MASTER EN SCIENCES INFORMATIQUES À FINALITÉ SPÉCIALISÉE EN DATA SCIENCE

Interprétabilité/Explicabilité des IA pour la Méta-Modélisation

Application au domaine de l'agro-écologie

Mattens, Simon

Award date:
2021

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2020–2021

Interprétabilité/Explicabilité des IA pour la Méta-Modélisation. Application au domaine de l'agro-écologie

Simon MATTENS

Maîtres de stage : Tassadit BOUADI, Véronique MASSON, Luis GALARRAGA DEL PRADO

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Benoît FRENAY

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Préface

En premier lieu, je remercie mon directeur de mémoire, Benoît FRENAY, pour sa supervision et ses conseils prodigués le long de cette recherche.

Ensuite, je tiens à exprimer ma profonde gratitude et toute ma reconnaissance à mes maîtres de stage Tassadit BOUADI, Véronique MASSON et Luis GALARRAGA DEL PRADO pour leur encadrement, leur confiance qu'ils m'ont accordée ainsi que leurs conseils judicieux. Je leur adresse spécialement mes remerciements pour leur compréhension dans le contexte particulier de la crise sanitaire.

Je remercie aussi mes chers parents Anne et Jean-Michel qui ont toujours été là pour moi et remercie spécialement mon père pour m'avoir donné goût à l'informatique dans ma jeunesse.

Enfin, je voudrais exprimer ma reconnaissance envers mes amis qui m'ont apporté leur soutien moral tout au long de mes études.

Résumé

Dans le contexte de la modélisation et de la simulation de systèmes dits complexes, le modèle est vu comme une abstraction du système en question, et la simulation comme un plan d'expérience permettant l'exploration d'une multitude de scénarios (i.e. exploration de l'espace des paramètres en entrée du modèle). Ces modèles souvent très complexes sont vus par l'utilisateur comme des boîtes noires. Dans le cadre de la problématique de ce mémoire, des experts du domaine agro-écologique cherchent à proposer un modèle simplifié d'un modèle existant : le simulateur TNT2. Ce simulateur permet de simuler l'impact de pratiques agricoles sur la pollution nitrique des sols. L'étude présentée dans ce rapport se concentre sur les techniques d'extraction de sous-ensembles de scénarios, appelés ensembles d'apprentissage, à partir d'un jeu de données de départ. Ces ensembles d'apprentissage de taille minimale sont utiles pour réduire le temps de calcul d'un nouveau scénario à simuler mais aussi pour rechercher, dans de futures études, des règles d'apprentissage sur le simulateur TNT2. Ces règles permettraient par après de fournir un méta-modèle aux scientifiques. L'étude exploratoire exposée dans ce mémoire débute par un état de l'art assez conséquent sur ce qui est déjà réalisé dans le domaine (optimisation multiobjectif, règles d'apprentissage, méthodes "post-hoc"...). La suite de la recherche théorique débouche sur l'idée de fusionner l'idée principale de la méthode du Submodular pick avec de l'optimisation multiobjectif. L'objectif du Submodular pick consiste à renvoyer un sous-ensemble de scénarios représentatif de la population totale sur la base de l'idée de couverture d'un ensemble. Par la suite, trois fonctions objectives pour le problème étudié ont été élaborées et une preuve de concept sur une base de données fictives a été produite.

Mots-clés : Méta-modèle, Interprétabilité, Simulations, Submodular pick, Optimisation multiobjectif, Solutions optimales de Pareto, Front de Pareto.

Table des matières

1 Introduction

1.1 Contexte et problématique

Depuis quelques dizaines d'années, les algues vertes prolifèrent sur les plages de Bretagne. Ce phénomène n'est pas à prendre à la légère car il peut provoquer une asphyxie de la faune et de la flore aquatique. De plus, la décomposition de ces algues émet des gaz toxiques à des concentrations pouvant être mortelles pour l'être humain en quelques minutes d'inhalation. Les causes directement liées à cette prolifération sont l'élevage et la culture intensive en Bretagne : énormément de nitrates se répandent dans les sols et les cours d'eau des parcelles agricoles et c'est ceux-ci qui font proliférer les algues vertes. Il existe un modèle de simulation de type "boîte noire" appelé simulateur TNT2 qui est utilisé par des chercheurs et des experts du domaine de l'agro-écologie depuis quelques années dans le cadre de cette problématique (voir section 3 de ce mémoire pour plus d'informations). C'est un simulateur, composé d'ensembles de fonctions elles-mêmes très complexes, qui synthétise diverses connaissances agronomiques établies à partir d'observations très nombreuses. Il prend en paramètre un ensemble de conditions sur un bassin versant et retourne un ensemble de flux sortants, en particulier en terme de nitrates, de ce bassin versant. Ces flux sont calculés par le simulateur TNT2 pour 30 années consécutives. Ce simulateur a pour but d'aider les scientifiques à étudier des combinaisons de facteurs pouvant influencer la concentration d'azote dans les sols et par conséquent, à servir de base théorique pour ensuite mener des mesures pouvant réduire celle-ci. Il est validé et performant mais nécessite un temps de calcul d'environ une heure pour simuler un scénario (ensemble de variables définies pour toutes les parcelles agricoles d'un bassin versant). Les scientifiques cherchent à pouvoir employer le simulateur TNT2 de manière interactive lors de réunions avec des acteurs politiques, des gestionnaires de bassins versants... Il est nécessaire pour eux de pouvoir donc simuler rapidement un nouveau scénario via TNT2 afin que le simulateur apporte des éléments de réponse à une situation précise pouvant éclairer les acteurs décisionnaires dans leur prise de décision. L'objectif est donc de réduire drastiquement le temps de calcul d'une simulation. Pour ce faire, il apparaît deux solutions : soit simplifier le simulateur en lui-même, soit construire un méta-modèle en extrayant des règles de fonctionnement qui expliquent le fonctionnement global du simulateur. Dans le cadre de cette étude, c'est la deuxième option qui a été retenue. Afin d'obtenir ces règles, il faut travailler avec un ensemble d'apprentissage constitué de scénarios (déjà simulés) représentatif de l'ensemble de simulations original. Cet ensemble de simulations minimal doit couvrir le plus d'espace possible de l'ensemble de simulations total (ensemble de taille très importante puisqu'il contient de très nombreuses variables d'entrées dont beaucoup sont numériques, les possibilités de simulation sont donc infinies). En résumé, la problématique de cette étude est de trouver comment construire cet ensemble d'apprentissage optimal et minimal qui servira, dans de futures études, à construire un méta-modèle avec une méthode d'apprentissage à base de règles. Afin de pouvoir comparer des scénarios entre eux selon certains critères pour conserver les plus optimaux dans les ensembles d'apprentissage, l'optimisation multiobjectif est au coeur de ce mémoire.

1.2 Contributions apportées

Comme cette étude est exploratoire, une importante partie “Etat de l’art” a été réalisée dans le but de donner aux lecteurs l’intuition de plusieurs concepts théoriques représentant ce qui a déjà été fait dans le domaine. Afin de pouvoir répondre à la problématique de cette étude à savoir trouver les scénarios les plus représentatifs de la population de départ et les regrouper dans un ensemble de taille minimale, il est nécessaire d’avoir la capacité de les comparer entre eux suivant certains critères définis. De ce fait, un rapprochement entre ces différents critères et le domaine de l’optimisation multiobjectif a été effectué. Au cours de cette investigation, trois fonctions objectives ont été construites : une fonction de couverture, une de coût et une de distance qui sont présentées respectivement aux sections 4.2.1, 4.2.2 et 4.2.3 de ce mémoire. La fonction de couverture calcule un score indiquant à quel point un scénario couvre un ensemble de situations possibles, la fonction de coût retourne le coût de simulation des variables de sortie d’un scénario tandis que la fonction de distance mesure la proximité entre un scénario et son plus proche centroïde. Enfin, une preuve de concept a été réalisée sur des données fictives. Cette preuve est disponible à la partie 5 de ce mémoire. L’importante partie “Etat de l’art” de ce mémoire pourrait servir de base à de potentielles futures études dans le domaine pour essayer les concepts présentés sur les données réelles de la problématique.

1.3 Organisation du mémoire

Ce mémoire débute par une présentation de l’état de l’art (apprentissage automatique, interprétabilité, approches “post-hoc”, optimisation mutliobjectif). Ensuite, une description synthétique du simulateur TNT2 est proposée. Cette partie est suivie du chapitre consacré aux contributions apportées et puis, pour terminer, une conclusion a été rédigée.

2 Etat de l'art

Dans cette section, des notions utiles à la bonne compréhension du champ d'études de la problématique analysée sont introduites. Certaines de ces notions seront exploitées dans les contributions qui ont été apportées (voir section 4 de ce mémoire).

2.1 Apprentissage automatique

L'exploration de données (en anglais *data mining*) est un sous-domaine interdisciplinaire de l'informatique et des statistiques dont l'objectif global est d'extraire des informations, avec des méthodes intelligentes, à partir d'un ensemble de données et de transformer ces informations en une structure compréhensible pour une utilisation extérieure [1].
L'apprentissage automatique (en anglais *machine learning*) est un champ d'études de l'informatique et plus précisément de l'intelligence artificielle [2].
Les algorithmes dits de machine learning se distinguent des autres algorithmes par leurs facultés à réaliser et améliorer des prédictions ou des comportements en se basant sur des données. Ces données, appartenant à un même jeu de données, sont nommées instances et sont composées d'un certain nombre de variables (en anglais *features*) qui sont utilisées pour apprendre un modèle appliqué dans une tâche telle que la classification, la régression... Il existe différentes tâches de machine learning : apprendre un modèle, des structures, des règles... Dans la suite de ce mémoire, l'étude se focalisera sur des modèles de machine learning dont la tâche est prédictive.

Il existe deux grands paradigmes de machine learning :

— L'apprentissage supervisé où les modèles de machine learning sont conçus sur la base de données étiquetées (les exemples sont labellisés).

— L'apprentissage non-supervisé où les modèles sont conçus à partir de données non-étiquetées. Les algorithmes de regroupement (en anglais *clustering*) appartiennent à ce paradigme.

Les techniques prédictives (apprentissage supervisé) visent à construire des modèles pouvant prédire les classes ou les valeurs de nouveaux exemples tandis que les techniques descriptives (apprentissage non-supervisé) cherchent à trouver des structures (en anglais *pattern*) compréhensibles à partir d'un jeu de données. L'aspect prédictif est rendu possible grâce au machine learning tandis que celui descriptif avec des techniques de data mining a pour but de trouver des différences interprétables et discriminantes entre des classes de données.

2.2 Interprétabilité

Un système/modèle est dit interprétable si sa logique est compréhensible aux humains. C'est donc une qualité/propriété qu'un modèle possède intrinsèquement ou non. Il n'y a pas encore une définition formelle de cette notion dans la littérature mais certains modèles comme les réseaux de neurones sont considérés comme non-interprétables tandis que les arbres de décision sont eux considérés

comme interprétables. Les modèles non-interprétables sont appelés “boites noires”. L'explicabilité est la capacité à décrire la logique d'un système lorsque celui-ci est appliqué dans un cas particulier. L'idée réside dans le fait d'expliquer le plus possible aux humains pourquoi un ensemble de données d'entrée produit une certaine sortie sur un modèle mais aussi de mettre en avant certains patterns de données qui se dégagent d'autres et qui seraient plus pertinents pour la problématique étudiée. Plusieurs techniques de visualisation sont employées à ce jour avec chacune leurs avantages dépendant des explications mises en avant pour les utilisateurs. Dans le cas où le modèle étudié est d'emblée trop complexe pour être interprétable de prime abord, des techniques d'interprétabilité “post-hoc” (voir section 2.4 de ce mémoire) peuvent s'appliquer : elles analysent le modèle après l'apprentissage pour fournir des explications.

2.3 Machine learning interprétable

Les modèles de machine learning interprétables sont de complexité réduite, simples à comprendre pour l'être humain comme les arbres de décision ou les modèles linéaires par exemple. Dans le cadre de cette étude, les règles d'apprentissage ont été retenues car grâce à celles-ci, les scientifiques auront plus de facilités à comprendre des corrélations entre des variables qui influencent fortement l'apport nitrique des sols.

2.3.1 Règles d'apprentissage

En machine learning, il existe un ensemble de méthodes qui se basent sur des règles pouvant identifier certains patterns liés aux données transmises au modèle étudié [2] [3]. Celles-ci sont considérées comme des approches intrinsèques d'interprétabilité et tendent à représenter collectivement les connaissances capturées par le système afin de découvrir des relations intéressantes entre des variables. Elles sont de la forme {SI 'conditon' ALORS 'prédiction'} et sont générées automatiquement par un algorithme. Ces règles constituent probablement la forme d'interprétabilité la plus éloquente pour des modèles car elles sont faciles à comprendre pour l'être humain.
Plus formellement, une règle d'association est définie comme suit [4]
Soit $I = \{i_1, ..., i_n\}$ un ensemble de n attributs nommés *items*, soit $D = \{t_1, ..., t_m\}$ un ensemble de transactions. Chaque transaction t_i de D a un identifiant unique et contient un sous-ensemble d'*items* de I ($I \subseteq t_i$). Un *itemset* est un sous-ensemble $\subseteq I$.
Une règle d'association est définie par $X \rightarrow Y$ où $X, Y \in D$ et $X \cap Y \neq \emptyset$.
Pour illustrer, dans un contexte de supermarché, si $I = \{$lait, pain, beurre, soda$\}$, si D est un ensemble de transactions où chaque identifiant unique représente un client et le reste de la transaction vaut la présence/absence des *items* de I dans le panier du client (par exemple la transaction du premier client qui a acheté un pain, du beurre et un soda vaudrait $t_1 = \{1, 0, 1, 1, 1\}$) alors une règle pourrait être $\{$beurre, soda$\} \rightarrow \{$pain$\}$.

Voici ci-dessous quelques concepts intéressants définis dans la théorie des règles d'apprentissage [3] :

Soit X, Y deux *itemset*, une règle d'association $X \rightarrow Y$ et D un ensemble

de transactions d'une base de données :

La **fréquence**(X, Y) représente la fréquence à laquelle tous les *items* de X et Y sont vérifiés dans D et fréquence(X) représente la fréquence à laquelle les *items* de X apparaissent dans D.

Une autre notion intéressante est celle du **support** qui est définie par

$$\text{support}(X, Y) = \frac{\text{fréquence}(X, Y)}{|D|}$$

Moins formellement, le support correspond au pourcentage d'instances qui vérifient la condition X dans le jeu de données D.

La **confiance** $(X \to Y)$ indique une estimation du maximum de vraisemblance de la probabilité conditionnelle $P(X|Y)$ qui est estimée par

$$\frac{\text{fréquence}(X, Y)}{\text{fréquence}(X)}$$

Moins formellement, la confiance d'une règle peut se définir comme une mesure de la précision de la règle dans la prédiction de la classe correcte pour les instances auxquelles la condition de la règle s'applique.

