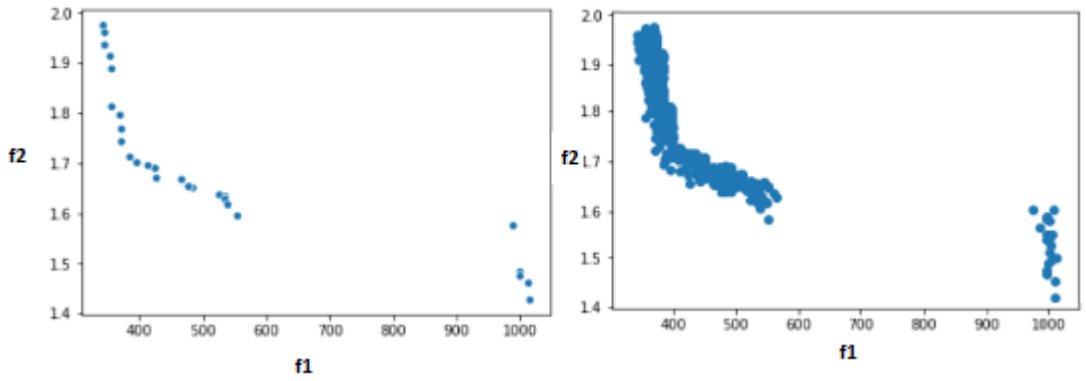

Figure 3.1 Extraction de points Pareto

Le rayon de la boule $\mathcal{B}_r(F(x))$ est choisi selon par l'utilisateur pour exlorer des zones éloignées du front Pareto. Une valeur de zéro pour ce rayon ne gardera strictement que les points Pareto, et plus on augmente le rayon, plus on s'éloigne de front pareto. Nous procédons par la suite au choix d'un algorithme de classification adapté à la nature du problème.

3.2 Choix de l'algorithme de classification

Dans ce projet, nous avons choisi et comparé les quatre algorithmes de segmentation présentés dans le chapitre précédent. Une des raisons pour lesquelles on a choisi ces quatre en particulier est la suivante. Il existe plusieurs

types d'algorithmes de segmentation [Xu and Wunsch2005], on se concentre ici sur deux de ces types qui nous intéressent pour la tâche à effectuer et qui englobent une partie des sous-types cités : les algorithmes se basant sur la compacité et ceux qui s'appuient sur la connectivité.

La compacité est définie ici comme regroupant des points qui sont proches suivant une mesure de distance. Dans notre cas on a choisi Kmeans car c'est un algorithme de base qui servira de référence ainsi que Meanshift.

En contraste avec la compacité, la connectivité est définie comme regroupant des points qui sont « connectés » ou immédiatement à côté entre eux. Deux points peuvent avoir une distance moindre entre eux qu'avec d'autres points de leurs voisinnages mais ne sont pas regroupés ensemble. Dans notre cas, on utilise HDBscan et l'algorithme de segmentation spectrale dans cette famille.

Le choix d'utilisation de l'un parmi ces quatres algorithmes est justifié par la nature du problème. Kmeans s'exécute très rapidement avec de bons résultats tant que ses hypothèses sont vérifiées, cependant il ne permet pas d'identifier et exclure d'éventuelles points aberrants ou zones de très faible densité. HDBscan étant un algorithme de segmentation par densité, a été développé spécifiquement pour être adapté quand les données contiennent des points aberrants (points assez loin des classes pour ne pas y appartenir et qui réduisent l'efficacité des autres algorithmes de segmentation qui tentent d'assigner tous les points à une classe tels que Kmeans et l'algorithme spectral), toutefois il ne marche pas aussi bien quand les classes ont des densités différentes ou que l'on doit segmenter dans un espace avec un grand nombre de dimensions. Meanshift est également robuste au points aberrants et a l'avantage considérable qu'il ne fait aucune hypothèse sur la forme de la variance a priori des points (sphérique, elliptique etc..) tel que Kmeans, néanmoins il ne marche pas bien non plus en hautes dimensions. L'algorithme spectral est adapté aux données dans un espace en hautes dimensions du fait qu'il effectue une réduction de dimensionnalité lors de la création du graphe de similarité, cependant son temps d'exécution evolue exponentiellement avec le nombre de

points, et introduit une perte d'information justement en créant le graphe de similarité.

On peut donc voir qu'il serait plus judicieux d'adopter l'un ou l'autre des algorithmes selon la nature du problème. Une autre raison pour le choix de ces quatre algorithmes de segmentation en particulier est justement pour que chacun puisse combler les lacunes d'un autre. Si la nature du problème n'est pas connue, il est suggéré d'utiliser les quatres algorithmes et comparer leurs résultats.

Un autre point important dont il faut se rappeler est que HDBscan et meanshift sont très sensibles à leurs hyperparamètres. Afin d'illustrer cela, on effectue une segmentation par les quatre algorithmes sur un jeu de données avec leurs paramètres de base, puis on compare les résultats obtenus.

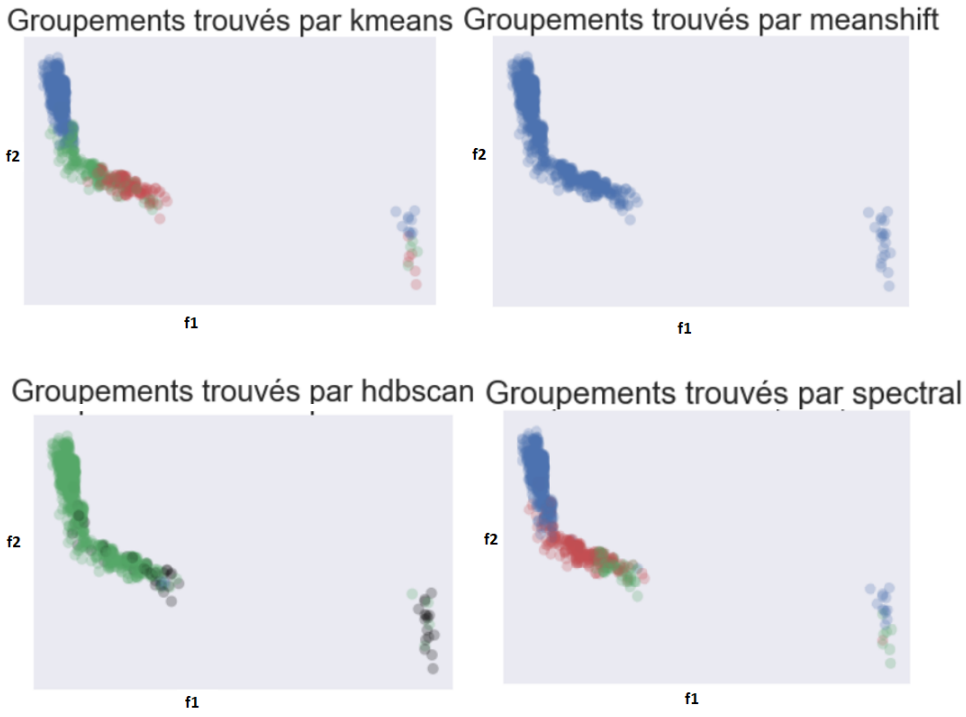

Figure 3.2 Comparaison des classes trouvées par chaque algorithme

On remarque que Meanshift et HDBscan ont des résultats médiocres comparé à Kmeans et l'algorithme spectral. Ceci ne veut pas dire que Meanshift et

HDBscan sont de mauvais algorithmes, mais plutôt que les hyperparamètres utilisés ne sont pas adaptés à ce problème. Il faut donc fournir un effort d'optimisation de ces derniers pour les adapter à chaque problème. Dans le chapitre suivant à la figure 4.6, une deuxième comparaison de ces algorithmes sera faite avec cette fois-ci des hyperparamètres optimisés et adaptés au problème pour démontrer la validité de cette déclaration.

3.3 Présentation des classes et de leurs caractéristiques

L'algorithme spectral fait partie des algorithmes de segmentation qui requièrent de leur fournir un nombre de classes souhaité au préalable. Il faut donc déterminer un nombre de classes pour chaque problème a priori. Pour ce faire, on utilise la méthode d'analyse de silouhette présentée dans l'article [Ng and Weiss2002] et développée par Rousseeuw en 1987 [Rousseeuw1987] qui utilise une mesure de validité de classes interne appelée l'index de silhouette.