Enfin, une règle $X \to Y$ est dite **productive** ssi

$$\forall Z \subset X \text{ confiance}(Z \to Y) < \text{confiance}(X \to Y)$$

2.4 Méthodes d'explication d'une boite noire

Dans le cas où le modèle de machine learning est trop complexe pour pouvoir l'interpréter de prime abord (le modèle est dit "boite noire"), une solution pour pouvoir l'interpréter est d'utiliser un module qui explique la logique du modèle à posteriori. Cette solution fait appel à des méthodes dites "post-hoc" qui créent un nouveau modèle interprétable (en anglais *proxy*) à partir du modèle boite noire étudié. Ce nouveau modèle peut être vu comme la couche visible de l'iceberg : malgré une perte de précision, le proxy est interprétable par l'être humain. Il existe différents types de proxy : des proxy d'explications locales ou globales (notions définies dans la suite de cette section), des proxy agnostiques (notion définie dans la suite de cette section), des proxy adaptés à certains types de boites noires... Dans le cadre de la problématique étudiée, le simulateur TNT2 est de type "boite noire". C'est pour cette raison que certaines méthodes "post-hoc" sont introduites dans la suite de cette section.

2.4.1 Méthodes agnostiques d'explication

Les modèles d'explication agnostiques sont des modèles pouvant être utilisés quel que soit le type de modèle d'apprentissage. Les explications fournies sur ces modèles sont donc applicables pour tous les modèles d'apprentissage. Les modèles spécifiques (i.e. non-agnostiques) sont quant à eux des modèles qui ne peuvent être utilisés que pour interpréter une famille spécifique d'algorithmes. Dans la suite de ce mémoire, les modèles agnostiques seront les seuls modèles

pris en compte afin de donner un caractère général à l'étude réalisée et aux contributions apportées.

2.4.2 Explications locales/ globales

Les méthodes d'explication **locale** [5] [6] produisent une explication pour une seule instance du jeu de données à la fois. Intuitivement, cela correspond à expliquer comment le modèle étudié se comporte dans le voisinage de l'instance qui a été choisie pour l'explication. A contrario, les méthodes **globales** [5] [6] permettent d'expliquer toutes les observations du jeu de données en même temps afin d'avoir une vision plus large sur le modèle étudié et de pouvoir approcher le fonctionnement global de l'algorithme. A noter que la fidélité locale n'implique pas forcément la fidélité globale : les variables d'entrée les plus pertinentes pour une méthode peuvent ne pas être les mêmes pour l'autre et vice versa.

Dépendant du type de modèle étudié, deux approches peuvent avoir lieu :

— Si le modèle est relativement simple et facile à interpréter, il est possible d'obtenir une explicabilité globale.

— Si le modèle est d'emblée très complexe, il est possible de le rendre plus interprétable en focalisant l'analyse sur les voisinages afin d'avoir une explicabilité locale.

2.4.3 LIME

L'algorithme de LIME (en anglais *Local Interpretable Model-agnostic Explanations*) est un algorithme qui est utilisé pour expliquer des prédictions individuelles sur un modèle de machine learning de type "boîte noire" [2] [5].
L'idée principale est qu'il construit un modèle interprétable - généralement un modèle linéaire - autour d'une seule instance du jeu de données de départ afin d'en extraire des explications locales autour de celle-ci.

Pour plus de détails, voici ci-dessous les différentes étapes du fonctionnement de la méthode de LIME :

1. Choisir une instance x dans le jeu de données de départ (selon des préférences des utilisateurs ou via la méthode du Submodular Pick - définie à la section 2.4.5 de ce mémoire) pour laquelle des explications locales sont souhaitées.

2. Générer des perturbations (nouvelles instances) autour de cette instance x à l'aide d'une fonction de perturbation.

3. Prédire la classe de chaque instance nouvellement créée avec le modèle de machine learning de type "boîte noire" pour lequel l'interprétabilité est souhaitée.

4. Attribuer un poids à chaque instance nouvellement créée en fonction de sa proximité avec l'instance x : plus la distance est petite, plus le poids est grand et inversement. Les instances les plus proches de l'instance x ont donc un poids plus important, ce qui est motivé par la volonté de créer un nouveau modèle le plus précis et fiable possible autour de l'instance x.

5. Entraîner un modèle linéaire sur base de ces nouvelles instances et leurs poids attribués.

6. Expliquer la prédiction en interprétant le modèle local, c'est-à-dire en regardant les contributions de chaque feature utilisée pour la construction du nouveau modèle.

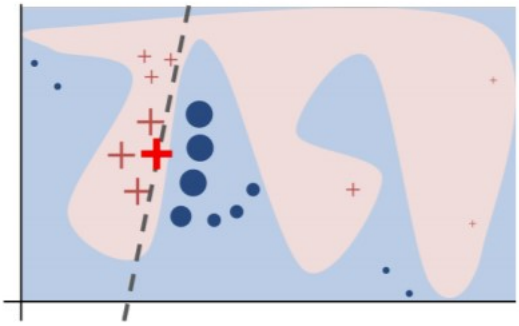

FIGURE 1 – Intuition de la méthode LIME. Source : [5]

Dans la figure 1, la fonction de décision f du modèle étudié (inconnue de LIME) est représentée par les couleurs roses et bleues en fond. L'instance x choisie pour l'explication est représentée par la croix rouge en gras. Les autres croix et ronds ont été créés à partir de f et ont une densité de surface proportionnelle à leurs poids. La droite noire en pointillé est l'explication locale retournée par LIME.

Plus formellement, l'explication produite par LIME pour une instance x du jeu de données de départ se définit comme suit [5]

$$\text{explication}(x) = \text{argmin}_{g \in G} L(f, g, \pi_x) + \Omega(g)$$

où :

— g est un modèle de machine learning et G représente un ensemble de modèles interprétables (modèle linéaire, arbre de décision, etc). Le domaine de g vaut $\{0,1\}^{d'}$ ce qui équivaut à la présence/absence de certains éléments interprétables pour l'être humain.

— $\Omega(g)$ correspond à la mesure de la complexité du modèle g (la complexité est l'inverse de l'interprétabilité). Elle peut par exemple, dans le cas d'un arbre de décision, correspondre à la profondeur de l'arbre ou dans le cas

d'un modèle linéaire, correspondre au nombre de poids différents de zéro.

— f est la boite noire que nous voulons expliquer et $f(x)$ est la probabilité que l'instance x appartienne à une certaine classe du problème.

— $\pi_x(z)$ correspond à la mesure de la proximité entre une nouvelle instance créée z et l'instance x choisie pour l'explication.

— $L(f, g, \pi_x)$ est la mesure de l'imprécision du modèle g pour approximer f dans la localité de x définie par π_x.

En résumé, pour garantir l'interprétabilité et la fidélité locale, il faut minimiser $L(f, g, \pi_x)$ tout en gardant $\Omega(g)$ bas pour que le modèle et les résultats soient interprétables par les humains.

Une sortie de l'algorithme LIME est une explication, reflétant la contribution de chaque feature à la prédiction d'un échantillon de données. Cela permet une interprétabilité locale et également de déterminer les changements de feature qui auront le plus d'impact sur la prédiction. Un exemple de sortie de LIME est représenté à la figure 2.

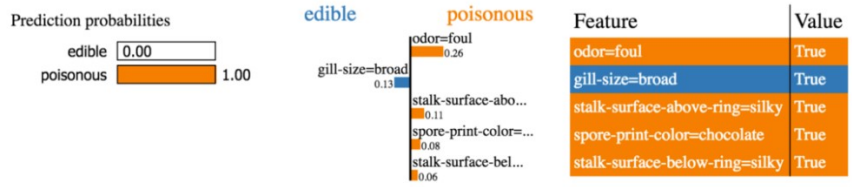

FIGURE 2 – Exemple de sortie de l'algorithme de LIME appliqué sur un problème de classification. Source : *https:// github.com/ marcotcr/ lime*

Le principal inconvénient de la méthode de LIME est lié à son fonctionnement local : LIME ne permet pas de généraliser l'interprétabilité issue du modèle local à un niveau plus global, ce qui veut dire qu'il ne peut pas s'adapter à une instance qui n'a pas encore été rencontrée.

2.4.4 Anchors

Pour pallier le problème rencontré dans LIME à savoir le manque de précision pour la prédiction d'une nouvelle instance, la méthode Anchors est introduite. C'est une méthode qui explique des prédictions localement pour n'importe quel modèle de type "boîte noire" en recherchant des règles d'apprentissage (voir section 2.3.1 de ce mémoire) qui ancrent (en anglais *anchor*) suffisamment la prédiction dans l'espace [2] [6]. L'algorithme va "découper" des zones limitées - les plus grandes possibles - dans l'espace à N dimensions du problème étudié où une nouvelle instance aura une grande probabilité d'être étiquetée d'un certain label dans ces zones. Le périmètre délimité par ces zones porte le nom de **couverture**. Ces zones seront construites sur base de règles dites d'Anchors,

règles de la forme {SI 'conditon' ALORS 'prédiction'}, qui doivent passer un test de **précision** pour pouvoir être validées : elles doivent avoir une valeur de précision supérieure à un seuil paramétré par les utilisateurs qui est souvent fixé à 95%.

Les principaux avantages de ces règles sont qu'elles sont faciles à interpréter et qu'elles ne dépendent pas des autres variables d'entrée une fois que la condition d'une règle a été vérifiée : les autres variables n'appartenant pas à la condition de la règle n'ont pas d'importance.

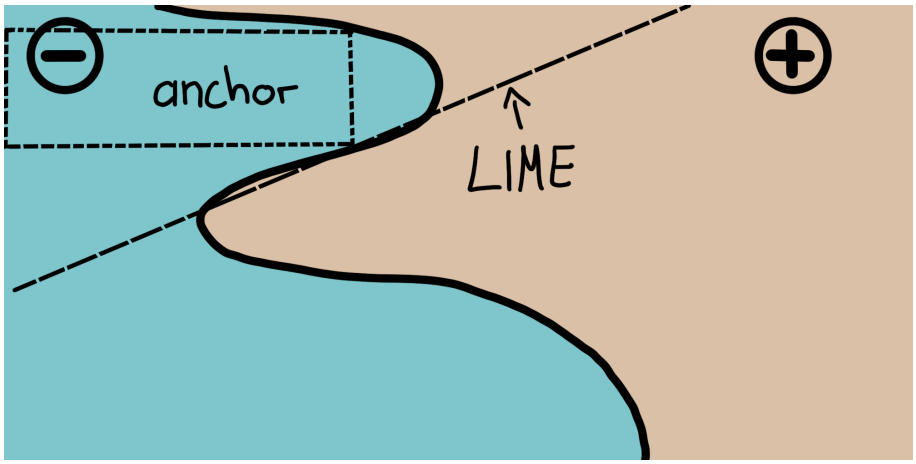

FIGURE 3 – Comparaison entre la représentation de LIME et celle d'Anchors. Source : [2]

Une des différences entre la méthode d'Anchors et celle de LIME est cette zone de l'espace délimitée par les règles Anchors. LIME construit un modèle linéaire basé sur une instance et ses perturbations mais ne va pas, à l'instar d'Anchors, quadriller une zone de l'espace où une future instance aura une grande probabilité d'appartenir à une certaine classe. C'est un avantage pour la prédiction de nouvelles instances qui n'ont encore jamais été rencontrées. Une représentation dans l'espace de ces deux méthodes est proposée à la figure 3.

Comme c'est une méthode locale, des perturbations D sont générées autour de l'instance x choisie. La fonction de perturbation D_x est très sensible aux différents types de données d'entrée et est très importante car elle influence fortement les règles Anchors créées.
Une Anchors A est formellement définie comme suit [6]

$$\mathbb{E}_{D_x(z|A)}[1_{f(x)=f(z)}] \geq \Gamma, A(x) = 1$$

où :

— x représente l'instance appartenant au jeu de données de départ choisie pour l'explication.

— A représente une Anchors c'est-à-dire un ensemble de prédicats tel que

$A(x) = 1$ quand toutes les variables des prédicats dans l'ensemble A correspondent aux valeurs des variables respectives dans l'instance x (peu importe les autres variables dans x qui ne sont pas représentées dans A).

— f est la boite noire à expliquer et $f(x)$ représente la classe de l'instance x prédite par le modèle f.

— $D_x(z \mid A)$ correspond aux instances z dans le voisinage de x (qui ont été générées avec la fonction de distribution D_x) qui vérifient la règle Anchors A.

— $0 \leq \Gamma \leq 1$ correspond au seuil de précision paramétré par les utilisateurs. Seules les règles qui parviennent au moins à cette précision sont considérées comme valides.

Calculer toutes les règles Anchors possibles à partir du voisinage de x est impossible dans les espaces continus et de grandes dimensions car cela demanderait beaucoup trop de temps de calcul. C'est pourquoi il faut passer par une définition probabiliste (paramètre δ dans la définition de la précision ci-dessous) : il faut échantillonner le voisinage de x et certains échantillons seront utilisés s'ils remplissent une condition de précision.
La **précision** d'une règle est la proportion d'exemples correctement classés par cette règle [6].

$$P(\text{précision}(A) \geq \Gamma) \geq 1-\delta \text{ avec précision}(A) = \mathbb{E}_{D_x(z|A)}[1_{f(x)=f(z)}] \; et \; 0 \leq \delta \leq 1$$

La **couverture** d'une règle est la proportion d'exemples couverts par cette règle [6]. Ce paramètre consiste à trouver des règles qui s'appliquent au plus large espace du modèle d'entrée afin de couvrir la zone la plus grande possible.

$$\text{couverture}(A) = \mathbb{E}_{D_z[A(z)]}$$

Etant donné que plusieurs règles Anchors sont générées, il faut trouver celles qui maximisent la couverture car elles sont plus intéressantes [6].

$$\max_{A \text{ tel que } P(\text{précision}(A) \geq \Gamma) \geq 1-\delta} \text{couverture}(A)$$

Les règles Anchors qui ont beaucoup de prédicats tendent à être plus précises que celles avec moins de prédicats mais sont plus spécifiques et couvrent donc moins d'espace. En résumé, Anchors utilise deux notions et fait un compromis entre elles : la notion de **couverture** et celle de **précision**.

Pour plus de détails, voici ci-dessous les différentes étapes du fonctionnement de la méthode d'Anchors :

1. Générer de nouveaux candidats : au début de l'algorithme, un candidat par variable d'entrée de l'instance x choisie pour l'explication est créé. Ensuite, les meilleurs candidats de l'itération précédente sont conservés dans la règle Anchors qui se voit agrandie par un prédicat qui n'appartenait pas encore à cet ensemble. Par exemple si $x =$ "Ce film est mauvais", au début de l'algorithme 4 candidats sont créés représentant chacun un

mot appartenant à la phrase de x (c_1= "Ce", c_2= "film", c_3 = "est", c_4= "mauvais"). A chaque itération de l'algorithme, l'ensemble des meilleurs candidats (anchor) trouvé jusqu'à présent est augmenté par le meilleur candidat de l'itération en cours jusqu'à ce que cet anchor réussit le test de précision.

2. Identifier les meilleurs candidats : comparer les règles Anchors entre elles et conserver celle qui est la plus précise pour l'instance x.

3. Vérifier si la meilleure règle Anchors trouvée à l'étape précédente réussit le test de précision. Si c'est le cas, arrêter l'algorithme et retourner cette règle Anchors. Sinon, continuer vers la prochaine étape.

4. Effectuer une recherche par faisceau : transporter les B meilleurs candidats vers la prochaine itération (étape 1). Ces candidats sont utilisés pour créer de nouvelles règles. Cette recherche permet d'éviter des optimaux locaux.

Cet algorithme retourne les règles Anchors les plus petites possibles afin qu'elles soient les plus génériques possibles pour qu'elles couvrent un maximum d'espace, pour qu'elles aient donc une meilleure couverture.

La sortie de l'algorithme d'Anchors est un ensemble de règles de la forme {SI 'conditon' ALORS 'prédiction'}. Un comparatif des sorties de l'algorithme de LIME et celles d'Anchors est proposé à la figure 4.

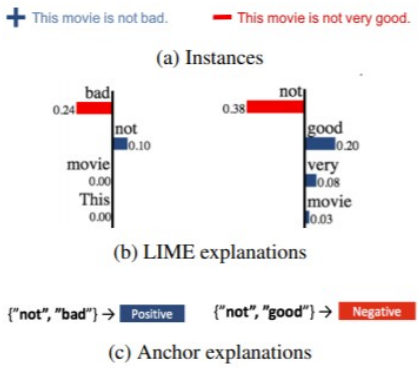

FIGURE 4 – Comparaison entre des exemples d'explications de LIME et celles d'Anchors. Source : [6]

Malgré le fait que les règles Anchors soient faciles à interpréter, il y a quelques désavantages :

1. Lorsque les prédictions sont à la limite entre deux zones Anchors différentes, cela peut mener à des soucis de classification. De plus, si une classe est sous-représentée dans un jeu de données, les règles Anchors liées à cette

classe risquent d'être très complexes et de ne pas couvrir assez l'espace de dimensions.

2. Différentes règles Anchors peuvent se contredire pour la même instance. Pour pallier cela, augmenter la valeur de Γ serait pertinent.

3. Lorsque l'espace de dimension est très grand, la représentation dans l'espace liée à Anchors risque de ne pas être claire au vu du nombre de zones délimitéés par les règles Anchors ainsi que la dimensionalité.

4. Il est difficile de concevoir une bonne fonction de distribution. En effet, les explications retournées par Anchors sont très sensibles à cette fonction. Celle-ci doit s'adapter au mieux aux types de données présentes dans la base de données de la boite noire et ce n'est pas toujours évident.

En conclusion, les règles Anchors produisent des explications faciles à comprendre pour l'être humain et ces règles sont faciles à interpréter. Malgré cela, cette approche reste très sensible à la configuration de ses hyperparamètres mais aussi à la définition de sa fonction de perturbation de ses données d'entrée. Le but ultime de cette approche serait de concevoir des règles permettant de couvrir tout l'espace du problème. De ce fait, la prédiction de nouvelles instances serait aisée.

2.4.5 Submodular pick

Les méthodes telles que LIME et Anchors visent à expliquer localement des prédictions. Or, fournir des explications plus globales sur un modèle afin de le considérer comme un tout pour pouvoir avoir confiance en ce qu'il produit de manière générale et comprendre comment ce modèle fonctionne au sens large est une approche privilégiée. La méthode du Submodular pick s'applique à des modèles agnostiques et est jugée pertinente dans le cadre de la problématique étudiée. Concrètement, cette méthode construit à partir du modèle choisi un ensemble d'instances qui est lui-même un sous-ensemble du jeu de données de départ [5] [7] [8]. Le challenge important réside dans le fait de choisir ces bonnes instances afin qu'elles couvrent un maximum l'espace du modèle étudié et qu'elles ne soient pas redondantes entre elles. L'ensemble de ces instances doit être d'une taille raisonnable (B éléments) car doit rester compréhensible pour l'être humain.

La tâche de sélectionner B instances à expliquer dans un jeu de données X est appelée Pick step. Celle-ci devrait choisir un ensemble diversifié et représentatif d'explications à montrer à l'utilisateur, c'est-à-dire des explications non redondantes qui représentent le comportement du modèle en sa globalité.

$$B \leq |X|$$

Pour plus de détails, voici ci-dessous les différentes étapes du fonctionnement de la méthode du Submodular pick :

1. Construire une matrice W qui représente l'importance locale de chaque composant interprétable (variables - colonnes j de la matrice) de chaque instance (rangées i de la matrice).

$$W = n \times d^{'} \, (|X| = n, \; d^{'} = \# \text{ Variables interprétables})$$

2. Calculer l'importance globale I de chaque composant j dans l'espace explicatif. Plus ce score d'importance I_j est grand, plus cette variable j explique plusieurs instances différentes et est jugée plus intéressante. Concrètement, I est défini comme [5]

$$I_j = \sqrt{\sum_{i=1}^{n} W_{ij}}$$

Le but est donc de choisir des instances couvrant des variables importantes. Ces instances ne doivent pas être redondantes vis-à-vis d'autres instances qui ont déjà été selectionnées pour faire partie de l'ensemble retourné par la méthode du Submodular pick. Elles ont également les mêmes explications pour ces variables importantes.

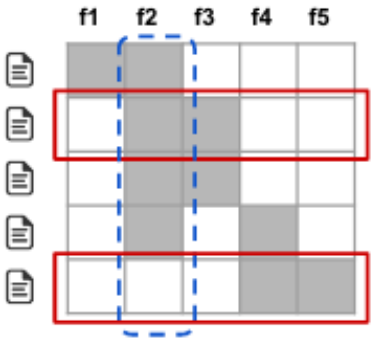

FIGURE 5 – Exemple de matrice binaire W ayant 5 variables et 5 instances. Source : [5]

Dans la figure 5, la fonction d'importance I attribue un score plus grand à $f2$ qu'à $f1$ ($I_2 > I_1$) car la variable $f2$ est utilisée pour expliquer plus d'instances que la variable $f1$ (4/5 pour $f2$ contre 1/5 pour $f1$ - zones grisées dans la matrice).
Si le Submodular pick doit retourner 2 instances parmi les 5 instances existantes, la première instance choisie sera soit la 2ème, la 3ème ou la 4ème car elles ont en commun la variable $f2$ qui est la plus intéressante et une autre variable ($f3$ ou $f4$) qui ont toutes les deux la même importance. Par ordre croissant d'exécution de la méthode, le Submodular pick choisira la 2ème instance. Une fois que la 2ème rangée a été choisie, il n'est pas pertinent de choisir la 3ème car celle-ci n'ajoute pas d'information supplémentaire : elle contient les mêmes valeurs de variables $f2$ et $f3$ que la rangée 2. Il est donc plus avantageux de choisir la

dernière rangée qui apporte deux nouvelles informations : les variables $f4$ et $f5$ (choisir la 1ère rangée ou la 4ème n'apporterait qu'une seule source d'information en plus). L'ensemble retourné par le Submodular pick est l'ensemble composé des instances 2 et 5 du jeu de données de départ si le Pick step a été fixé à 2.

Il y a donc cette notion de **couverture** dans la méthode du Submodular pick. Elle correspond à un ensemble c de fonctions qui, en transmettant W et I comme paramètres à cet ensemble, calcule l'importance totale des variables qui apparaissent au moins dans une instance d'un ensemble V [5].

$$c(V, W, I) = \sum_{j=1}^{d'} \mathbb{1}_{(\exists i \in V : W_{ij} > 0)} I_j$$

Le Pick problem est la tâche qui consiste à trouver un ensemble V, $|V| \leq B$ qui atteint la meilleure couverture [5].

$$\text{Pick}(W, I) = \text{argmax}_{V, |V| \leq B} c(V, W, I)$$

Le fonctionnement global de la méthode du Submodular pick est décrit dans l'algorithme 1. La notation explication$(x_i, x_i^{'})$ fait référence aux explications retournées par l'algorithme de LIME ou même celui d'Anchors.

Algorithm 1: Algorithme Submodular pick

Entrées: Jeu de données X, Budget B
Sortie : Ensemble V
forall $x_i \in X$ **do**
  $W_i = \text{explication}(x_i, x_i^{'})$
for $j \in \{1...d^{'}\}$ **do**
  $I_j = \sqrt{\sum_{i=1}^{n} W_{ij}}$
$V = \{\}$
while $|V| < B$ **do**
  $V = V \cup \text{argmax}_i c(V \cup \{i\}, W, I)$
return V

La méthode du Submodular pick est une méthode liée à un problème monoobjectif : retourner un ensemble V de B instances représentatives de la population X qui couvre le mieux l'espace de recherche. C'est donc un objectif de couverture. Dans le cadre de la problématique traitée, plusieurs fonctions objectives ont été décrites à la section 4.2 de ce mémoire. L'une d'entre elles est une fonction de couverture et a été inspirée par la méthode du Submodular pick. C'est à cette étape de la recherche qu'un lien entre la problématique étudiée et l'optimisation multiobjectif s'est construit.

2.5 Optimisation multiobjectif

L'optimisation multiobjectif est une branche de problèmes d'optimisation où plusieurs fonctions objectives sont définies [9] [10]. Ces problèmes sont fréquents dans le monde de l'industrie où des managers sont souvent forcés à atteindre

des objectifs qui peuvent rentrer parfois en conflit entre eux : par exemple un objectif pourrait être de maximiser le profit d'une entreprise et un autre limiter les émissions de gaz à effet de serre. Dans la suite de cette section, quelques concepts utiles à la problématique ainsi qu'un large panel de définitions vont être présentés.

2.5.1 Concepts de base

Il existe trois approches différentes dans l'optimisation multiobjectif dépendant du moment où l'utilisateur interagit ou apporte ses informations de préférence sur les fonctions objectives et/ou sur les solutions retournées :

1. L'approche **a priori** où un ordre total est défini sur l'espace objectif, par exemple en déterminant une fonction utilitaire $\mathbb{R}^m \to \mathbb{R}$ (m étant le nombre d'objectifs). La solution retournée pour ce type d'approche consiste en un point minimal et une valeur minimale concernant l'ordre établi. L'utilisateur peut apporter ses préférences de choix (comme faire correspondre à chaque objectif un poids) avant l'optimisation.

2. L'approche **a posteriori** : un ordre partiel est défini sur $\mathbb{R}^m$, généralement **l'ordre Pareto**. L'algorithme recherche l'ensemble minimal concernant cet ordre sur l'ensemble des solutions possibles. C'est après avoir été informé sur les solutions non dominées que l'utilisateur peut apporter ses préférences de choix.

3. L'approche **interactive** [11] : la prioritisation des différentes fonctions objectives est redéfinie à plusieurs moments de l'exécution de l'algorithme en demandant à l'utilisateur ses préférences à différents endroits.

Dans la suite, c'est l'approche **a posteriori** qui a été retenue étant donné que trois fonctions objectives ont été élaborées à la section 4.2 de ce mémoire, chacune retournant un score en fonction d'une instance donnée. Ces scores sont ensuite rassemblés afin de constituer une solution pour une instance. Pour pouvoir comparer ces solutions entre elles, un ordre partiel doit être défini.

Afin de fournir le plus d'informations possibles sur le sujet de l'optimisation multiobjectif, un certain nombre de définitions vont être introduites [9].

Definition 1 (Problème d'optimisation multiobjectif) *Soit m fonctions objectives $f_1 : \chi \to \mathbb{R}, ..., f_m : \chi \to \mathbb{R}$ qui font correspondre un espace de décision χ vers $\mathbb{R}$, un problème d'optimisation multiobjectif (MOP) peut être formulé comme suit :*

$$minimiser\ f_1(x), ..., f_m(x), x \in \chi$$

Definition 2 (Propriétés des relations binaires) *Soit un ensemble X, une relation binaire sur X - qui est un ensemble R $R \subseteq X \times X$ — est dite :*

— *réflexive, ssi* $\forall x \in X : (x, x) \in R$,
— *irréflexive, ssi* $\forall x \in X : (x, x) \notin R$,

— *symétrique, ssi* $\forall x \in X, \forall y \in X, (x,y) \in R \Leftrightarrow (y,x) \in R$,
— *asymétrique, ssi* $\forall x \in X, \forall y \in X, (x,y) \in R \Rightarrow (y,x) \notin R$,
— *antisymétrique, ssi* $\forall x \in X, \forall y \in X, (x,y) \in R \wedge (y,x) \in R \Rightarrow x = y$,
— *transitive, ssi* $\forall x \in X, \forall y \in X, \forall z \in X, (x,y) \in R \wedge (y,z) \in R \Rightarrow (x,z) \in R$.

Definition 3 (Ordres) *On dit qu'une relation binaire R est :*

— *Un pré-ordre ssi R est transitive et réflexive.*
— *Un ordre partiel ssi R est un pré-ordre antisymétrique.*
— *Un ordre partiel strict ssi R est irréflexive et transitive.*

Definition 4 (Dominance Pareto) *Soit R une relation d'**ordre partiel** définie sur* $\mathbb{R}^m$*, soit deux vecteurs dans l'espace objectif :* $y^{(1)} \in \mathbb{R}^m$ *et* $y^{(2)} \in \mathbb{R}^m$*, le point* $y^{(1)} \in \mathbb{R}^m$ ***Pareto domine*** *le point* $y^{(2)}$ *(en symbole* $y^{(1)} \prec_{Pareto} y^{(2)}$ *) ssi :*

$$\forall i \in \{1,..,m\} : y_i^{(1)} \leq y_i^{(2)} \text{ et } \exists j \in \{1,..,m\} : y_j^{(1)} < y_j^{(2)}$$

En d'autres termes, si $y^{(1)} \prec_{Pareto} y^{(2)}$ *alors cela veut dire que le premier vecteur n'est pas pire pour chaque objectif que le deuxième et est au moins meilleur pour un objectif que le deuxième.*

Dans les problèmes d'optimisation multiobjectif, deux espaces se font face : l'espace de décision (χ) qui comprend tous les candidats "solutions" et l'espace objectif $\mathbb{R}^m$ où les vecteurs des fonctions objectifs y sont représentés.
La fonction vecteur-valeur $f = (f_1, ..., f_m)^t$ fait correspondre χ vers $\mathbb{R}^m$.
Soit $x_1, x_2 \in \chi$. La solution x_1 Pareto domine x_2 ssi $f(x_1) \prec_{Pareto} f(x_2)$.
Notation : x_1 Pareto domine x_2 est dénoté par $x_1 \prec_f x_2$.

Definition 5 (Element minimal) *Un élément minimal* $x \in X$ *dans un ensemble (strictement) partiellement ordonné (X, R) est un élément pour lequel il n'existe pas* $x' \in X$ *avec* $x' Rx$ *et* $x' \neq x$

Definition 6 (Point et ensemble efficace) *Les éléments minimaux de l'ordre Pareto* $\prec_f$ *sur* χ *sont appelés des points efficaces et le sous-ensemble* χ_E *qui comprend tous les points efficaces dans* χ *est dénommé l'ensemble efficace.*
Un point $x \in \chi$ *est appelé faiblement efficace ssi il n'existe pas* $u \in \chi$ *tel que* $f(u) < f(x)$. $f(x)$ *est dit **faiblement non dominé**.*

Definition 7 (Front de Pareto) *Soit* $Y := f(\chi)$ *l'ensemble des vecteurs objectifs atteignables. Les éléments minimaux de l'ordre Pareto sur Y sont dits **non-dominés**. Le sous-ensemble de tous les vecteurs objectifs non-dominés dans Y est appelé le **front de Pareto**(*Y_N*).*
Ces points du front de Pareto sont les points les plus intéressants à trouver dans l'ensemble objectif car ne sont dominés par aucun autre point.

Le challenge important de l'optimisation multiobjectif est d'approcher au mieux ces points du front de Pareto.

Soit A et B deux ensembles d'approximation d'un front de Pareto dans $\mathbb{R}^m$. On dit que A est meilleur que B ssi chaque $b \in B$ est faiblement dominé (voir définition 6) par au moins un élément $a \in A$ et $A \neq B$. Notation : $A \triangleright B$.