La méthode d'analyse de silhouette est une méthode de validation de la cohésion et séparation des classes. En effet, un score est attribué à chaque point mesurant sa similitude aux autres points de la même classe comparé à son appartenance à une autre classe. Le déroulement de la méthode de l'analyse de silhouette présenté dans l'algorithme 8 est comme suit :

D'abord, on assigne un score de silhouette $S(x_i)$ à chaque point.

On note $a(x_i)$ la dissimilitude du point x_i avec les points de la même classe. C'est à dire la moyenne des distances entre x_i et tous les points de la classe à laquelle il a été assigné. Et $b(x_i)$ la dissimilitude du point x_i avec les points de la classe la plus proche à x_i. La classe la plus proche à x_i est celle du point le plus proche de x_i qui n'appartient pas à la même classe. Pour ce qui suit, on note $c(x)$ la classe assignée au point x et $\mathcal{C}$ l'ensemble des classes.

Ces quantités sont calculées de la manière suivante :

$$a(x_i) = \frac{1}{n_1} \sum_{j=1}^{n_1} d(x_i, x_j) \qquad x_j, j \in \{1, 2, ..n_1\} \textit{ tel que } c(x_j) = c(x_i) \quad (3.1)$$

$$b(x_i) = \frac{1}{n_2} \sum_{j=1}^{n_2} d(x_i, x_j) \quad x_j, j \in \{1, 2, ..n_2\} \textit{ points de la classe la plus proche à } x_i \quad (3.2)$$

$S(x_i)$ est calculé de la manière suivante :

$$S(x_i) = \frac{b(x_i) - a(x_i)}{\max\{a(x_i), b(x_i)\}} \quad (3.3)$$

Ensuite, on calcule la moyenne de ces scores pour chaque classe et enfin la moyenne de ces dernières pour avoir un score final.

Algorithme 8 Analyse de silhouette

Entrée: Ensemble de classes : $\mathcal{C} = \{\mathcal{C}_k\}, k \in \{1..p\}$

1: **fonction** ANALYSE DE SILHOUETTE POUR UNE CLASSE
2: **pour** $\mathcal{C}_k \in \mathcal{C}$ **faire**
3: **pour** $x \in \mathcal{C}_k$ **faire**
4: $a(x) = \frac{1}{n_1} \sum_{i=1}^{n_1} d(x, x_i)$ $n_1 = card(\mathcal{C}_k)$
5: $b(x) = \frac{1}{n_2} \sum_{i=1}^{n_2} d(x, x_i)$ $n_2 = card(\mathcal{C}_j)$ *avec* $\mathcal{C}_j$ *la classe la plus proche de* x
6: $S(x) = \frac{b(x)-a(x)}{\max\{a(x),b(x)\}}$
7: **fin pour**
8: $S_k = \frac{1}{n_1} \sum_{i=1}^{n_1} S(x_i)$
9: **fin pour**
10: $S_{moy} = \frac{1}{p} \sum_{i=1}^{p} S_p$
11: **retourne** S_{moy}
12: **fin fonction**

Le score de silhouhette prend des valeurs entre -1 et 1. Quand il est proche de -1, le point serait mieux classifié dans une classe voisine et son attribution

actuelle n'est pas pertinente. Quand il est autour de 0, cela indique que ça ne change pas si le point est attribué à cette classe ou à l'une des classes voisines. Quand le score est proche de 1, cela indique que le point est bien assigné.

La méthodologie de choix d'un nombre de classes optimal consiste à faire une analyse de silhouette sur les données pour différents nombres de classes, et choisir le nombre qui nous donne la plus grande moyenne d'index de silhouette.

Après déroulement de l'algorithme, on affiche les figures des classes colorés dans l'espace (f_1, f_2) et on dresse un tableau récapitulatif des statistiques pour chaque classe. Ce dernier a pour but de permettre à l'expert de vérifier que les classes font en effet du sens. Ce tableau récapitulatif comporte pour chaque classe et pour chaque variable les statistiques de base suivantes :

- Le nombre de point contenus dans la classe ;
- La moyenne ;
- L'écart-type ;
- La valeur minimale ;
- La valeur maximale ;
- Les 3 quartiles.

Ces statistiques permettent d'avoir une idée sur la taille de la classe, sa variabilité et dispersion dans l'espace grâce à l'écart-type et à l'écart interquartile ainsi que son intervalle de valeurs grâce aux valeurs minimales et maximales.

3.4 Heuristiques de choix de représentants par classe

Pour finir et pour atteindre notre objectif de sélectionner des solutions pertinentes, on doit développer des heuristiques qui nous permettent d'élire un ou des points représentants pour chacune des classes, et ainsi avec le tableau récapitulatif, l'utilisateur pourra choisir les solutions qui lui conviennent.

Deux heuristiques ont été développées pour répondre à cet objectif, la première est construite juste à partir des données et résultats de la méthodologie sans introduction d'information ou critère de choix par l'utilisateur et se veut

être une heuristique naïve proposant une solution rapide sans effort fourni par l'utilisateur, la deuxième cependant requiert des informations fournies par le décideur et a été développée spécifiquement pour répondre au deuxième aspect de la problèmatique que la méthodologie tente de résoudre, à savoir valoriser l'expertise de l'utilisateur en lui permettant de filtrer les solutions suivant ses critères. On pourra voir des exemples pratiques de chacune des deux heuristiques dans le chapitre suivant.

Heuristique 1 : Pour chaque classe, on choisit le point Pareto le plus proche du centroïde de la classe et inclus dans la classe. Ce point constitue un bon candidat car c'est le point Pareto qui a la plus petite distance moyenne avec tous les autres points de la classe. Cette heuristique cependant risque d'échouer si la classe n'est pas convexe ou même est discontinue dans l'espace (f_1, f_2). Dans ce cas, il sera peut être plus pertinent d'élire plus d'un représentant (un pour chaque partie discontinue) ou de n'en élire un que dans la partie discontinue de la classe la plus dense (contenant le plus gand nombre de points).

Algorithme 9 Heuristique 1

Entrée: Ensemble de classes : $\mathcal{C} = \{\mathcal{C}_k\}$, ensemble Pareto P

1: **fonction** HEURISTIQUE 1
2: **pour** $\mathcal{C}_k \in \mathcal{C}$ **faire**
3: $c_k = \frac{1}{|\mathcal{C}_k|} \sum_{x \in \mathcal{C}_k} (x_1, .., x_p)$
4: Choisir $x_{repr_k} = \min_{x \in P} d(x - c_k)$
5: **fin pour**
6: **fin fonction**

Heuristique 2 : L'utilisateur pourra valoriser son expertise en traduisant ses critères sous forme de fonctions (coût monétaire, composition chimique, masse, impact environmental etc..) avec lesquels il souhaitera trier les points de chaque classe. Ces critères sont utilisés pour créer une fonction de score qui permettra de trier ces points, ou effectuer une optimisation mono-objectif

simple pour minimiser ou maximiser cette fonction.

Il faut noter qu'utiliser ces critères pour sélectionner des solutions est très différent de les introduire en tant que fonctions objectif. L'ajout de fonctions objectif augmente grandement le temps d'exécution et complexité de calcul d'un solveur d'optimisation [Audet et al.2010]. La résolution d'un problème d'optimisation en boîte noire à deux objectifs peut prendre des jours tel que le problème du chapitre suivant, ajouter un troisème objectif pourrait prendre le solveur des mois pour résoudre le problème. Il est donc plus judicieux de ne garder que deux fonctions objectifs, et introduire tous les autres critères dans la fonction score de la deuxième heuristique plutôt que de ne pas les utiliser du tout.

CHAPITRE 4 APPLICATION PRATIQUE DE LA MÉTHODOLOGIE

Ce chapitre aborde une application pratique de la méthodologie à un problème réel. Il est structuré comme suit : Dans un premier temps, le problème sera présenté ainsi que son contexte. Ensuite, la méthodologie définie dans le chapitre précédent lui sera appliquée de manière commentée et enfin la dernière partie concerne l'interprétation et synthèse des résultats.

4.1 Présentation du problème

La génération d'énergie se tourne de plus en plus vers le renouvelable, l'énergie solaire est l'une des technologies majeures dans le domaine des énergies vertes. Il existe plusieurs méthodes de génération d'énergie électrique à partir d'énergie solaire, celle sur laquelle on se focalisera dans cette application est l'énergie solaire thermique. C'est une technologie qui repose sur le fait de chauffer un fluide caloporteur en concentrant le rayonnement solaire sur ce dernier, pour ensuite libérer cette énergie soit pour un but de chauffage d'eau ou d'infrastructures, ou pour génération d'énergie électrique en utilisant cette chaleur pour faire évaporer de l'eau qui sera passée à travers des turbines dont la rotation génère de l'énergie électrique [Gheribi et al.2014].