Definition 8 (Ensemble d'approximation d'un front de Pareto) *Un ensemble fini A de $\mathbb{R}^m$ est un ensemble d'approximation d'un front de Pareto (ou une "couche") ssi A est composé de points (Pareto) non dominés mutuellement.*

Definition 9 (Cône) *Un ensemble $C \subset \mathbb{R}^m$ with $\emptyset \neq C \neq \mathbb{R}^m$ est appelé cône non-trivial ssi $\forall \alpha \in \mathbb{R}, \alpha > 0, \forall c \in C, \alpha c \in C$. Un exemple de cône est repris à la figure 6.*

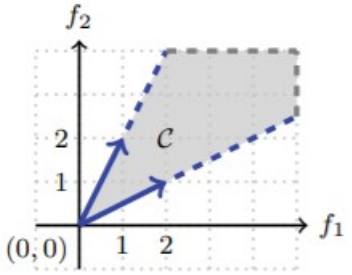

FIGURE 6 – Exemple de cône C. Source : [9]

Definition 10 (Somme de Minkowski) *La somme de Minkowski de deux ensemble $A \in \mathbb{R}^m$ and $B \in \mathbb{R}^m$ est définie par $A \oplus B := \{a + b | a \in A \wedge b \in B\}$. Une représentation de la somme de Minkowski est montrée à la figure 7.*

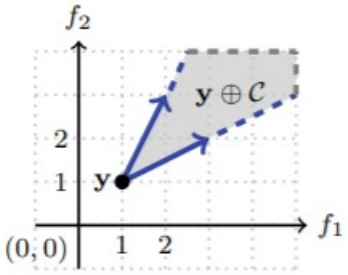

FIGURE 7 – Somme de Minkowski entre un singleton $\{y\}$ et un cône C. Source : [9]

Definition 11 (Orthant) *Soit m un nombre naturel plus grand ou égal à 1. Un ensemble orthant non-négatif de $\mathbb{R}^m$, dénoté par $\mathbb{R}^m_{\geq 0}$ est l'ensemble de tous les éléments de $\mathbb{R}^m$ qui ont leurs coordonnées non-négatives.*
L'orthant dominé par zéro ($\mathbb{R}^m_{\succ 0}$) est l'ensemble $\mathbb{R}^m_{\geq 0} \backslash \{0\}$.
Par opposition, l'ensemble orthant non-positif de $\mathbb{R}^m$, dénoté par $\mathbb{R}_{\leq 0}$ est l'ensemble des éléments dans $\mathbb{R}^m$ qui ont leurs coordonnées non-positives.
L'ensemble des éléments dans $\mathbb{R}^m$ qui dominent le vecteur zéro 0, dénoté par $\mathbb{R}^m_{\prec 0}$ est l'ensemble $\mathbb{R}^m_{\leq 0} \backslash \{0\}$.
L'ensemble des réels positifs est dénoté par $\mathbb{R}_{>0}$ et celui des non-négatifs par $\mathbb{R}_{\geq 0}$.

Definition 12 (Front de Pareto convexe) *Un front de Pareto est convexe ssi $Y_N \oplus \mathbb{R}^m_{\geq 0}$ est convexe. Y_N représentant le front de Pareto. Un exemple de front de Pareto convexe est montré à la figure 8.*

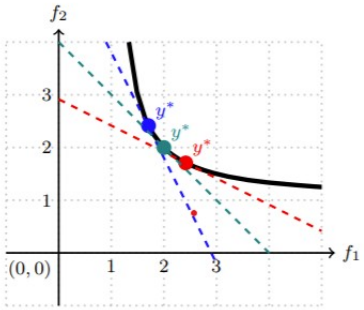

FIGURE 8 – Exemple de front de Pareto convexe (courbe noire en gras). Source : [9]

Definition 13 (Front de Pareto concave) *Un front de Pareto est concave ssi $Y_N \oplus \mathbb{R}^m_{\leq 0}$ est convexe. Y_N représentant le front de Pareto. Un exemple de front de Pareto concave est montré à la figure 9.*

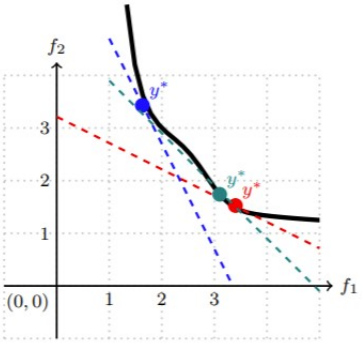

FIGURE 9 – Exemple de front de Pareto concave (courbe noire en gras). Source : [9]

Definition 14 (Scalarization) *La **scalarization** signifie que les différentes fonctions objectives d'un problème d'optimisation multiobjectif sont agrégées (ou reformulées comme des contraintes) pour n'avoir qu'un problème mono-objectif à résoudre. Il existe différentes techniques de scalarization avec chacune leurs avantages et inconvénients. En utilisant et modifiant les hyperparamètres de ces techniques, il est possible d'obtenir différents points du front de Pareto, qui sont donc les points les plus intéressants car dominés par aucun autre point.*

Definition 15 (Algorithmes génétiques) *Les algorithmes évolutionnistes sont une branche majeure des heuristiques de recherche basés sur la biologie.*
Ils sont utilisés pour résoudre des problèmes d'optimisation numérique combinatoire et non-convexe. Ils utilisent des paradigmes liés à l'évolution naturelle tels

que la sélection, la reproduction et la mutation à partir d'une population donnée d'individus afin de les faire évoluer et de rendre les prochaines générations plus intéressantes que les anciennes (plus proche du front de Pareto).

Les concepts les plus intéressants retenus dans la partie contribution de ce mémoire se résument à l'ordre Pareto, le front de Pareto (convexe et concave) et la scalarization.

2.5.2 Scalarization linéaire

La façon la plus simple de scalarizer un problème est d'attacher des poids ≥ 0 (dont au moins un > 0) à chaque fonction objective et ensuite minimiser la somme des fonctions objectives pondérées.
Formellement, la scalarization linéaire d'un problème multiobjectif (m objectifs) utilisant un vecteur poids $w \in \mathbb{R}^m_{\succ 0}$ s'écrit comme suit [9]

$$\text{Minimiser} \sum_{i=1}^{m} w_i f_i(x), x \in \chi$$

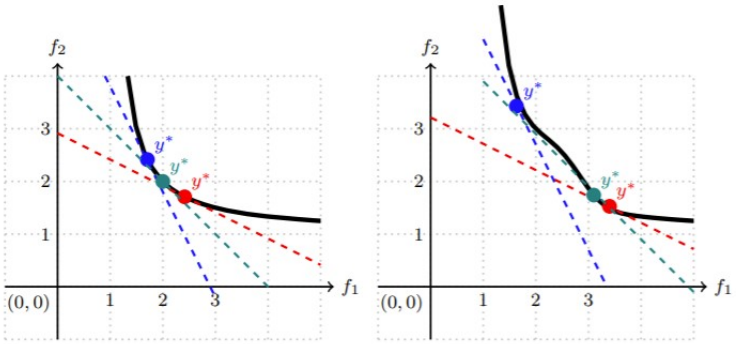

FIGURE 10 – Scalarization linéaire avec différents poids. Source : [9]

Cette scalarization est vraiment efficace lorsque le front de Pareto est convexe car elle permet de bien approcher les points de cette couche moyennant une bonne paramétrisation du vecteur poids.
Dans le cas du front de Pareto concave, elle n'est pas adaptée car certains points du front de Pareto ne pourraient pas être approchés par la scalarization linéaire : elle pourra atteindre seulement les points les plus extrêmes du front c'est à dire les meilleurs points de chaque fonction objective mais ne trouvera pas les points au milieu du front de Pareto. Cette intuition est montrée à la figure 10.

2.5.3 Scalarization de Chebychev

Cette méthode de scalarization permet d'approcher les zones concaves du front de Pareto en se basant sur la distance de Chebychev et d'un point référence dans l'espace des solutions.

La scalarization de Chebychev d'un problème multi-objectif utilisant un vecteur poids $\lambda \in \mathbb{R}^m_{\succ 0}$ se formule comme suit [9]

$$\text{Minimiser } \max_{i\in\{1,\dots,m\}}\lambda_i \mid f_i(x) - z_i^* \mid, x \in \chi$$

où z^* est un point référence (le point idéal) défini par $z_i^* = \inf_{x\in\chi} f_i(x)$ avec $i = 1...m$.

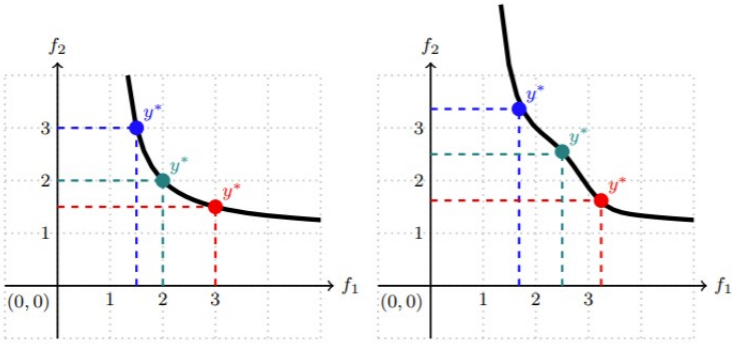

FIGURE 11 – Scalarization de Chebychev avec différents poids. Source : [9]

Pour un problème de minimisation, z^* peut prendre la valeur de l'origine du plan dans lequel on recherche les solutions tandis que pour un problème de maximisation, z^* peut prendre une valeur très grande qui tendrait vers l'infini. Une représentation de la scalarization de Chebychev est montrée à la figure 11.

2.5.4 Scalarization par contrainte ϵ

Etant donné un problème multiobjectif, la scalarization par contrainte ϵ est précisée comme suit [9]. Soit $m-1$ constantes données : $\epsilon_1 \in \mathbb{R}, .., \epsilon_m \in \mathbb{R}$,

$$\text{Minimiser } f_1(x), \text{ avec les contraintes } g_1(x) \leq \epsilon_1, ..., g_{m-1}(x) \leq \epsilon_{m-1}$$

où $f_1, g_1, .., g_{m-1}$ constituent les m composants du vecteur fonction f du problème d'optimisation multiobjectif.

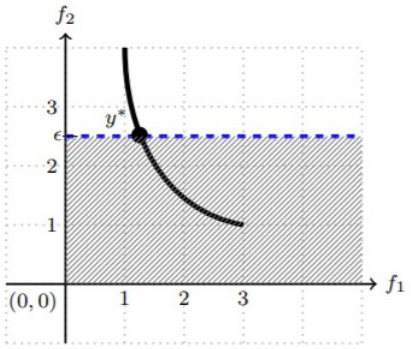

FIGURE 12 – Scalarization par contrainte ϵ où ϵ_1 vaut 2,5 . Source : [9]

Cette approche se focalise donc sur la minimisation d'une seule fonction objective en faisant passer les autres objectifs comme contraintes du premier. Une représentation de la scalarization par contrainte ϵ est montrée à la figure 12.

2.5.5 Scalarization par intersection de frontières

Soit $d \in \mathbb{R}^m_{\succ 0}$ un vecteur de direction et $r \in \mathbb{R}^m$ un vecteur référence. Le problème de scalarization par intersection de frontières se formule comme suit [9] [12] [13]

$$\begin{aligned} &\text{minimiser t,} \\ &\qquad \text{avec les contraintes} \\ &\qquad (a)\ r + td - f(x) = 0, \\ &\qquad (b)\ x \in \chi, et \\ &\qquad (c)\ t \in \mathbb{R}_{\geq 0} \end{aligned}$$

Les variables de cette définition seront explicitées dans la suite de ce mémoire, une fois que l'intuition de cette scaralization sera exposée.

Les principales étapes de cette méthode sont :

- Trouver le minimum individuel de chaque fonction objective.
- Construire le CHIM (en anglais *convex hull of individual minima*) en reliant un à un les minimums individuels dans l'espace de solutions.
- Construire les vecteurs normaux à chaque point du CHIM pointant en direction de l'origine du plan.
- Déterminer les points d'intersection entre ces vecteurs normaux et la frontière de l'espace de solution. Ces points d'intersection appartiennent au front de Pareto.

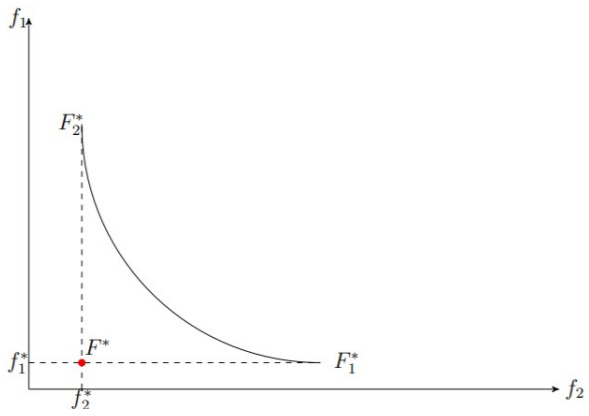

FIGURE 13 – Représentation de deux minimums individuels F_1^* et F_2^* pour un problème d'optimisation biobjectif . Source : [13]

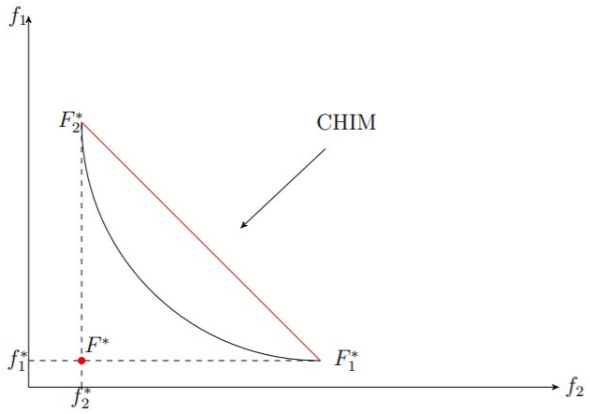

FIGURE 14 – Construction du CHIM à partir de F_1^* et F_2^* . Source : [13]

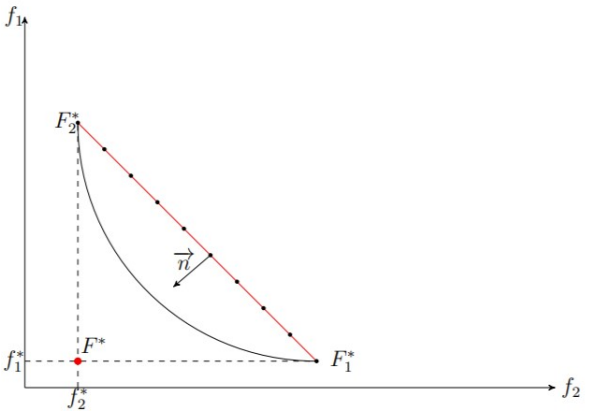

FIGURE 15 – Représentation d'un vecteur normal ($\vec{n}$) au CHIM . Source : [13]

Les figures 13, 14, 15 représentent les différentes étapes du fonctionnement de la scalarization par intersection de frontières. Pour commencer, la figure 13 représente deux minimums individuels F_1^* et F_2^* pour un problème d'optimisation biobjectif. Ensuite, la figure 14 montre la représentation du CHIM à partir de ces deux minimums individuels. Enfin, la figure 15 désigne un vecteur normal ($\vec{n}$) au CHIM. Les points du front de Pareto (qui est représenté par la ligne noire en gras dans les figures 13, 14, 15) sont situés à l'intersection entre les vecteurs normaux ($\vec{n}$) au CHIM et l'espace des solutions.

Une fois l'intuition de cette scalarization développée, les variables dans la définition évoquée en début de section peuvent se lier aux éléments inculqués dans les fi-

gures 13, 14, 15.

$$\begin{aligned} &\text{minimiser t,} \\ &\qquad \text{avec les contraintes} \\ &\qquad (a)\ r + td - f(x) = 0, \\ &\qquad (b)\ x \in \chi, et \\ &\qquad (c)\ t \in \mathbb{R}_{\geq 0} \end{aligned}$$

où :

— d est un vecteur normal au CHIM pointant vers l'origine du plan.
— t représente un point sur cette normale d.
— r représente un point du CHIM.
— $f(x)$ représente l'espace des solutions.

Cette méthode est fiable pour trouver des points sur la limite de l'espace des solutions (excepté pour les cas où le front de Pareto n'est pas convexe, voir figure 16) mais nécessite quelques connaissances préalables sur la zone du front de Pareto.

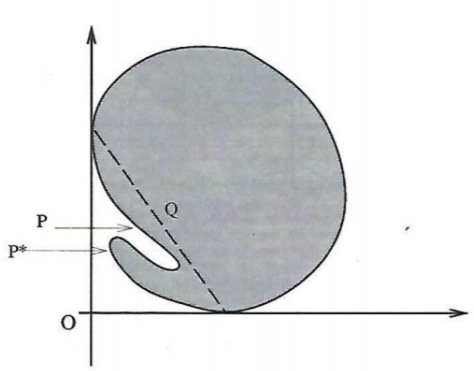

FIGURE 16 – Exemple de cas où la méthode ne trouvera pas le point P^* mais le point P en partant d'un certain point de Q (le CHIM). Source : [12]

2.5.6 NSGA-II

NSGA-II (en anglais *Non-dominated Sorting Genetic Algorithm*) est un algorithme génétique qui se base sur l'ordre de Pareto pour approcher au mieux le front de Pareto [9] [14]. Plus précisément, cet algorithme effectue un classement sur les individus en deux parties : la première partie du classement concerne les relations de dominance Pareto tandis que la deuxième se concentre sur la notion de diversité, notion qui s'applique sur les individus qui partagent le même ensemble d'approximation du front de Pareto.

C'est un algorithme qui se base principalement sur trois idées :

— Il utilise le principe de l'élitisme qui se définit comme la conservation des meilleurs individus dans la génération suivante.

— Il favorise les solutions non-dominées (en anglais *non-dominated sorting*).

— Il utilise des solutions variées grâce à la distance d'encombrement (en anglais *crowding distance*). Elle correspond à l'espace d'influence autour d'une instance.

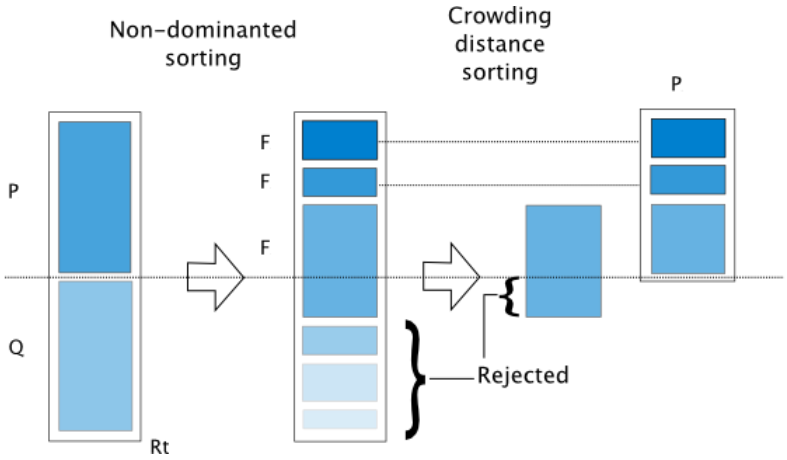

FIGURE 17 – Représentation du fonctionnement de NSGA-II.

A la figure 17, une représentation du fonctionnement de l'algorithme de NSGA-II est exposée : P correspond à l'ensemble de la population d'individus de base (composé de t individus) tandis que Q correspond à la progéniture de P, ensemble de taille t qui a été construit avec des opérations de sélection, reproduction et mutation. Concrètement, une fois que la population P a été entièrement classée par un tri de solutions non-dominées et par la crowding distance (notions qui sont définies dans ce qui suit), deux individus de P sont tirés au sort, pour la création d'un nouvel individu de Q et ceux-ci sont comparés entre eux sur base des deux critères de classement. L'individu avec le meilleur ensemble d'approximation F est retenu. Si ces deux individus partagent la même couche F alors un départage suivant la crowding distance s'effectue. Cette opération est répétée une autre fois afin d'obtenir deux parents pour le prochain enfant et porte le nom de sélection. C'est grâce à ce principe que l'élitisme est garanti. Une fois que les deux parents ont été désignés, ceux-ci fusionnent entre eux et leurs gènes se voient mélangés suivant une fonction définie. Cette opération se nomme la reproduction. Enfin, une fois les gènes de la progéniture construits, une opération de mutation est accomplie suivant une fonction définie. Le but est d'attribuer un ou plusieurs nouveaux gènes propres à l'enfant. L'ensemble Q est construit ainsi de suite jusqu'à ce qu'il soit de taille égale à P et l'union de ces deux ensembles est désigné par la variable R_t. Les deux tris (non-dominated sorting et crowding distance) sont ensuite appliqués à l'ensemble R_t. En résumé, le but de NSGA-II est, à partir d'une population P donnée de t individus, de retourner la prochaine génération P_{t+1} (de t individus) qui est censée être composée des instances les plus intéressantes. Etant donné que la taille de R_t est deux fois supérieure à la taille de P_{t+1}, t individus ne doivent pas être choisis pour la prochaine génération. Pour ce faire, les individus des couches de Pareto

les plus importantes sont retenus dans P_{t+1} jusqu'à ce que cet ensemble soit trop petit pour accueillir tous les individus d'une nouvelle couche de Pareto souhaitant entrer dans cet ensemble. Dans ce cas, le deuxième critère (crowding distance) est appliqué au sein de cette couche de Pareto pour n'accueillir que les individus apportant le plus de diversité.

Non-dominated sorting :
Soit $ND(P)$ les solutions non-dominées dans une population P. Cette méthode classe la population en sous-ensembles (couches) en se basant sur la non-dominance Pareto. Ces ensembles d'approximation R peuvent être définis récursivement comme suit [9]

$$R_1 = ND(P)$$
$$R_{k+1} = ND(P \setminus \cup_{i=1}^{k} R_i), k = 1, 2, ..$$

A chaque étape étape de la récursion, au moins une solution est retirée de la population P. Le rang de ces solutions après avoir été trié avec cette méthode est donné par l'indice k de R_k. A la figure 18, une représentation de trois ensembles d'approximation d'une population est proposée.

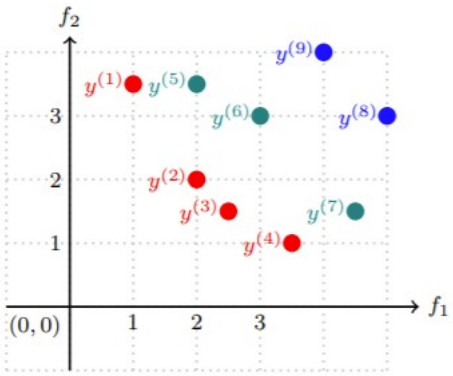

FIGURE 18 – Représentation de 3 couches - représentées en 3 couleurs différentes - d'une population de 9 individus. Source : [9]

Crowding distance :
S'il y a plus d'un individu dans une couche R, un second tri est utilisé pour classer les individus entre eux à l'intérieur de celle-ci. Cette méthode applique le critère de distance d'encombrement. La crowding distance d'une solution $x \in R$ est calculée par une somme des contributions c_i de la ième fonction objective et est définie par [9]

$$l_i(x) := max(\{f_i(y) | y \in R \setminus \{x\} \wedge f_i(y) \leq f_i(x)\} \cup \{-\infty\})$$
$$u_i(x) := min(\{f_i(y) | y \in R \setminus \{x\} \wedge f_i(y) \geq f_i(x)\} \cup \{\infty\})$$
$$c_i(x) := u_i - l_i, i = 1, .., m$$

$$c(x) := \frac{1}{m} \sum_{i=1}^{m} c_i(x), x \in R$$

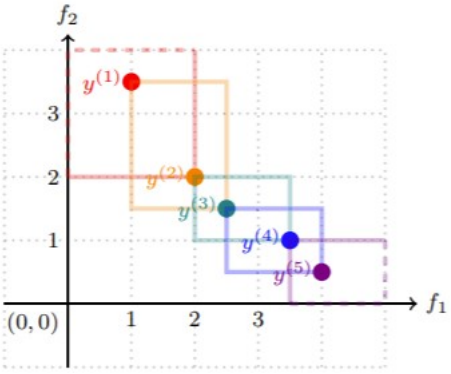

FIGURE 19 – Représentation de la crowding distance (aire de couleur délimitée autour d'une instance de la même couleur) d'une population . Source : [9]

A la figure 19, une représentation de la distance d'encombrement d'une population est présentée.

L'algorithme NSGA-II a une complexité en $O(mN^2)$ où m représente le nombre d'objectifs à optimiser et N la taille de la population [15]. C'est un algorithme qui ne nécessite que très peu d'hyperparamètres et un grand nombre de fonctions objectives ne pose pas de problème à sa bonne réalisation. Par contre il est parfois compliqué de couvrir un espace régulier des solutions et de garantir une convergence.

2.5.7 SMS-EMOA

SMS-EMOA (en anglais *φ metric selection - evolutionary multiobjective optimisation algorithms*) est un algorithme génétique qui se base sur un indicateur qui mesure la performance d'un ensemble [9] [16]. Souvent cet indicateur fait référence à l'hypervolume et est utilisé pour mesurer la qualité d'une couche de front de Pareto. La procédure de sélection de cet algorithme privilégie les individus qui tendent à améliorer cet indicateur.

L'hypervolume HI mesure la taille de l'espace dominé, borné par un point référence r (souvent très grand lors de problème de minimisation).
Pour un ensemble d'approximation $A \subset \mathbb{R}^m$ il est défini comme suit [9]

$$HI(A) = Vol(\{y \in \mathbb{R}^m : y \preceq_{Pareto} r \wedge \exists a \in A : a \preceq_{Pareto} y\})$$

où Vol désigne la mesure de Lebesgue [17].
La contribution d'hypervolume d'un point y pour un ensemble d'approximation Y vaut [9]

$$\Delta HI(y, Y) = HI(Y) - HI(Y \backslash \{y\})$$

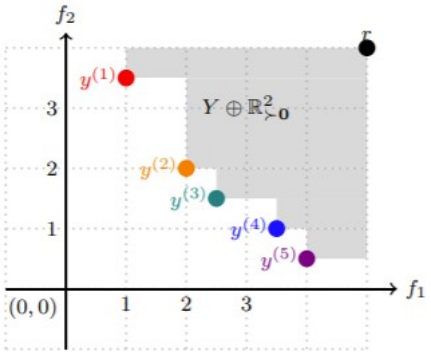

FIGURE 20 – Représentation de l'hypervolume (zone grisée) déterminé par 5 instances $y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}, y^{(5)}$ et un point référence r. Source : [9]

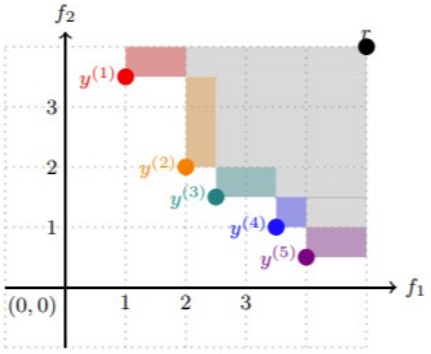

FIGURE 21 – Représentation de la contribution de l'hypervolume de 5 instances $y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}, y^{(5)}$ et un point référence r. Source : [9]

A la figure 20, une représentation graphique de l'hypervolume est montrée tandis qu'à la figure 21, une représentation de la contribution de l'hypervolume pour des instances et un point référence est présentée.

L'algorithme SMS-EMOA cherche à maximiser l'hypervolume d'un ensemble d'instance. Il débute avec une population P d'individus (composée de t éléments) et crée une seule progéniture grâce à des opérations de sélection, reproduction et mutation (notions définies à la section 2.4.6 de ce mémoire). La nouvelle population $P_t^{'}$, composée de P et de la nouvelle progéniture, a donc une taille de $t + 1$ individus après cette étape. Afin de réduire la taille de la population à t individus, SMS-EMOA extrait le sous-ensemble de $P_t^{'}$ de taille t qui maximise l'hypervolume pour donner l'ensemble P_{t+1}. L'algorithme répète ces étapes, en repartant de celle de la sélection, un certain nombre de fois défini par l'utilisateur avant de retourner la nouvelle population P_{t+1}.

C'est un algorithme qui a la même complexité que NSGA-II, c'est-à-dire une complexité en $O(mN^2)$ où m représente le nombre d'objectifs à optimiser et N la taille de la population [16].

Malgré cela, son temps d'exécution augmente rapidement avec le nombre de dimensions du problème étudié et les résultats obtenus varient fortement en fonction du point de référence choisi et des paramètres de l'indicateur.

2.5.8 MOEA/D

MOEA/D (en anglais *Multiobjective Evolutionary Algorithm based on Decomposition*) est un algorithme génétique qui se base sur la décomposition du problème principal en plusieurs sous-problèmes qui se focalisent chacun sur une partie du front de Pareto [9] [15] [18] [19]. Chaque sous-problème défini utilise une paramétrisation différente d'une méthode de scalarization choisie (voir les sections 2.4.2, 2.4.3, 2.4.4 et 2.4.5 de ce mémoire).

C'est un algorithme qui se base principalement sur trois idées :

— La décomposition d'un problème d'optimisation multiobjectif en un ensemble scalaire de problème d'optimisation objective.

— L'utilisation des méthodes de scalarization en se basant sur un vecteur poids.

— La définition des relations de voisinage entre chaque sous-problème en se basant sur les distances entre leurs coefficients de vecteurs respectifs.

A chaque génération, la population étudiée est composée des meilleurs individus trouvés depuis le lancement de l'algorithme pour chaque sous-problème. Une illustration de l'idée générale derrière l'algorithme de MOEA/D est montrée à la figure 22.