Le principal inconvénient des centrales solaires est qu'elles ne peuvent pas produire d'énergie électrique la nuit ainsi que dans des conditions météorologiques difficiles. Pour résoudre ce problème, des solutions de sels fondus sont utilisées pour stocker le surplus d'énergie électrique dans des réservoirs sous forme d'énergie thermique. Cette énergie est ensuite récupérée quand la centrale ne peut produire d'énergie électrique depuis l'énergie solaire directement, comme quand il fait nuit.

L'application de ce chapitre provient d'une problématique réelle en génie chi-

mique. On souhaite synthétiser un mélange de sels fondus à partir de huit sels différents : Le Chlorure de Lithium ($LiCl$), le Chlorure de Sodium ($NaCl$), Le Chlorure de Potassium (KCl), Le Chlorure de Calcium ($CaCl_2$), Le Chlorure de Magnésium ($MgCl_2$), Le Chlorure de Manganèse ($MnCl_2$), Le Chlorure de Nickel ($NiCl_2$) et le Chlorure de Cobalt ($CoCl_2$). Le but est d'obtenir un constituant de température de fusion minimale pour lui permettre de rapidement atteindre l'état liquide ainsi qu'une chaleur latente maximale, lui permettant de retenir une plus grande quantité d'énergie thermique. Il est en effet important d'avoir un fluide caloporteur qui soit capable d'atteindre l'état liquide rapidement et capable de stocker et libérer de grandes quantités d'énergie au besoin [Gheribi et al.2014].

Des efforts de recherche en optimisation dans ce secteur ont déjà lieu au sein du GERAD comme en atteste le travail de Lemyre Garneau [LemyreGarneau2015] et les travaux de Audet, LeDigabel et Gheribi [Gheribi et al.2011, Gheribi et al.2013].

Ce problème peut être reformulé sous forme de problème d'optimisation comme suit :

$$\begin{aligned} \min_{x\in\Omega} \quad & F(x) = (f_1(x), f_2(x)) \\ \text{avec} \quad & F : \mathcal{R}^8 \longrightarrow \mathcal{R}^2 \; et \; x \in \Omega \subseteq \mathcal{R}^8 \\ \text{où} \quad & f_1 \text{ est la température de fusion} \\ \text{et} \quad & f_2 \text{ est l'opposé de la chaleur latente.} \end{aligned}$$

Les seules contraintes auquelles est soumis ce problème sont les contraintes de bornes des contentrations des sels.

4.2 Extraction des données

On commence par lire les données fournies pour ce problème en les normalisant. On choisis dans ce cas d'effectuer une normalisation car les données ne contiennent pas de valeur extrêmes, et que les fonctions objectifs sont

sur des échelles très différentes. Une partie de ces données a été générée par un échantillonnage avec hypercube latin [McKay et al.1979], tandis qu'une autre provient de la cache d'un déroulement de l'algorithme BIMADS [Audet et al.2008]. Les valeurs des fonctions objectif ont été obtenues par simulation à travers le logiciel de simulation thermodynamique FactSage [Bale et al.2002]. Chaque appel pour générer un point prenait entre 30 secondes à 12 minutes. Les points non réalisables ont été filtrés, et tous les points qui restent sont réalisables. Les données sont ensuite normalisées. La figure 4.1 montre les cinq premières des 34554 lignes que comporte la base de données avant normalisation.

	LiCl	NaCl	KCl	CaCl_2	MgCl_2	MnCl_2	NiCl_2	CoCl_2	f1	f2
0	33.0	54.0	66.0	49.0	6.0	28.0	9.0	66.0	413.2337	1.111307
1	90.0	85.0	81.0	54.0	93.0	66.0	100.0	16.0	564.5982	0.880445
2	15.0	39.0	51.0	21.0	54.0	61.0	91.0	18.0	658.5661	0.808576
3	66.0	24.0	83.0	35.0	70.0	30.0	19.0	91.0	472.5585	1.086090
4	23.0	20.0	54.0	77.0	65.0	77.0	47.0	57.0	615.7641	0.845349

Figure 4.1 Aperçu des variables du problème

En appliquant la normalisation comme expliqué au chapitre précédent, on obtient la base de données normalisée de la figure 4.2. Les variables de x_1 à x_8 représentent respectivement les concentrations des huis sels : le Chlorure de Lithium ($LiCl$), le Chlorure de Sodium ($NaCl$), Le Chlorure de Potassium (KCl), Le Chlorure de Calcium ($CaCl_2$), Le Chlorure de Magnésium ($MgCl_2$), Le Chlorure de Manganèse ($MnCl_2$), Le Chlorure de Nickel ($NiCl_2$) et le Chlorure de Cobalt ($CoCl_2$).

	x1	x2	x3	x4	x5	x6	x7	x8	f1	f2
0	0.106109	0.173633	0.212219	0.157556	0.019293	0.090032	0.028939	0.212219	0.396616	0.806089
1	0.153846	0.145299	0.138462	0.092308	0.158974	0.112821	0.170940	0.027350	0.541894	0.638632
2	0.042857	0.111429	0.145714	0.060000	0.154286	0.174286	0.260000	0.051429	0.632083	0.586502
3	0.157895	0.057416	0.198565	0.083732	0.167464	0.071770	0.045455	0.217703	0.453555	0.787798
4	0.054762	0.047619	0.128571	0.183333	0.154762	0.183333	0.111905	0.135714	0.591002	0.613176

Figure 4.2 Aperçu des variables du problème après normalisation

Ensuite, on calcule l'approximation des points Pareto. La figure 4.3 montre la totalité des points dans l'espace (f_1, f_2) à gauche ainsi que les points non dominés à droite. On voit dans cette figure que l'espace (f_1, f_2) est composé de quatres zones majeures ; une grosse partie dense qui occupe la majeure partie de l'espace, ansi que trois régions plus petites rectilignes en bas à droite sur la figure. Pour les points non dominés, ils sont au nombre de 107, un nombre assez petit pour pouvoir effectuer une segmentation en huit dimensions. On peut apercevoir à première vue cinq zones distinctes, trois petites zones déconnectées vers le maximum de f_2, maximum de f_1 ainsi qu'au milieu, mais aussi une large partie vers le coude de la figure que l'on peut découper en deux section là où est présente une discontinuité.

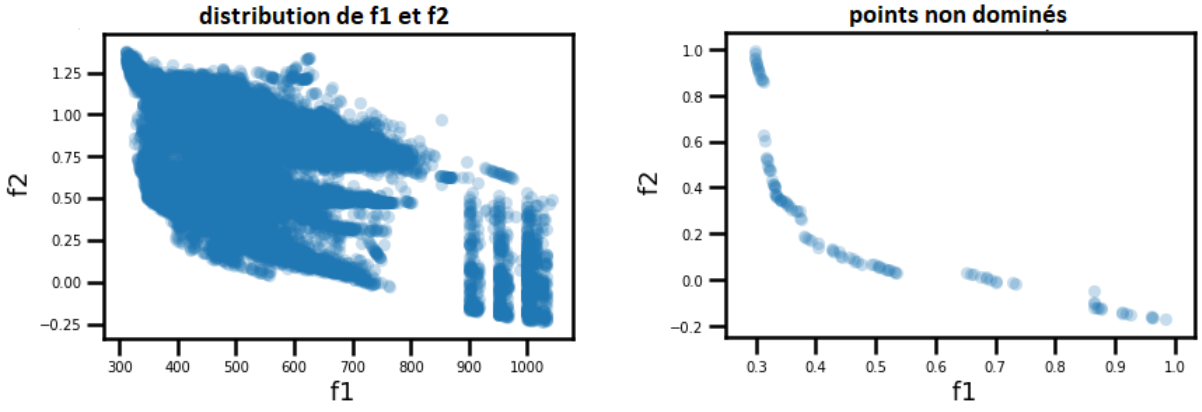

Figure 4.3 Distribution totale de f_1 et f_2 (gauche) et les points non dominés (droite)

La prochaine étape est d'augmenter les points à analyser par l'algorithme 7.

On peut aussi ajuster le rayon pour manipuler à quelle distance de front Pareto on est prêt à explorer des solutions intéréssantes plus loin.