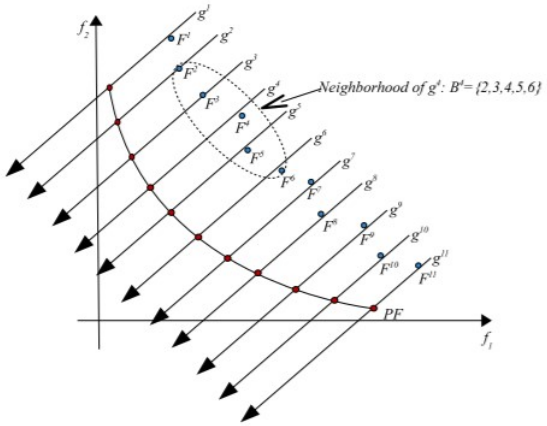

FIGURE 22 – Illustration de l'idée générale derrière MOEA/D. Source : [19]

Généralement c'est la scalarization de Chebychev qui est utilisée. MOEA/D fait évoluer une population P d'individus : chaque individu $x^{(i)} \in P_t$ se voit associé avec un vecteur poids $\lambda^{(i)}$. Le ième sous-problème $g(x|\lambda^{(i)}, z^*)$, z^* étant

un point référence (notion définie à la section 2.5.3 de ce mémoire), est défini comme suit suivant la scalarization de Chebychev [9]

$$g(x|\lambda^{(i)}, z^*) = max_{j\in\{1,\dots,m\}}\{\lambda_j^{(i)} \mid f_j(x) - z_j^* \mid\} + \epsilon \sum_{j=1}^{m}(f_j(x) - z_j^*)$$

Après avoir attribué un poids à chaque sous-problème, le voisinage de ces sous-problèmes est calculé sur la base de la distance euclidienne entre les vecteurs poids liés à ceux-ci. Suivant un paramètre k fixé par les utilisateurs, les k sous-problèmes les plus proches d'un sous-problème i sont rassemblés dans l'ensemble B nommé voisinage de i.

A chaque itération j, le voisinage du sous-problème j est consulté et des opérations de sélection, reproduction ainsi que mutation (notions explicitées à la section 2.4.6 de ce mémoire) sont effectuées afin d'obtenir un individu y. Ensuite, chaque solution x du voisinage de j est comparée à ce nouvel individu y et si y est meilleur que x, la valeur de x est remplacée par y et ainsi de suite jusqu'à ce que l'algorithme renvoie une nouvelle population P_{t+1} en un nombre d'itérations fixé par les utilisateurs.

C'est un algorithme qui a quelques avantages :

— Il convient bien aux problèmes avec un nombre conséquent de fonctions objectives.

— Une importante variété de méthodes de scalarization peut y être incorporée.

— Il a une complexité inférieure à celles de NSGA-II et SMS-EMOA : elle est en $O(mNT)$ où m représente le nombre d'objectifs à optimiser, N la taille de la population et T la taille du voisinage de chaque sous-problème [15]. Comme $T \leq N$, la complexité est donc plus faible que les deux autres algorithmes génétiques présentés en amont de cette section.

— Il produit des solutions de meilleure qualité ou de qualité équivalente à celles retournées par NSGA-II sur la plupart des instances d'un jeu de données de départ [15].

Malgré ces avantages, MOEA/D requiert des connaissances a priori sur la position du front de Pareto dans l'espace objectif et le nombre de vecteurs poids peut grandir exponentiellement en fonction de la taille de l'espace objectif.

3 Simulateur TNT2 et méta-modélisation

Comme indiqué en introduction à ce mémoire, la problématique de cette étude est de trouver comment construire un ensemble d'apprentissage à partir d'un ensemble de scénarios. Le but d'un futur travail serait d'extraire des règles d'apprentissage sur le simulateur TNT2 en partant d'un ensemble d'apprentissage. Les experts du domaine agro-écologique obtiendraient donc un méta-modèle construit sur base de ces règles. Il est donc pertinent de donner une intuition sur le simulateur TNT2.

3.1 Présentation du simulateur TNT2

TNT2 (en anglais *topography-based nitrogen transfer and transformation model*) est un simulateur qui fournit une description "physique" des bassins versants et des processus hydrologiques et agrologiques s'y déroulant [20] [21].
Il a été développé pour étudier les flux d'eau et d'azote dans des petits bassins versants ($< 50km^2$). Son but est de pouvoir être facilement pris en main par ses utilisateurs afin qu'ils acquièrent des connaissances supplémentaires sur les systèmes de bassins versants en simulant des scénarios, composés d'un certain nombre de variables d'entrée et de sortie, qui pourront aider à diminuer la concentration d'azote dans ces bassins.

Un bassin versant se voit modélisé par un tableau de colonnes tri-dimensionnelles, chacune correspondant à un pixel de *Digital Elevation Model* (DEM) du bassin versant. Une DEM est une représentation graphique 3D des données d'altitude pour représenter un terrain. Elle est souvent utilisée dans les systèmes d'information géographique et constitue la base la plus courante pour les cartes en relief produites numériquement [22].

Le simulateur TNT2 est basé sur un calcul quotidien du bilan hydrique et azoté dans chaque cellule (parcelle du bassin versant) de la DEM et un réseau de drainage détermine le sens et la direction des transferts eau-azote entre ces cellules. Les calculs de transfert d'eau sont basés sur quatre hypothèses dérivées d'un autre simulateur : le simulateur Top-model [23] :

— Le gradient hydraulique dans chaque cellule est constant et est contrôlé par la topographie.

— La conductivité hydraulique du substrat diminue exponentiellement avec la profondeur.

— Les flux saturés sont décrits par la loi de Darcy [24] : "La circulation d'un fluide entre deux points est déterminée par la conductivité hydraulique ou le coefficient de perméabilité du substrat et par le gradient de pression du fluide. Dans le cas d'un cours d'eau ou d'un réservoir alimentant une nappe, ce gradient est lié à la hauteur de l'eau."

— L'écoulement terrestre est généré sur les zones saturées.

Chaque colonne de la grille DEM quadrillée est divisée en 2 parties : le sol

et la régolithe (partie du sol recouvrant la roche-mère) et peut être représentée comme montré à la figure 23.

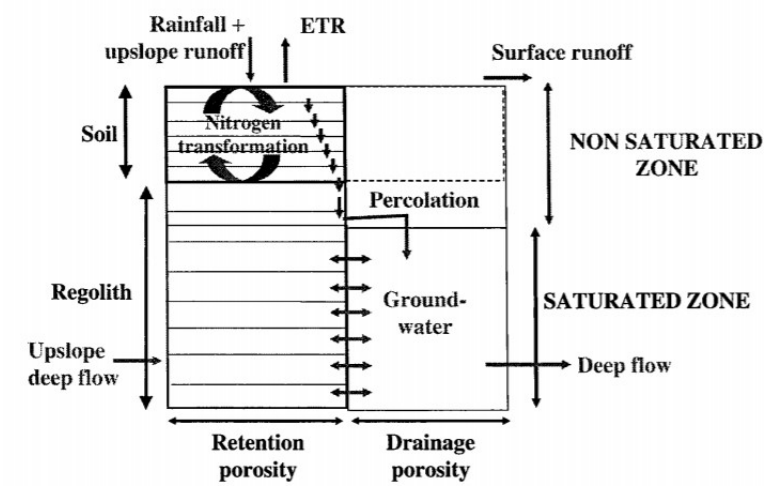

FIGURE 23 – Equilibre hydrique et azoté dans une cellule du DEM. Source : [25]

Trois flux de sorties sont calculés pour chaque cellule de la grille :

1. L'écoulement de surface qui résulte de la saturation du sol et qui est acheminé vers la surface du sol des cellules en dessous de celle étudiée.

2. L'exfiltration qui provient des eaux souterraines en excès dans le sol et qui est aussi acheminée vers la surface des cellules en dessous.

3. L'écoulement souterrain qui provient des eaux souterraines et qui est acheminé vers les eaux souterraines de la cellule en dessous de celle étudiée.

De cette manière, l'écoulement de surface et l'exfiltration se distinguent l'un de l'autre, ce qui est très important pour différencier la qualité des eaux : les eaux de l'écoulement de surface concernent les eaux de pluie tandis que celles de l'exfiltration ont la concentration des eaux souterraines. Il existe des cellules spéciales appelées cellules "rivière" (cellule contenant une portion de la rivière). Les flux de sortie liés à ces cellules sont supposés être drainés par la rivière et acheminés directement depuis la cellule "rivière" à la sortie sans aucune interaction avec d'autres cellules.

Il existe deux types de routage : le monodirectionnel où l'eau se dirige d'une cellule vers son voisin le plus bas et le multidirectionnel où l'eau se déplace d'une cellule vers tous ses voisins les plus bas par rapport à une hauteur définie.
En résumé, le simulateur TNT2 permet de comprendre et de simuler les transferts d'eau et d'azote entre chaque parcelle d'un bassin versant.

3.2 Présentation du jeu de données

Voici représentés à la figure 24, les types de paramètres à fournir en entrée au simulateur TNT2 et les sorties produites.

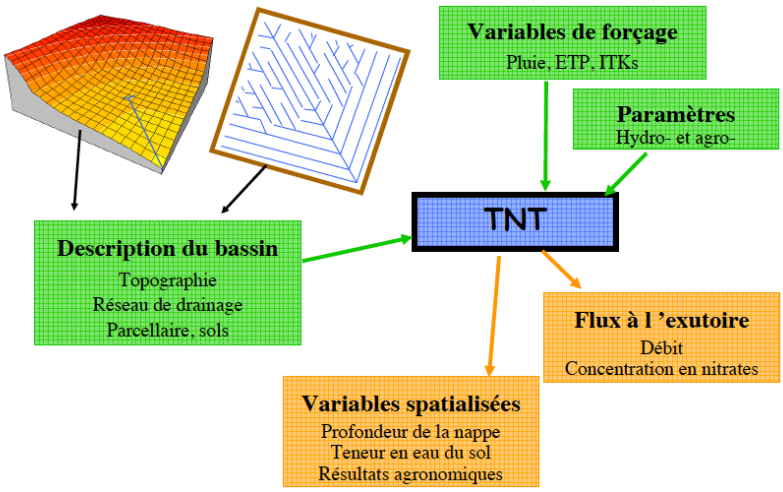

FIGURE 24 – Diagramme de la construction d'un simulateur d'un bassin versant.

On y retrouve en entrée des données climatiques, des données liées à la description du bassin versant ainsi que deux paramètres T et m. La variable m correspond à la profondeur d'eau à laquelle la transmissivité devient x fois plus petite qu'à la surface. T est quant à lui un paramètre contrôlant le taux d'épuisement du réservoir d'eau souterraine. La transmissivité est définie comme la vitesse à laquelle l'eau souterraine s'écoule horizontalement dans un aquifère (terrain perméable). En sortie on y retrouve des données spatialisées, les flux d'eau ainsi que la concentration et les flux d'azote à une étape journalière pour chaque parcelle du bassin versant. Au total, il y a 44 variables de sorties. A titre d'exemple, 16 d'entre elles sont présentées à la figure 25.

Une base de données de 100 simulations (notion définie au point 3.3 de ce mémoire) est disponible. En moyenne, une simulation pour un bassin versant moyen (entre 4000-5000 parcelles) correspond à :

— Taille jeu de données : $\sim$ 300 MO.
— Durée simulée : entre 20 et 30 ans.
— 44 variables/indicateurs de sortie par parcelle.
— Une dizaine de cultures.
— $\sim$ 100 ITK (itinéraire techniques : suite d'opérations culturales).

Variable	Description
GW	Groundwater table (height from the impermeable layer) (m)
WSC	Soil water storage capacity (mm)
N_Soilwater	Nitrogen stored in soil water storage ($kg.ha^{-1}$)
N_Weathered	Nitrogen stored in weathered layer ($kg.ha^{-1}$)
N_GW	Nitrogen stored in groundwater ($kg.ha^{-1}$)
N_Fix	Atmospheric nitrogen fixed by plants ($kg.ha^{-1}$)
N_Mine	Mineral nitrogen resulting from mineralization ($kg.ha^{-1}$)
N_Denit	Denitrified nitrogen ($kg.ha^{-1}$)
N_Sequestre	Nitrogen stored in organic matter ($kg.ha^{-1}$)
N_Plant	Nitrogen fixed by plants ($kg.ha^{-1}$)
N_Mine_Grazing	Amount of mineral N returns from cows during grazing (kg.ha-1)
N_Mine_Manure	Amount of mineral N from manure (kg.ha-1)
N_Volat_Manure	Amount of N from manure to atmosphere by volatilization (kg.ha-1)
N_Volat_Ferti	Amount of N from fertilizers to atmosphere by volatilization (kg.ha-1)
N_Atm	Amount of nitrogen from atmospheric deposition (kg.ha-1)
N_Fertilizer	Amount of N mineral fertilizer (kg.ha-1)

FIGURE 25 – Description de 16 variables de sortie du simulateur TNT2. Source : [20]

3.3 Simulation

Dans ce contexte, une simulation est définie comme suit : des flux de données (concentrations en eau et en nitrates) à un pas de temps journalier, pendant une période de simulation de 30 ans, avec une répartition spécifique de l'utilisation des terres, et dans un espace géographique donné.

4 Contributions

4.1 Submodular pick et optimisation multiobjectif

Pour rappel, l'objectif de cette étude est de trouver comment construire un ensemble d'apprentissage composé de scénarios à partir d'un jeu de données. Cet ensemble devrait être exhaustif et de taille minimale. Après des recherches théoriques dans le domaine, l'idée de combiner la méthode du Submodular pick avec l'optimisation multiobjectif a été retenue : l'objectif de couverture de la méthode du Submodular pick a été développé et deux autres objectifs ont été élaborés. Ces différents objectifs servent de critères permettant de différencier les scénarios appartenant à l'ensemble d'apprentissage ou non.

Le but des trois fonctions objectives décrites dans ce mémoire est de trouver des scénarios qui maximisent/minimisent leurs objectifs respectifs.
La finalité de la méthode du Submodular pick est quant à elle de retourner un sous-ensemble (β) de α ($|\beta| \leq |\alpha|$) composé de scénarios qui sont les plus représentatifs de la population totale (voir à la section 2.3.4 de ce mémoire pour plus d'informations). L'idée du Submodular pick d'attribuer une variable binaire, à chaque scénario, qui représente si le scénario a été choisi pour appartenir à l'ensemble d'apprentissage ou non a été retenue pour la problématique étudiée.
Soit s_i un scénario,

$$x_i = 1 \text{ si } s_i \text{ est choisi pour appartenir à l'ensemble d'apprentissage}$$

$$x_i = 0 \text{ sinon}$$

Cette variable x_i est présente dans chaque fonction objective formalisée dans cette section et permet donc de savoir quand le scénario s_i est choisi pour faire partie du sous-ensemble β représentatif de la population α.

4.2 Modèle proposé :

Dans cette section, la présentation de trois fonctions objectives ainsi que la formalisation de la syntaxe et la sémantique utilisée pour représenter les données de la problématique étudiée vont être introduites.
Soit α l'ensemble des scénarios qui ont déjà été simulés ou non :

$$\begin{array}{c} \\ M \text{ scénarios} \end{array} \begin{array}{c} \overset{N \text{ variables } (N_i + N_o)}{} \\ \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1N} \\ f_{21} & NULL & \cdots & f_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M1} & f_{M2} & \cdots & f_{MN} \end{bmatrix} \end{array}$$

N_i = Ensemble des variables d'entrée (qui peuvent avoir un domaine numérique ou catégoriel).
N_o = Ensemble des variables de sortie (qui ne peuvent qu'avoir un domaine numérique).
f_{ij} = Notation désignant la valeur de la j^{ieme} variable dans le scénario s_i

Un scénario s_i consiste en un ensemble de $N(N_i + N_o)$ valeurs. Parfois des valeurs peuvent ne pas être présentes. Dans ce cas, la valeur spéciale "NULL" sera attribuée à la variable correspondante. Une différence majeure entre N_i et N_o est que toutes les valeurs de N_o peuvent prendre la valeur spéciale "NULL" ce qui signifie qu'aucune variable de sortie n'a été simulée tandis que certaines variables de N_i peuvent être à "NULL" mais pas toutes en même temps. Dans le cas où une valeur de N_i est à "NULL", cela signifie que c'est un choix humain et que la variable correspondante n'a pas été jugée pertinente à renseigner.
Pour illustrer, si N_i = {Type de sol, Climat, Surface cultivée, Ph du sol} et N_o = {Lessivage d'azote, Variable fictive}, un scénario s_i composé de $N(N_i + N_o)$ valeurs pourrait valoir l'ensemble {'Hume', 'EF', NULL, 2, 50, 78}.

4.2.1 Fonction de couverture

Cette fonction cherche à favoriser les scénarios avec des valeurs de variables qui apparaissent souvent dans α pour appartenir à l'ensemble d'apprentissage. L'objectif est de s'assurer que le sous-ensemble représentatif de la population prenne en compte une large variété de valeurs. Pour ce faire, la fonction objective utilisera la notion de TF-IDF [26].

Pour pouvoir comparer deux valeurs d'une même variable de deux scénarios différents, concevoir des classes d'équivalence (intervalles de valeurs) qui seraient fournies par des experts du domaine agro-écologique pour chaque variable est une solution. En effet, il est paru approprié d'utiliser des intervalles de valeurs pour comparer deux données numériques entre elles. Ce choix est motivé par la perte de précision dans le cas de comparaison entre des valeurs numériques "brutes". Il aurait été difficile de retrouver exactement la même valeur d'une variable j dans un scénario s_k, au chiffre décimal près, pour la même variable dans un scénario $s_i (i \neq k)$. A chaque valeur d'une variable j d'un scénario s_i, un intervalle de valeurs $f_{ij}^{'}$ est donc associé. De cette manière, comparer deux valeurs d'une même variable à partir de deux scénarios différents revient à voir si ces deux valeurs appartiennent au même intervalle défini par les experts.
Notation : $f_{ij}^{'}$ = classe d'équivalence de f_{ij}. Si j est une variable catégorielle alors $f_{ij}^{'} = f_{ij}$ et si f_{ij} = "NULL" alors la classe d'équivalence $f_{ij}^{'}$ associée à cette valeur vaut "NULL" aussi.

La définition de la fréquence d'une valeur d'une variable j ($j \in N$) dans un scénario s_i ($Freq_{ij}$) dans une population α de M scénarios se formule comme suit

$$Freq_{ij} = \frac{\sum_{z=1}^{M} 1 \text{ si } f_{zj}^{'} = f_{ij}^{'} \text{ et } z \neq i}{M-1}$$

De plus, la fréquence d'un ensemble T de valeurs de variables ($\in N$) dans un scénario s_i ($Freq_{iT}$) dans une population α de M scénarios se formule comme suit

$$Freq_{iT} = \frac{\sum_{z=1}^{M} 1 \text{ si } \forall j \in T,\ f_{zj}^{'} = f_{ij}^{'} \text{ et } z \neq i}{M-1}$$

Dans les deux définitions ci-dessus les valeurs "NULL" sont considérées comme des valeurs à part entière. Ce choix est motivé par le fait que l'ensemble d'ap-

prentissage recherché doit être le plus représentatif possible de la population de départ. Or si beaucoup de valeurs de variables sont à "NULL" dans α, cette proportionnalité doit être conservée dans l'ensemble β.

En partant de la définition de $Freq_{ij}$, la notion de présence W_{ij} se définit comme suit

$$W_{ij} = 1 \text{ si } Freq_{ij} \neq 0$$

$$W_{ij} = 0 \text{ sinon}$$

En partant de la définition de $Freq_{iT}$, la notion de présence W_{iT} se définit comme suit

$$W_{iT} = 1 \text{ si } Freq_{iT} \neq 0$$

$$W_{iT} = 0 \text{ sinon}$$

Intuitivement, $W_{ij}/W_{iT} = 1$ quand la(les) classe(s) d'équivalence de la variable j (de toutes les variables $j \in T$) du scénario s_i a(ont) déjà été retrouvée(s) dans un scénario s_j $(j \neq i)$ de α.

Finalement, la définition de la fonction de couverture $f_{couverture}$ d'un scénario s_i se formalise comme suit
Soit $I \subset N$, $T = \{T_1, .., T_n |\ T_1, ..T_n \subset N\}$ et $\forall T_k \in T, T_k \cap I = \emptyset$

$$f_{couverture}(s_i) = x_i\ (\sum_{j=1}^{I} \frac{W_{ij}}{log(Freq_{ij} + 2)} + \sum_{k=1}^{n} |T_k| \frac{W_{iT_k}}{log(Freq_{iT_k} + 2)})$$

L'ensemble I est composé de variables qui sont considérées comme indépendantes par les experts du domaine agro-écologique tandis que l'ensemble T est composé de sous-ensembles de variables qui doivent être considérées comme un tout : elles sont dépendantes les unes des autres dans leur sous-ensemble commun.
Des erreurs de domaine pour la fonction logarithme pourraient survenir lorsque $Freq_{ij}/Freq_{iT_k}$ valent 0 ou 1 : $log(0)$ n'appartient pas au domaine de la fonction logarithme et $log(1) = 0$ ce qui provoque une erreur de division par zéro si ce cas n'est pas évité dans les fractions de la fonction de couverture. Ces cas se produisent lorsque $W_{ij}/W_{iT_k} = 0$ ou $W_{ij}/W_{iT_k} = 1$ et $Freq_{ij}/Freq_{iT_k} = 1$. Pour régler ces problèmes, additionner le naturel 2 à la fréquence à l'intérieur des fonctions logarithmes est une solution. Par cette manière, les cas où $log(0)$ et $log(1)$ apparaissent sont toujours évités. Enfin, dans l'optique de rendre plus importants (car plus rares) les sous-ensembles de variables dépendantes de grandes tailles qui ont vu toutes leurs valeurs de variables apparaître au moins une fois dans un autre scénario s_j $(j \neq i)$ de α, le poids $|T_k|$ représentant la cardinalité de ce sous-ensemble k a été multiplié par $\frac{W_{iT_k}}{log(Freq_{iT_k}+2)}$.

Cet objectif sera optimisé pour des valeurs maximales retournées par la fonction de couverture.

4.2.2 Fonction du coût de simulation

Cette fonction permet d'estimer le prix d'une simulation qui est directement lié au nombre de variables de sortie à simuler. Elle tentera de renvoyer les scénarios pour lesquels le nombre de variables en sortie $j \in N_o \neq NULL$ est

élevé.

La notion de présence γ_{ij} d'une variable $j \in N_o$ dans un scénario s_i se formalise comme suit

$$\gamma_{ij} = 1 \text{ si la valeur de la variable j de } s_i = \text{NULL}$$

$$\gamma_{ij} = 0 \text{ sinon}$$

Le temps de calcul pour simuler une variable de sortie est supposé différent pour chaque variable de sortie. Les variables les plus chères en termes de temps de calcul se verront donc attribuer un poids plus important. Ces constantes sont définies par les experts du domaine agro-écologique et sont supposées pouvoir évoluer dans le temps.

La définition du poids du temps de calcul w_j d'une variable $j \in N_o$ s'écrit comme suit

$$w_j = \text{constante}$$

Finalement, la définition de la fonction de coût f_{cout} d'un scénario s_i se formalise comme suit

$$f_{cout}(s_i) = x_i \sum_{j \in N_o} \gamma_{ij} w_j$$

Cette fonction est optimisée lorsqu'elle est minimisée : l'ensemble d'apprentissage préférera choisir des scénarios pour lesquels il y a peu (ou pas) de variables de sortie à simuler pour ces scénarios.

4.2.3 Fonction de distance

Le but de cette fonction est d'atteindre toutes les zones de l'espace de recherche du problème étudié et de n'en oublier aucune afin de s'assurer que chaque zone est couverte par au moins un scénario appartenant à l'ensemble d'apprentissage. Pouvoir comparer deux scénarios différents afin d'étudier leurs similitudes est donc primordial.

Afin de pouvoir calculer la distance entre deux scénarios, la population α est transformée en $\alpha^{'}$:

$$\begin{array}{c} \\ M \text{ scénarios} \end{array} \begin{array}{c} \overset{N^{'} \text{ variables } (N_i + N_o + N_o^{'})}{} \\ \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1N'} \\ f_{21} & NULL & \cdots & f_{2N'} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M1} & f_{M2} & \cdots & f_{MN'} \end{bmatrix} \end{array}$$

où $N_o^{'}$ correspond à de nouvelles variables de sortie ($|N_o^{'}| = |N_o|$) ayant les valeurs "True" ou "False" indiquant si la variable correspondante dans N_o a été simulée ou non, c'est à dire si la valeur de la variable dans N_o vaut "NULL" ou non. Comme les valeurs des variables de N_o sont numériques, si une valeur "NULL" est présente dans une variable de sortie, il y a une perte d'information

dans le calcul de la distance car cette valeur ne sera pas comprise comme "étant simulée" ou non puisque la définition de la distance entre deux scénarios ne prend pas en compte les valeurs "NULL". Ces nouvelles variables $N_o^{'}$ sont donc utiles lors du calcul de la distance entre deux scénarios et pallient cette perte d'information.

Une fois que l'ensemble de données $\alpha^{'}$ a été construit, une normalisation en z [27] des valeurs numériques est appliquée à chaque variable numérique j de $\alpha^{'}$ pour empêcher les entités qui opèrent dans un ordre de grandeur supérieur de dominer.

$$f_{ij} = \frac{f_{ij} - \mu_j}{\sigma_j} \ \forall j \text{ variable numérique de } \alpha^{'}$$

La définition de la distance entre deux scénarios s_i et s_k, Distance (s_i, s_k), se formalise comme suit
Distance (s_i, s_k)

$$\forall j \in N^{'} \mid j \text{ variable numérique }, \ f_{ij} \neq NULL \wedge f_{kj} \neq NULL :$$

$$= \frac{\#j}{|N' \backslash \{v \in N' | f_{iv} = NULL \vee f_{kv} = NULL\}|} \sqrt{\sum_j (f_{ij} - f_{kj})^2}$$

$$\forall z \in N^{'} \mid z \text{ variable catégorielle }, \ f_{iz} \neq NULL \wedge f_{kz} \neq NULL :$$

$$= \frac{\#z}{|N' \backslash \{v \in N' | f_{iv} = NULL \vee f_{kv} = NULL\}|} \sum_z HamDist(f_{iz}, f_{kz})$$

où $HamDist(x, y)$ fait référence à la distance de Hamming [28].

Cette distance définie-ci dessus sera utilisée pour trouver les différents clusters du jeu de données $\alpha^{'}$. L'intuition derrière l'utilisation de méthodes de clustering pour cette fonction objective est simple : l'objectif de cette fonction est optimisé lorsque chaque zone de l'espace de recherche est atteinte par au moins un scénario appartenant à β. Pour pouvoir délimiter ces zones, les méthodes de clustering sont pertinentes. Pour rappel, un cluster est une collection de données où celles-ci présentent des similitudes entre elles. La tâche du clustering est d'identifier k clusters dont les données à l'intérieur de ceux-ci ont des similitudes les unes avec les autres et dont les données de différents clusters sont dissemblables entre elles. La tâche du clustering équitable [29] est la même que celle du clustering à la différence que les k clusters doivent être de taille équilibrée, ce qui est intéressant dans la problématique pour s'assurer au mieux que le plus de zones différentes de l'espace de recherche soient représentées par des scénarios.

Plus formellement, soit M le nombre de scénarios dans le jeu de données α, N l'ensemble des centroïdes possibles ($|N| = n$) et k le nombre de clusters qu'on souhaite obtenir. La tâche du clustering proportionnellement équitable est de trouver un ensemble de centres de cluster (centroïdes) X ($|X| = k$) tel que $\forall i \in X \Rightarrow i \in N$.
Une solution X est proportionnellement juste s'il n'existe pas de groupe de points $S \subseteq \alpha$ avec $|S| \geq \frac{M}{k}$ tel que chacun d'entre eux préfère strictement un

point $y \notin X$ à leur plus proche centroïde dans X.

Dans les exemples décrits ci-dessous : k=2, M = {1,2,3,4,5,6} (les cercles bleus) et N ={1,2,3,4} (les rectangles).
Lorsque la couleur d'un rectangle est orange, cela signifie que ce centroïde $\in X$.

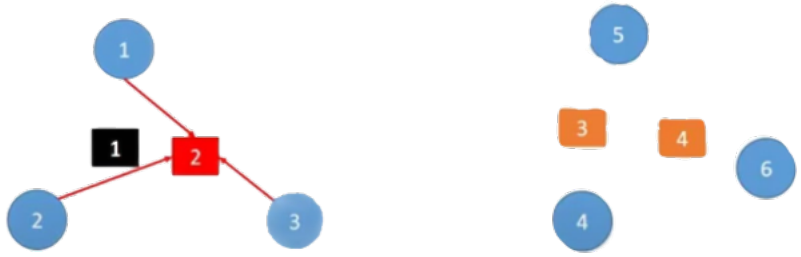

FIGURE 26 – Exemple de clustering proportionnellement équitable qui ne fonctionne pas

Dans la figure 26, le clustering proportionnellement équitable ne fonctionne pas car il existe un sous-ensemble S={1,2,3} de trois éléments tel que tous ces éléments sont plus proches d'un centroïde ($2 \notin X$) que de leur plus proche centroïde dans X (3).

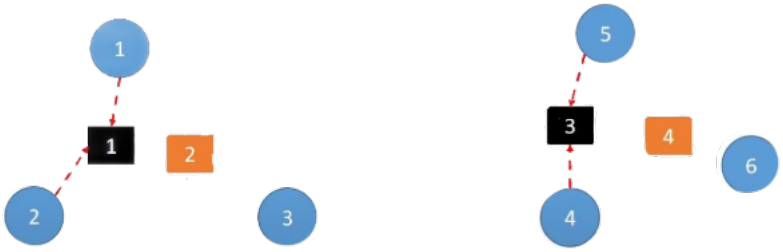

FIGURE 27 – Exemple de clustering proportionnellement équitable qui fonctionne

La solution de la figure 27 fonctionne car il n'existe pas de sous-ensemble d'au moins trois éléments tel que tous les éléments de ce sous-ensemble sont plus proches du même centroïde qui ne fait pas partie de l'ensemble X que de leur plus proche centroïde dans X.

Mathématiquement, la tâche du clustering équitable peut s'écrire comme suit [29]

$$\text{Minimiser} \sum_{i,j \in \alpha'} distance(s_i, s_j)\ z_{ij} \tag{1}$$

avec les contraintes (2)

$$\sum_{j} z_{ij} = 1 \tag{3}$$

$$z_{ij} \leq y_j \tag{4}$$

$$\sum_{j} y_j \leq k \tag{5}$$

$$\sum_{j' \in B(j,\Gamma R_j)} y_{j'} \geq 1 \tag{6}$$

$$z_{ij}, y_j \in \{0, 1\} \tag{7}$$

où :
- z_{ij} =1 quand le scénario s_i est associé au centroïde j, z_{ij} =0 dans le cas contraire.
- y_j=1 quand le centroïde $j \in X$ et 0 sinon.
- k symbolise le nombre de clusters souhaité.
- $B(s_i, \Gamma) = \{s_j \in \alpha^{'} : distance(s_i, s_j) \leq \Gamma\}$.
- R_j est la valeur minimale telle que $|B(j, R_j)| \geq \frac{n}{k}$. En d'autres termes, R_j est la distance des $\frac{M}{k}$ scénarios les plus loin depuis j dans $\alpha^{'}$.