Pour ce problème, on opte pour rayon d'approximation la valeur 0.05. On prend donc tous les points dans un rayon de 0.05 autour de chaque point non dominé dans l'espace objectif (f_1, f_2), et on obtient environ 6000 points, ce qui constitue 17% des données. La figure 4.4 montre les points que l'on considère. On peut voir qu'ils sont disposés en îlots avec six régions distinctes.

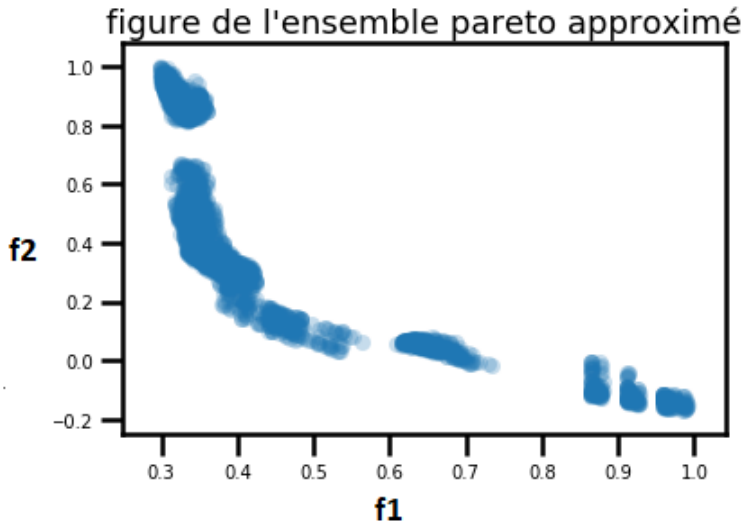

Figure 4.4 Points approximés autour des points non dominés

4.3 Segmentation

La prochaine étape est celle d'appliquer un algorithme de segmentation. On ne connait pas la structure des données a priori, on utilise donc les quatre algorithmes de segmentation en optimisant leurs paramètres. Pour ce faire, on doit dans un premier temps définir le meilleur nombre de classes pour les algorithmes qui le requièrent en paramètre, à savoir Kmeans et l'algorithme

spectral. On applique la méthode de l'analyse de silhouette pour chacun et on compare le score de silhouette de différents nombres de classes. On obtient les résultats de la figure 4.5 pour Kmeans.

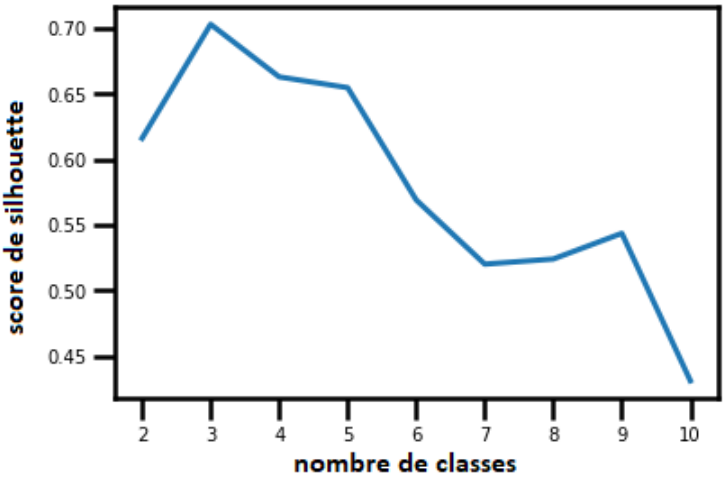

Figure 4.5 Scores de silhouette pour chaque nombre de classes

On voit que la courbe n'est pas régulière et qu'il existe deux pics aux nombres de 3 et 9. Le score de silhouette au nombre de trois classes étant le plus grand, on le retient donc pour les algorithmes k-moyennes. L'algorithme spectral fournis un résultat similaire et un meilleur nombre de classe de trois.

On applique à présent les quatre algorithmes de segmentation et on compare les classes trouvées par chacun. On peut voir sur la figure 4.6 que les quatre algorithmes trouvent des résultats similaires. Dans la figure 4.6, les hyperparamètres des algorithmes Meanshift et HDBscan ont en effet été optimisés pour ce problème avec une méthode de recherche par grille.

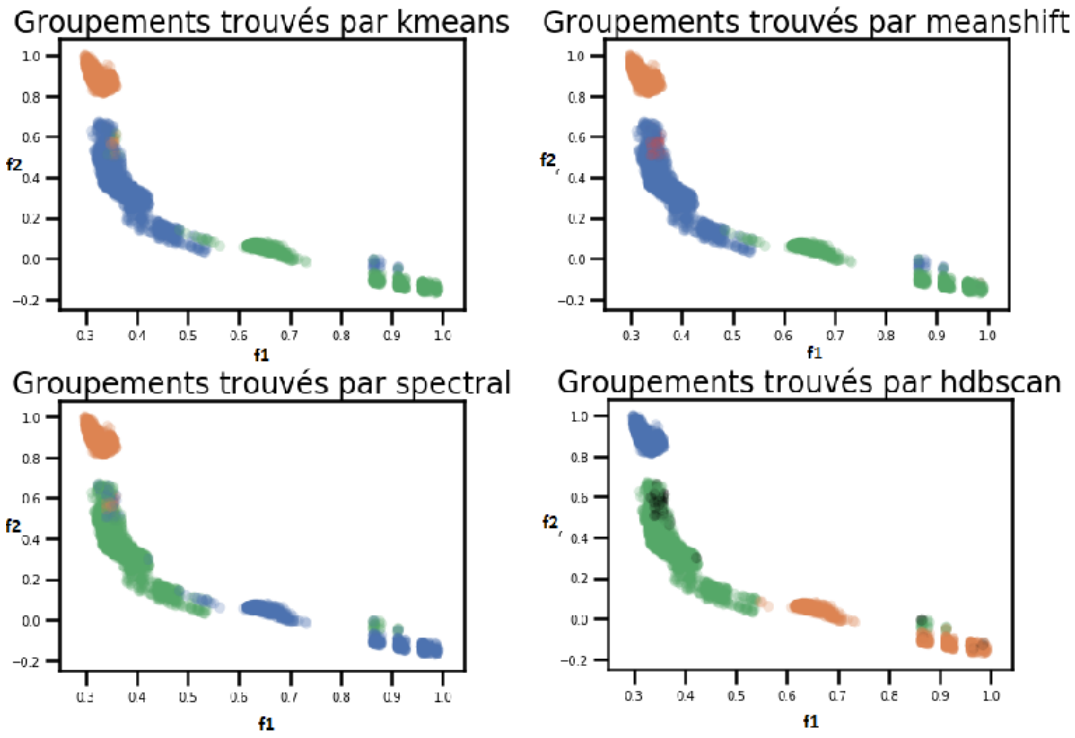

Figure 4.6 Classes trouvées par chaque algorithme

On remarque des effets intéressants sur la figure 4.6. Chaque classe correspondant à une couleur pour chacun des algorithmes, on voit que globalement les classes sont distinctes, mais il existe bien une petite partie de chaque classe très proche d'autres classes. Cependant HDBscan assigne ces petites zones comme étant du bruit. Ceci nous indique que ces régions sont des régions peu denses se trouvant aux extrémités des classes dans l'espace huit-dimensionnel des variables, et ne sont donc pas intéressants à considérer.

Avant de passer aux prochaines étapes de la méthodologie, on effectue certaines expériences pour découvrir, pour ce problème, si une segmentation sur les variable donne des résultats différents qu'une segmentation sur les fonctions objectif. Puis on explore si inclure des points plus loin de la région optimale changerait les résultats de la segmentation. Pour les expériences qui suivent, on utilise l'algorithme Kmeans pour les expériences.

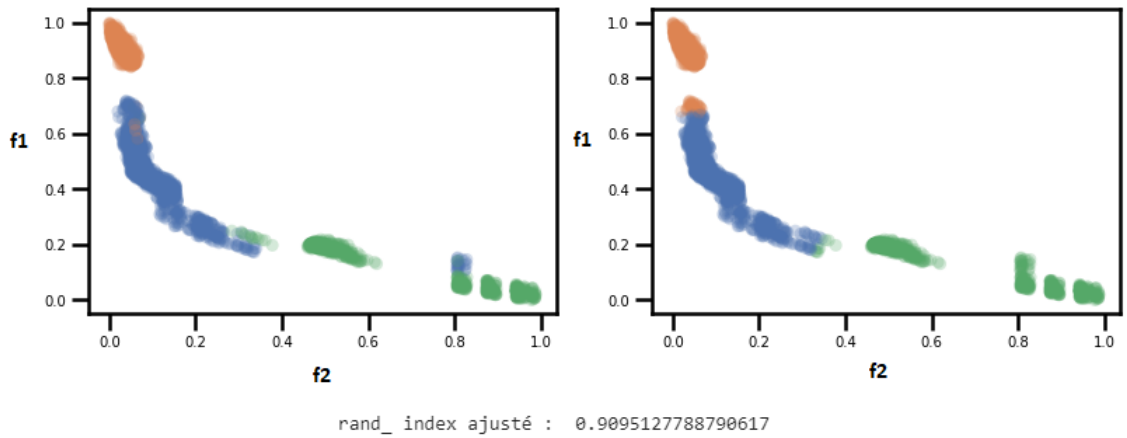

Figure 4.7 Segmentation sur l'espace des variables (gauche) et sur l'espace objectif (droite)

La figure 4.7 montre les résultats d'une segmentation sur l'espace des variable utilisée dans cette méthode à gauche, ainsi qu'une segmentation effectuée sur l'espace des fonctions objectif à droite. On utilise la mesure du Rand-Index ajusté pour calculer la dissimilarité entre les deux segmentations [Rand1971]. Plus cette valeur est proche de 1 plus les segmentations sont similaires, au contraire plus la valeur est proche de 0 plus grande est la proportion de points qui a été segmentée différamment. On peut voir que pour cette expérience, on obtient un Rand-Index ajusté de $0,9$. C'est-à-dire, environ 10% des solutions ont été classifiées différemment, ce qui peut être conséquent. Ce résultat montre que segmenter sur les variables donne des résultats différents que de segmenter sur les fonctions objectif. Or pour l'objectif de cette méthodologie de regrouper les variables selon leur similarité dans l'espace des variables, la première segmentation est plus pertinente.

Une deuxième expérience étudie l'inclusion de solutions plus éloignées du front Pareto et son influence sur la segmentation. En effet, il serait contre-intuitif si une telle influence existe car cela indiquerais qu'inclure ces points biaise l'algorithme de segmentation, et que donc les résultats de la segmentation ne seront pas valides. La figure 4.8 montre les résultats d'une segmentation

faite uniquement sur les points Pareto à gauche ainsi que les résultats trouvés précedemment pour les points relaxés à droite. Pour mesurer une éventuelle différence de segmentation, le Rand-Index ajusté sur les points partagés par les deux ensembles (c'est à dire les points Pareto) est calculé. On obtient une valeur de 1 qui indique que les deux segmentation assignent les points exactement au mêmes classes. On peut donc dire avec confiance qu'explorer des solutions plus ou moins loins des solutions optimales n'influence pas les résultats de la segmentation.

Cependant, la segmentation sur l'ensemble des points relaxés est plus riche. Elle révèle des points proches dans l'ensemble des variables mais éloignés dans l'espace des foncions objectifs.

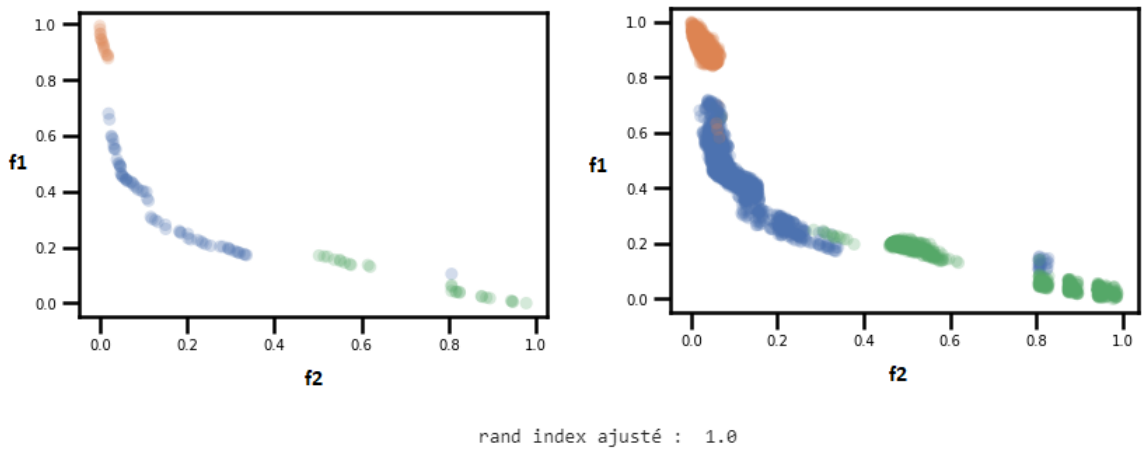

Figure 4.8 Segmentation sur les points Pareto (gauche) et les points relaxés (droite)

La figure 4.9 permet de visualiser les statistiques descriptives pour chaque classe. Ceci permet à l'expert de connaître les propriétés de chaque classe ainsi que le sel prédominant dans celle-ci. Par exemple dans la première classe, le sel prédominant est celui correspondant à x_3, le Chlorure de Potassium KCl. Les sels prédominant dans la deuxième classe sont x_8 (le Chlorure de Cobalt $CoCl_2$) principalement, ainsi que x_2 (le Chlorure de Sodium $NaCl$) et x_3 (le

Chlorure de Potassium KCl) dans une moindre mesure. Pour la troisième classe, les sels prédominants sont x_1 et x_3 qui correspondent au Chlorure de Lithium $LiCl$ et le Chlorure de Potassium KCl. Ces statistiques portent un très grand grand intérêt pour les experts, car elles leur permettent de rapidement connaître la constitution de chaque classe, ses bornes, sa densité ainsi que ses caractéristiques.

La distribution des Xi pour la classe 1 est :

	x1	x2	x3	x4	x5	x6	x7	x8
count	662.000000	662.000000	662.000000	662.000000	662.000000	662.000000	662.000000	662.000000
mean	0.102261	0.003901	0.792250	0.014973	0.011571	0.031722	0.029404	0.013917
std	0.035357	0.009242	0.068488	0.028516	0.015871	0.033431	0.029988	0.037031
min	0.017544	0.000000	0.503448	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.081538	0.000000	0.777778	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.097561	0.000000	0.806452	0.000000	0.000000	0.024096	0.023529	0.000000
75%	0.114754	0.000000	0.830932	0.016598	0.016227	0.046110	0.046875	0.015385
max	0.268293	0.081633	0.925000	0.280899	0.087719	0.265957	0.202703	0.347518

La distribution des Xi pour la classe 2 est :

	x1	x2	x3	x4	x5	x6	x7	x8
count	899.000000	899.000000	899.000000	899.000000	899.000000	899.000000	899.000000	899.000000
mean	0.036920	0.190741	0.189605	0.109073	0.089589	0.161444	0.001913	0.220715
std	0.020052	0.029444	0.049380	0.053933	0.028608	0.042488	0.002709	0.057699
min	0.000000	0.030928	0.086817	0.000000	0.000000	0.016713	0.000000	0.020305
25%	0.024096	0.189024	0.168496	0.056131	0.079832	0.122899	0.000000	0.199203
50%	0.036403	0.196464	0.177515	0.103030	0.088710	0.174419	0.000000	0.206612
75%	0.043668	0.202492	0.202052	0.152720	0.105051	0.192308	0.002941	0.275135
max	0.170103	0.260450	0.500000	0.283828	0.219178	0.365546	0.024064	0.406504

La distribution des Xi pour la classe 3 est :

	x1	x2	x3	x4	x5	x6	x7	x8
count	1181.00000	1181.00000	1181.00000	1181.00000	1181.00000	1181.00000	1181.00000	1181.00000
mean	0.414739	0.048527	0.479904	0.006008	0.011523	0.028545	0.004088	0.006666
std	0.111601	0.039562	0.068510	0.014327	0.018476	0.038971	0.007400	0.015046
min	0.213592	0.000000	0.180451	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.344156	0.005348	0.458333	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.377483	0.053892	0.492611	0.000000	0.005714	0.014815	0.000000	0.000000
75%	0.492857	0.081633	0.520710	0.006494	0.016043	0.033898	0.005988	0.006757
max	0.800000	0.176471	0.644444	0.135338	0.120690	0.191919	0.045977	0.205021

Figure 4.9 Statistiques des classes

On remarque que le Chlorure de Potassium est présent dans les trois classes. L'expert peut à l'aide des figures des groupements 4.6 ainsi que du tableau des

statistiques 4.9 choisir une solution dans une région d'intérêt avec l'une des deux heuristiques présentées au chapitre précédent. Malgré qu'une normalisation a été utilisée pour ce problème, on n'aperçoit pas les valeurs extrémales de 0 et 1 en tant que minimum et maximum pour chaque variable dans le tableau. Ceci est dû que le tableau ne regoupe que les points gardés suite à la relaxation du front Pareto. Or la normalisation a été faite immédiatement après la lecture des données. Les points correspondant à ces valeurs extrémales n'ont pas été sélectionnés lors de la relaxation du front Pareto car en sont trop éloignés.

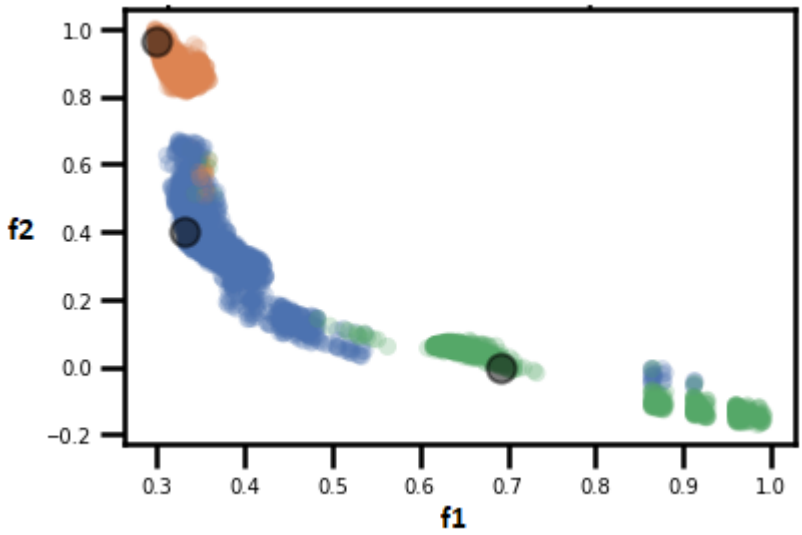

Figure 4.10 Représentant par classe 1ère heuristique

La figure 4.10 montre un exemple d'utilisation de la première heuristique discutée au chapitre précédent. On sélectionne un représentant dans la région la plus dense de chaque classe. L'intérêt de cette heuristique est d'effectuer une sélection rapide d'un représentant quand on n'a a pas forcément d'autres critères à respecter.

Cependant, il est préférable d'utiliser la deuxième heuristique quand le choix de solution par l'expert est motivé par des critères tels que le prix, impact écologique ou autres qui ne sont pas intégrées dans les fonctions objectif. En

effet, bien que ces critères soient importants pour l'ingénieur qui résout le problème d'optimisation, il ne peut pas se permettre de les inclure dans les fonctions objectif sous peine de trop alourdir le problème.

Pour illustrer cette heuristique, on s'intéresse aux solutions les moins chères possibles. On considère les valeurs de prix des sels du tableau 4.1 en $/kg.

Tableau 4.1 Tableau des prix

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
2745.00	32.25	105.60	51.90	112.95	157.05	861.00	2692.50

On définit le coût d'une solution comme suit :

$$\text{pour } x \in \Omega \qquad C(x) = \frac{\sum p_i * x_i}{\sum p_i}$$

avec p_i : le prix du sel i.

On commence par donner un score ou coût à chaque solution d'après les critères retenus, (ici uniquement le prix). Ensuite on trie les points de chaque classe. L'expert peut ensuite explorer les points ayant le meilleur score de chaque classe pour en choisir un. On peux voir dans le tableau de la figure 4.11 un aperçu des solutions les moins coûteuses de chaque classe.

	x1	x2	x3	x4	x5	x6	x7	x8	f1	f2	Coût	labels
550	0.152174	0.054348	0.532609	0.141304	0.000000	0.108696	0.010870	0.000000	0.334740	0.634193	0.075386	0
519	0.159091	0.045455	0.534091	0.125000	0.000000	0.125000	0.011364	0.000000	0.328234	0.641776	0.078493	0
706	0.164835	0.054945	0.538462	0.120879	0.000000	0.120879	0.000000	0.000000	0.348575	0.569744	0.079364	0
511	0.170455	0.045455	0.545455	0.125000	0.000000	0.113636	0.000000	0.000000	0.350997	0.560320	0.081574	0
330	0.170455	0.045455	0.534091	0.125000	0.000000	0.125000	0.000000	0.000000	0.327724	0.648465	0.081661	0
9	0.006042	0.178248	0.163142	0.235650	0.151057	0.229607	0.012085	0.024169	0.358238	0.861207	0.026693	1
2733	0.022843	0.157360	0.182741	0.230964	0.177665	0.197970	0.010152	0.020305	0.352394	0.872306	0.031611	1
319	0.000000	0.218341	0.218341	0.218341	0.072052	0.218341	0.000000	0.054585	0.349576	0.890067	0.034155	1
1665	0.000000	0.218818	0.218818	0.218818	0.070022	0.218818	0.000000	0.054705	0.349716	0.906245	0.034193	1
1141	0.000000	0.219780	0.219780	0.219780	0.065934	0.219780	0.000000	0.054945	0.349995	0.899537	0.034270	1
2445	0.037500	0.000000	0.925000	0.000000	0.012500	0.025000	0.000000	0.000000	0.706872	0.016649	0.030475	2
968	0.017544	0.000000	0.842105	0.000000	0.052632	0.000000	0.087719	0.000000	0.686435	0.061866	0.032339	2
409	0.033898	0.000000	0.864407	0.016949	0.033898	0.016949	0.033898	0.000000	0.918039	-0.107324	0.032684	2
2455	0.037037	0.000000	0.876543	0.000000	0.049383	0.000000	0.037037	0.000000	0.691357	0.042695	0.034283	2
2702	0.048387	0.000000	0.887097	0.000000	0.000000	0.064516	0.000000	0.000000	0.690095	0.023490	0.035014	2

Figure 4.11 Tableau des solutions triées par coût

La figure 4.12 illustre les points avec le meilleur score. On voit que deux des points sont très différents que ceux trouvés par la première heuristique. D'où la suggestion d'utiliser cette deuxième au possible car elle permet d'intégrer autant de critères que souhaité. Il est aussi important de noter qu'il est possible de contrôler le nombre de points explorés en modifiant le rayon de relaxation du front Pareto comme énoncé précedemment.

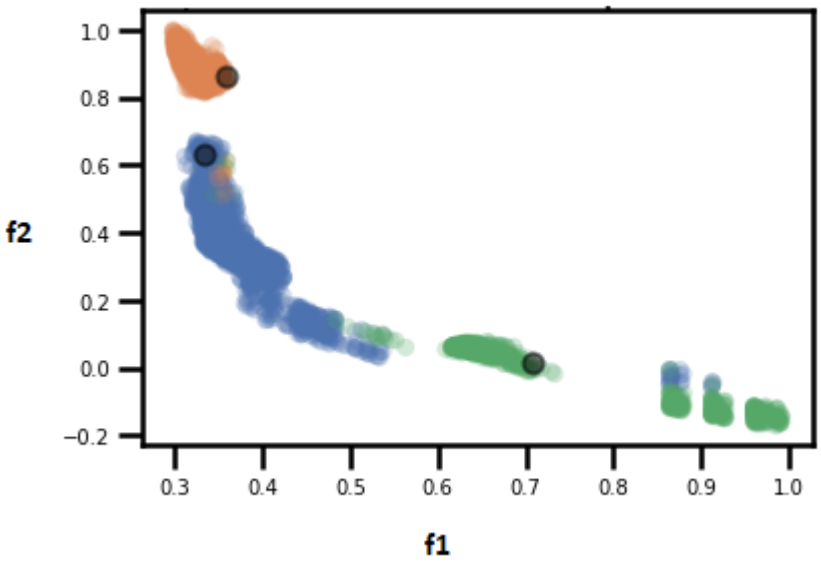

Figure 4.12 Représentant par classe 2ème heuristique

CHAPITRE 5 CONCLUSION ET RECOMMANDATIONS

Dans ce travail, nous avons proposé une méthodologie permettant d'identifier des solutions intéressantes sur un front Pareto en segmentant les solutions non dominées ainsi qu'un ensemble de solutions proches de l'optimalité dans l'espace des variables. Ceci nous permet d'identifier des classes de caractéristiques similaires qui peuvent guider vers un meilleur choix d'une solution à adopter. La première étape consiste à générer et préparer les données. On choisit ensuite un algorithme de segmentation. Ce choix d'algorithme dépend de la nature du problème, et chacun des quatre algorithmes utilisés dans ce travail traite d'un problème de nature différente. On dresse par la suite les figures et tableaux décrivant les classes trouvées. Et on procède enfin au choix d'une solution au sein d'une classe d'intérêt en incorporant éventuellement de nouveaux critères pour trier les solutions au sein de cette classe.

Une application de cette méthodologie à un problème d'industrie réel a été effectuée au chapitre 4. Cette application a permis de vérifier la validité de cette approche. Des expériences ont également été réalisées pour valider le choix d'une segmentation sur les variables ainsi que le fait de relaxer le front Pareto n'influence pas négativement la segmentation.

Une telle méthodologie est utile pour les problèmes à systèmes complexes tels que la synthétisation et caractérisation de matériaux. En effet, de manière classique, l'impact des constituants dans de tels problèmes sont comparés deux à deux. Des méthodes telles que CALPHAD [Saunders and Miodownik1998] ont été développées pour permettre d'en comparer un plus grand nombre. Cependant, il reste très difficile d'effectuer une optimisation car les appels aux simulations que l'on peut considérer comme des boîtes noires peuvent prendre de nombreuses minutes. Il est donc difficile d'effectuer une optimisation sur les variables de tels problèmes. Les algorithmes d'optimisation spécialisés dans un contexte en boîte noires tels que MADS et BIMADS ont été développés

pour résoudre cette problèmatique. Toutefois, les solutions Pareto optimales trouvées par ces algorithmes ne sont pas toujours les plus intéressantes à considérer pour l'expert. Car d'après des correspondances avec un expert, il est souvent le cas que des solutions plus économiquement ou écologiquement plus viables se trouvent proches de ces solutions optimales et il est difficile de les repérer. C'est sous cette lumière que cette méthodologie entre en jeu. En effet, comme dans le problème traité dans ce document, on a été capable de non seulement séparer les solutions optimales en classes de compositions différentes, mais également d'identifier des solutions adjacentes sur le plan objectif appartenant à ces mêmes classes. Il est ainsi possible d'avoir une description plus large des solutions pour choisir par exemple une solution à une classe d'intérêt, mais également chercher au sein de cette même classe un point qui a une concentration minimale d'un certain sel qui réagit de manière corrosive à tel acier.

La méthode présentée dans ce travail est principalement axée dans un cadre biobjectif. L'analyse et interprétation des classes trouvées peut s'avérer difficile dans un cadre multiobjectif avec un plus grand nombre de fonctions objectif. Une idée permettant d'étendre cette méthode sur un plus grand nombre de fonctions objectifs serait d'intégrer des techniques de réduction de dimensionalité pour permettre de garder la capacité de visualisation en deux dimensions malgré l'utilisation de nombreux objectifs. On perdra cependant le pouvoir d'interprétation à partir des fonctions objectifs car les axes en dimensions réduites sont formées à partir de combinaisons des dimensions de base. Un autre défi de l'utilisation de techniques de réduction de dimensionalité est que les distances entre les points en dimensions réduites ne sont pas forcément corrélées à leur distance dans l'espace multidimensionnel d'origine. Il existe cependant un algorithme qui résout ce problème permettant d'obtenir des visualisations en dimensions réduites en spécifiant en paramètre de conserver soit une structure locale (respect des distances entre les points dans l'espace d'origine) ou bien une structure globale (respect de la forme générale

des points et espacement entre les groupement dans l'espace d'origine).

RÉFÉRENCES

[Abramson et al.2009] Abramson, M., Audet, C., Dennis, Jr., J., and Le Digabel, S. (2009). OrthoMADS : A Deterministic MADS Instance with Orthogonal Directions. *SIAM Journal on Optimization*, 20(2) :948–966.

[Agrawal, Shubham and Panigrahi, Bijaya Ketan and Tiwari2008] Agrawal, Shubham and Panigrahi, Bijaya Ketan and Tiwari (2008). Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch. *IEEE Transactions on Evolutionary Computation*, 12(5) :529–541.

[Alarie et al.2013] Alarie, S., Audet, C., Garnier, V., Le Digabel, S., and Leclaire, L.-A. (2013). Snow water equivalent estimation using blackbox optimization. *Pacific Journal of Optimization*, 9(1) :1–21.

[Arthur2006] Arthur, David et Vassilvitskii, S. (2006). k-means++ : The advantages of careful seeding. Technical report, Stanford.

[Audet and Dennis, Jr.2006] Audet, C. and Dennis, Jr., J. (2006). Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, 17(1) :188–217.

[Audet and Hare2017] Audet, C. and Hare, W. (2017). *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, Cham, Switzerland.

[Audet et al.2014] Audet, C., Ianni, A., Le Digabel, S., and Tribes, C. (2014). Reducing the Number of Function Evaluations in Mesh Adaptive Direct Search Algorithms. *SIAM Journal on Optimization*, 24(2) :621–642.

[Audet et al.2008] Audet, C., Savard, G., and Zghal, W. (2008). Multiobjective Optimization Through a Series of Single-Objective Formulations. *SIAM Journal on Optimization*, 19(1) :188–210.

[Audet et al.2010] Audet, C., Savard, G., and Zghal, W. (2010). A mesh adaptive direct search algorithm for multiobjective optimization. *European Journal of Operational Research*, 204(3) :545–556.

[Ayodele2010] Ayodele, T. O. (2010). Types of machine learning algorithms. *New advances in machine learning*, pages 19–48.

[Bale et al.2002] Bale, C. W., Chartrand, P., Degterov, S., Eriksson, G., Hack, K., Ben Mahfoud, R., Melançon, J., Pelton, A., and Petersen, S. (2002). FactSage thermochemical software and databases. *Calphad*, 26(2) :189–228.

[Bazaraa et al.2013] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M. (2013). *Nonlinear programming : theory and algorithms*. John Wiley & Sons.

[Belton and Stewart2002] Belton, V. and Stewart, T. (2002). *Multiple criteria decision analysis : an integrated approach*. Springer Science & Business Media.

[Bishop2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[Boyd et al.2004] Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[Campello2013] Campello, Ricardo J. G. B. et Moulavi, D. e. S. J. e. J. e. T. V. S. e. C. L. e. M. H. e. X. G. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In *Advances in Knowledge Discovery and Data Mining*, pages 160–172. Springer Berlin Heidelberg.

[Carneiro and Vasconcelos2005] Carneiro, G. and Vasconcelos, N. (2005). Formulating semantic image annotation as a supervised learning problem. 2 :163–168.

[Cheeger1969] Cheeger, J. (1969). A lower bound for the smallest eigenvalue of the Laplacian. In *Proceedings of the Princeton conference in honor of Professor S. Bochner*, pages 195–199.

[Cherkassky and Mulier2007] Cherkassky, V. and Mulier, F. M. (2007). *Learning from data : concepts, theory, and methods*. John Wiley & Sons.

[Chung and Graham1997] Chung, F. R. and Graham, F. C. (1997). graph theory. (92).

[Clarke1983] Clarke, F. (1983). *Optimization and Nonsmooth Analysis*. John Wiley & Sons, New York. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

[Conn et al.2009] Conn, A., Scheinberg, K., and Vicente, L. (2009). *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization. SIAM, Philadelphia.

[Ester et al.1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. e. a. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

[Fermi and Metropolis1952] Fermi, E. and Metropolis, N. (1952). Numerical solution of a minimum problem. Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos, USA.

[Fiedler1973] Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2) :298–305.

[Fowler et al.2008] Fowler, K., Reese, J., Kees, C., Dennis Jr., J., Kelley, C., Miller, C., Audet, C., Booker, A., Couture, G., Darwin, R., Farthing, M., Finkel, D., Gablonsky, J., Gray, G., and Kolda, T. (2008). Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5) :743–757.

[Gheribi et al.2013] Gheribi, A. E., Le Digabel, S., Audet, C., and Chartrand, P. (2013). Identifying optimal conditions for magnesium based alloy design using the Mesh Adaptive Direct Search algorithm. *Thermochimica Acta*, 559 :107–110.

[Gheribi et al.2011] Gheribi, A. E., Robelin, C., Le Digabel, S., Audet, C., and Pelton, A. D. (2011). Calculating all local minima on liquidus surfaces using the factsage software and databases and the mesh adaptive direct search algorithm. *The Journal of Chemical Thermodynamics*, 43(9) :1323–1330.

[Gheribi et al.2014] Gheribi, A. E., Torres, J. A., and Chartrand, P. (2014). Recommended values for the thermal conductivity of molten salts between the melting and boiling points. *Solar energy materials and solar cells*, 126 :11–25.

[Hinton et al.1999] Hinton, G. E., Sejnowski, T. J., Poggio, T. A., et al. (1999). *Unsupervised learning : foundations of neural computation*. MIT press.

[Hsieh2009] Hsieh, W. W. (2009). *Machine learning methods in the environmental sciences : Neural networks and kernels*. Cambridge university press.

[K. Fukunaga1975] K. Fukunaga, L. H. (1975). The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition . pages 32–40.

[Keeney et al.1993] Keeney, R. L., Raiffa, H., et al. (1993). *Decisions with multiple objectives : preferences and value trade-offs*. Cambridge University Press.

[Kenisarin2014] Kenisarin, M. M. (2014). Thermophysical properties of some organic phase change materials for latent heat storage. A review. *Solar Energy*, 107 :553 – 575.

[Kipouros et al.2008] Kipouros, T., Jaeggi, D. M., Dawes, W. N., Parks, G. T., Savill, A. M., and Clarkson, P. J. (2008). Biobjective design optimization for axial compressors using tabu search. *AIAA journal*, 46(3) :701–711.

[Langley and Simon1995] Langley, P. and Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, 38(11) :54–64.

[Larson et al.2019] Larson, J., Menickelly, M., and Wild, S. (2019). Derivative-free optimization methods. *Acta Numerica*, 28 :287–404.

[LemyreGarneau2015] LemyreGarneau, M. (2015). Modelling Of A Solar Thermal Power Plant For benchmarking Blackbox Optimization Solvers. Master's thesis, Polytechnique Montréal.

[MacQueen et al.1967] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Marsh et al.2016] Marsh, K., IJzerman, M., Thokala, P., Baltussen, R., Boysen, M., Kaló, Z., Lönngren, T., Mussen, F., Peacock, S., Watkins, J., et al. (2016). Multiple criteria decision analysis for health care decision making—emerging good practices : report 2 of the ISPOR MCDA Emerging Good Practices Task Force. *Value in health*, 19(2) :125–137.

[Masud1979] Masud, C.-L. H. A. S. M. (1979). *Multiple objective decision making, methods and applications : a state-of-the-art survey*. Berlin ; New York : Springer-Verlag.

[Maulik et al.2009] Maulik, U., Mukhopadhyay, A., and Bandyopadhyay, S. (2009). Combining pareto-optimal clusters using supervised learning for identifying co-expressed genes. *BMC bioinformatics*, 10(1) :27.

[McKay et al.1979] McKay, M., Beckman, R., and Conover, W. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2) :239–245.

[Miettinen1999] Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management Science. Springer US.

[Miettinen2012] Miettinen, K. (2012). *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media.

[Mitchell et al.1997] Mitchell, T. M. et al. (1997). Machine learning. 1997. *Burr Ridge, IL : McGraw Hill*, 45(37) :870–877.

[Morse1980] Morse, J. N. (1980). Reducing the size of the nondominated set : Pruning by clustering. *Engineering Optimization*, 7(1-2) :55–66.

[Ng et al.] Ng, A. Y., Jordan, M. I., and Weiss, Y.

[Ng and Weiss2002] Ng, Andrew Y., M. I. J. and Weiss, Y. (2002). On Spectral Clustering : Analysis and an algorithm . pages 849–856.

[Pareto1896] Pareto, V. (1896). *Manual of Political Economy.* F. Rouge, Lausanne.

[Rajagopal et al.2011] Rajagopal, D. et al. (2011). Customer data clustering using data mining technique. *arXiv preprint arXiv :1112.2663.*

[Rand1971] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336) :846–850.

[Roman et al.2003] Roman, N., Wang, D., and Brown, G. J. (2003). Speech segregation based on sound localization. *The Journal of the Acoustical Society of America*, 114(4) :2236–2252.

[Rosemann and GERO1985] Rosemann, M. A. and GERO, J. S. (1985). Reducing the Pareto optimal set in multicriteria optimization(with applications to Pareto optimal dynamic programming). *Engineering Optimization*, 8(3) :189–206.

[Rosenman and Gero1985] Rosenman, M. and Gero, J. (1985). Reducing the Pareto optimal set in multicriteria optimization (with applications to Pareto optimal dynamic programming). *Engineering Optimization*, 8(3) :189–206.

[Rousseeuw1987] Rousseeuw, P. J. (1987). Silhouettes : a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20 :53–65.

[Russell and Lodwick1999] Russell, S. and Lodwick, W. (1999). Fuzzy clustering in data mining for telco database marketing campaigns. In *18th International Conference of the North American Fuzzy Information Processing Society-NAFIPS (Cat. No. 99TH8397)*, pages 720–726. IEEE.

[Russell and Norvig2002] Russell, S. and Norvig, P. (2002). *Artificial intelligence : a modern approach*.

[Saunders and Miodownik1998] Saunders, N. and Miodownik, A. P. (1998). *CALPHAD (calculation of phase diagrams) : a comprehensive guide*. Elsevier Science.

[Seo and Sakawa2012] Seo, F. and Sakawa, M. (2012). *Multiple criteria decision analysis in regional planning : concepts, methods and applications*, volume 10. Springer Science & Business Media.

[Shi and Malik2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8) :888–905.

[Spall2005] Spall, J. C. (2005). *Introduction to stochastic search and optimization : estimation, simulation, and control*, volume 65. John Wiley & Sons.

[Steuer and Harris1980] Steuer, R. E. and Harris, F. W. (1980). Intra-set point generation and filtering in decision and criterion space. *Computers & Operations Research*, 7(1-2) :41–53.

[Taboada and Coit2006] Taboada, H. A. and Coit, D. W. (2006). Data mining techniques to facilitate the analysis of the Pareto-optimal set for multiple objective problems. In *IIE Annual Conference. Proceedings*. Institute of Industrial and Systems Engineers (IISE).

[Törn1980] Törn, A. A. (1980). A sampling-search-clustering approach for exploring the feasible/efficient solutions of MCDM problems. *Computers & Operations Research*, 7(1-2) :67–79.

[Torres et al.2011] Torres, R., Bès, C., Chaptal, J., and Hiriart-Urruty, J.-B. (2011). Optimal, Environmentally-Friendly Departure Procedures for Civil Aircraft. *Journal of Aircraft*, 48(1) :11–22.

[Van Dyke and Asaki2013] Van Dyke, B. and Asaki, T. (2013). Using QR Decomposition to Obtain a New Instance of Mesh Adaptive Direct Search

with Uniformly Distributed Polling Directions. *Journal of Optimization Theory and Applications*, 159(3) :805–821.

[Von Luxburg2007] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416.

[Wolsey and Nemhauser1999] Wolsey, L. A. and Nemhauser, G. L. (1999). *Integer and combinatorial optimization*, volume 55. John Wiley & Sons.

[Xu and Wunsch2005] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3) :645–678.

[Yu1974] Yu, P. (1974). Cone convexity, cone extreme points and nondominated solutions in decision problems with multi-objectives. *Journal of Optimization Theory and Application*, 14 :319–377.

[Yu and Leitmann1974] Yu, P.-L. and Leitmann, G. (1974). Compromise solutions, domination structures, and Salukvadze's solution. *Journal of Optimization Theory and Applications*, 13(13) :362–378.

[Zeleny1973] Zeleny, M. (1973). Compromise programming, multiple criteria decision-making. *Multiple criteria decision making. University of South Carolina Press, Columbia*, pages 263–301.

[Zghal et al.2011] Zghal, W., Audet, C., and Savard, G. (2011). A New Multi-Objective Approach for the Portfolio Selection Problem with Skewness. In Lee, C., editor, *Advances in Quantitative Analysis of Finance and Accounting*, volume 9, chapter 12, pages 317–355. Airiti Press.

[Zhang et al.2014] Zhang, J., Fu, M., Zhang, Z., Tao, J., and Fu, W. (2014). A trade-off approach of optimal land allocation between socio-economic development and ecological stability. *Ecological Modelling*, 272 :175–187.

[Zimmermann2012] Zimmermann, H.-J. (2012). *Fuzzy set theory—and its applications*. Springer Science & Business Media.

[Zitzler and Thiele1999] Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms : a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4) :257–271.