La contrainte (3) nécessite que chaque scénario soit mis en correspondance avec un centroïde, la contrainte (4) permet uniquement à un scénario d'être assorti à un centre ouvert. La contrainte (5) permet d'ouvrir au plus k centroïdes et la contrainte (7) assouplit les variables d'indicateur à des valeurs réelles comprises entre 0 et 1. La contrainte (6) encode la proportionnalité.

Après avoir utilisé la méthode décrite ci-dessus, k clusters différents ont été trouvés. Dans le cadre de la problématique traitée, un nombre k supposé grand a été choisi afin d'obtenir beaucoup de clusters différents et de ne pas avoir à utiliser la notion de prototype [30].

Soit K l'ensemble des k centroïdes trouvés.
La définition de la distance d'un scénario s_i, $f_{distance}(s_i)$, se formule comme suit :

$$f_{distance}(s_i) = x_i\ (min(distance(s_i, k_i)))\ \forall k_i \in K$$

L'ensemble d'apprentissage préférera choisir des scénarios les plus proches des centroïdes déterminés par le clustering équitable [29]. En effet, ces scénarios sont les plus représentatifs des classes/zones de la population α.

L'objectif de cette fonction est optimisé lorsqu'il est minimisé.

5 Expérimentations et résultats

Dans cette section, des expérimentations encodées en langage Python 3.7 sur des données fictives sont exposées. En effet, ces données sont fictives car leur pré-traitement en vue de leur utilisation aurait demandé un travail conséquent, ce qui n'était pas réaliste au vu du temps imparti. Les données représentées dans les tables de cette section ont donc été fabriquées et ne reflètent pas la réalité de la problématique. Le code Python lié à ces expérimentations est disponible à l'adresse suivante : *https://github.com/SimonMattens/Memoire.git*.
Dans la table 1, un jeu de données fictif est proposé. Il contient 25 scénarios différents ayant chacun 6 variables : 4 d'entrées (Type de sol, Climat, Surface cultivée, Ph du sol) et 2 de sorties (Lessivage d'azote, Variable fictive). Parmi ces variables, 2 d'entre elles ont un domaine catégoriel (Type de sol et Climat) et les 4 autres (Surface cultivée, Ph du sol, Lessivage d'azote et Variable fictive) ont un domaine numérique. Des valeurs "NULL" ont été injectées dans certains scénarios et le chiffre arbitraire de 10% du nombre total de scénarios a été choisi pour représenter le nombre de scénarios n'ayant pas encore été simulé.

TABLE 1 – Exemple de jeu de données fictif

Scénario	Type de sol	Climat	Surface cultivée	Ph du sol	Lessivage d'azote	Variable fictive
0	Tifton	Am	NULL	13	80	81
1	Baynes	Dfd	97	7	95	81
2	Tifton	Cfb	47	7	65	NULL
3	Baynes	BSh	75	3	10	3
4	Tifton	Af	17	3	5	81
5	Tifton	Af	57	0	20	-1
6	Baynes	BWk	63	7	NULL	-56
7	Jory	ET	100	12	10	-1
8	Miami	Csc	48	9	90	-1
9	Miami	BWk	20	8	5	-56
10	Jory	EF	89	2	NULL	NULL
11	Antigo	Dfd	60	5	65	-1
12	Baynes	BWk	99	6	90	-1
13	Miami	NULL	58	11	50	3
14	Hume	ET	13	10	NULL	NULL
15	Baynes	EF	84	4	55	3
16	Miami	Af	5	4	20	-56
17	Jory	Af	6	14	40	81
18	Seitz	Csc	41	5	25	81
19	Tifton	Cfb	41	9	55	-1
20	Hume	BWk	88	11	50	0
21	Miami	BWk	36	7	70	78
22	Hume	BSh	97	3	85	81
23	Baynes	EF	26	1	30	3
24	Seitz	Am	51	8	5	0

Les fonctions objectives de couverture et de coût ont été implémentées sui-

vant les définitions proposées aux points 4.2.1 et 4.2.2 de ce mémoire. Concernant la fonction de couverture, des classes d'équivalence aléatoires ont été conçues pour les variables numériques. Dans la pratique, celles-ci sont supposées être fournies par les experts du domaine.
Afin de concevoir la fonction de distance définie au point 4.2.3 de ce mémoire, une normalisation en z [27] a été réalisée sur les valeurs des variables numériques de la table 1. Le résulat obtenu est représenté dans la table 2.

TABLE 2 – Exemple de jeu de données fictif après une normalisation en z

Scénario	Type de sol	Climat	Surface cultivée	Ph du sol	Lessivage d'azote	Variable fictive
0	Tifton	Am	NULL	1.646	1.085	1.318
1	Baynes	Dfd	1.356	0.063	1.569	1.318
2	Tifton	Cfb	-0.255	0.063	0.601	NULL
3	Baynes	BSh	0.647	-0.992	-1.173	-0.320
4	Tifton	Af	-1.222	-0.992	-1.335	1.318
5	Tifton	Af	0.067	-1.784	-0.850	-0.405
6	Baynes	BWk	0.260	0.063	NULL	-1.560
7	Jory	ET	1.453	1.382	-1.173	-0.405
8	Miami	Csc	-0.222	0.591	1.408	-0.405
9	Miami	BWk	-1.125	0.327	-1.335	-1.560
10	Jory	EF	1.098	-1.256	NULL	NULL
11	Antigo	Dfd	0.163	-0.464	0.601	-0.405
12	Baynes	BWk	1.421	-0.200	1.408	-0.405
13	Miami	NULL	0.099	1.119	0.117	-0.320
14	Hume	ET	-1.351	0.855	NULL	NULL
15	Baynes	EF	0.937	-0.728	0.278	-0.320
16	Miami	Af	-1.609	-0.728	-0.850	-1.560
17	Jory	Af	-1.576	1.910	-0.205	1.318
18	Seitz	Csc	-0.448	-0.464	-0.689	1.318
19	Tifton	Cfb	-0.448	0.591	0.278	-0.405
20	Hume	BWk	1.066	1.119	0.117	-0.383
21	Miami	BWk	-0.609	0.063	0.762	1.255
22	Hume	BSh	1.356	-0.992	1.247	1.318
23	Baynes	EF	-0.932	-1.520	-0.528	-0.320
24	Seitz	Am	-0.126	0.327	-1.335	-0.383

Les deux nouvelles variables de sortie, représentant la présence/absence des variables de sorties déjà existantes, ont été rajoutées dans la table 3 afin de pouvoir calculer la distance entre 2 scénarios différents conformément à la définition de distance mentionnée à la section 4.2.3 de ce mémoire.

Table 3 – Transformation du jeu de données

Scénario	Type de sol	Climat	Surface cultivée	Ph du sol	Lessivage d'azote	Variable fictive	Présence lessivage d'azote	Présence variable fictive
0	Tifton	Am	NULL	1.646	1.085	1.318	True	True
1	Baynes	Dfd	1.356	0.063	1.569	1.318	True	True
2	Tifton	Cfb	-0.255	0.063	0.601	NULL	True	False
3	Baynes	BSh	0.647	-0.992	-1.173	-0.320	True	True
4	Tifton	Af	-1.222	-0.992	-1.335	1.318	True	True
5	Tifton	Af	0.067	-1.784	-0.850	-0.405	True	True
6	Baynes	BWk	0.260	0.063	NULL	-1.560	False	True
7	Jory	ET	1.453	1.382	-1.173	-0.405	True	True
8	Miami	Csc	-0.222	0.591	1.408	-0.405	True	True
9	Miami	BWk	-1.125	0.327	-1.335	-1.560	True	True
10	Jory	EF	1.098	-1.256	NULL	NULL	False	False
11	Antigo	Dfd	0.163	-0.464	0.601	-0.405	True	True
12	Baynes	BWk	1.421	-0.200	1.408	-0.405	True	True
13	Miami	NULL	0.099	1.119	0.117	-0.320	True	True
14	Hume	ET	-1.351	0.855	NULL	NULL	False	False
15	Baynes	EF	0.937	-0.728	0.278	-0.320	True	True
16	Miami	Af	-1.609	-0.728	-0.850	-1.560	True	True
17	Jory	Af	-1.576	1.910	-0.205	1.318	True	True
18	Seitz	Csc	-0.448	-0.464	-0.689	1.318	True	True
19	Tifton	Cfb	-0.448	0.591	0.278	-0.405	True	True
20	Hume	BWk	1.066	1.119	0.117	-0.383	True	True
21	Miami	BWk	-0.609	0.063	0.762	1.255	True	True
22	Hume	BSh	1.356	-0.992	1.247	1.318	True	True
23	Baynes	EF	-0.932	-1.520	-0.528	-0.320	True	True
24	Seitz	Am	-0.126	0.327	-1.335	-0.383	True	True

Une fois la table 3 créée, un clustering équitable pour un nombre de centroïdes égal à 6 (chiffre représentant $\frac{1}{4}$ de la taille du jeu de données de base) a été effectué et un ensemble de centroïdes vérifiant la condition du clustering proportionnellement équitable a été trouvé. Il est composé des scénarios {12, 13, 16, 21, 22, 23}.

Une fois les trois fonctions objectives implémentées, la recherche d'un ensemble d'apprentissage minimal s'est opéré. Pour commencer, un tuple composé de trois éléments a été attribué à chaque scénario. Par exemple pour le scénario 0, le tuple (0, 6.621, 1.700) a été retourné. Le premier élément de ce tuple correspond au score de la fonction de coût, le deuxième au score de la fonction de couverture et le troisième au score de la fonction de distance. Un récapitulatif des scores des trois fonctions objectives pour chaque scénario est disponible à la table 4.

TABLE 4 – Récapitulatif des scores des trois fonctions objectives pour chaque scénario

Scénario	Score
0	(0, 6.621, 1.700)
1	(0, 7.656, 1.375)
2	(1, 7.756, 1.722)
3	(0, 7.603, 1.393)
4	(0, 7.646, 1.961)
5	(0, 7.560, 1.543)
6	(1, 7.876, 1.447)
7	(0, 7.742, 1.517)
8	(0, 7.625, 0.716)
9	(0, 7.777, 1.129)
10	(2, 8.045, 2.245)
11	(0, 6.341, 1.401)
12	(0, 7.426, 0.0)
13	(0, 6.164, 0.0)
14	(2, 8.108, 2.697)
15	(0, 7.4355, 1.092)
16	(0, 7.775, 0.0)
17	(0, 6.611, 1.819)
18	(0, 7.863, 1.777)
19	(0, 7.565, 0.962)
20	(0, 7.527, 1.055)
21	(0, 7.648, 0.0)
22	(0, 7.756, 0.0)
23	(0, 7.823, 0.0)
24	(0, 7.613, 1.406)

Une fonction permettant de comparer ces scores entre eux a été ensuite implémentée en se basant sur l'idée de l'ordre Pareto. Afin de constituer les meilleurs ensembles de scénarios de tailles variables sur base des trois fonctions objectives, une fonction retournant les meilleurs ensembles de scénarios de taille 1 jusque 25 (nombre total de scénarios dans la base de données fictive) a été implémentée. Les résultats obtenus sont repris à la table 5.

TABLE 5 – Récapitulatif des meilleurs ensembles de scénarios de taille 1 à 25 avec leurs scores correspondants

Taille de l'ensemble	Meilleur ensemble de scénarios	Score de l'ensemble
1	[23]	(0, 7.823, 0.0)
2	[23, 9]	(0, 15.601, 1.129)
3	[23, 9, 6]	(1, 23.477, 2.577)
4	[23, 16, 9, 6]	(1, 31.253, 2.577)
5	[23, 22, 16, 9, 6]	(1, 39.009, 2.577)
6	[23, 22, 21, 16, 9, 6]	(1, 46.658, 2.577)
7	[23, 22, 21, 16, 9, 8, 6]	(1, 54.284, 3.294)
8	[23, 22, 21, 18, 16, 12, 9, 6]	(1, 61.949, 4.354)
9	[23, 22, 21, 16, 12, 9, 8, 6, 1]	(1, 69.367, 4.669)
10	[23, 22, 21, 19, 18, 16, 12, 9, 8, 6]	(1, 77.140, 6.034)
11	[23, 22, 21, 19, 18, 16, 9, 8, 7, 6, 1]	(1, 85.112, 8.927)
12	[23, 22, 21, 18, 16, 9, 8, 7, 6, 3, 2, 1]	(2, 92.906, 11.080)
13	[23, 22, 21, 18, 16, 12, 9, 8, 7, 6, 3, 2, 1]	(2, 100.333, 11.080)
14	[23, 22, 21, 19, 18, 16, 9, 8, 7, 6, 4, 3, 2, 1]	(2, 108.118, 14.004)
15	[24, 23, 22, 21, 19, 18, 16, 14, 9, 8, 7, 6, 3, 2, 1]	(4, 116.194, 16.146)
16	[23, 22, 21, 20, 19, 18, 16, 12, 10, 9, 8, 7, 6, 3, 2, 1]	(4, 123.472, 15.344)
17	[24, 23, 22, 21, 19, 18, 16, 12, 10, 9, 8, 7, 6, 4, 3, 2, 1]	(4, 131.204, 17.657)
18	[24, 23, 22, 21, 20, 19, 18, 16, 12, 10, 9, 8, 7, 6, 4, 3, 2, 1]	(4, 138.731, 18.713)
19	[24, 23, 22, 21, 20, 19, 18, 16, 15, 12, 10, 9, 8, 7, 6, 4, 3, 2, 1]	(4, 146.167, 19.805)
20	[24, 23, 22, 21, 20, 19, 18, 16, 15, 12, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]	(4, 153.727, 21.348)
21	[24, 23, 22, 21, 20, 19, 18, 16, 15, 13, 12, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]	(4, 159.892, 21.348)
22	[24, 23, 22, 21, 20, 19, 18, 16, 15, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]	(4, 166.691, 24.450)
23	[24, 23, 22, 21, 20, 19, 18, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]	(6, 174.342, 25.447)
24	[24, 23, 22, 21, 20, 19, 18, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]	(6, 180.964, 27.147)
25	[24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]	(6, 187.575, 28.966)

Enfin, pour travailler avec un ensemble minimal représentatif du jeu de données de départ, l'utilisateur devrait donner des contraintes sur la taille minimale que cet ensemble représentatif devrait avoir. Si l'on suppose que cette taille doit valoir au moins 25% de la taille de la base de données originale (i.e une taille de 5 minimum), l'ensemble retourné serait l'ensemble {6, 9, 16, 21,

22, 23} composé de 6 éléments. En effet, entre l'ensemble {6, 9, 16, 22, 23} de score (1, 39.009, 2.577) et l'ensemble {6, 9, 16, 21, 22, 23} de score (1, 46.658, 2.577), l'ensemble {6, 9, 16, 21, 22, 23} est meilleur car il a les mêmes résultats pour les fonctions de coût et de distance mais a un meilleur résultat pour la fonction de couverture puisque cet objectif est optimal lorsqu'il est maximisé.

6 Conclusion

Le contexte de cette étude s'inscrit dans le cadre de la problématique des algues vertes qui prolifèrent depuis des dizaines d'années en Bretagne. Ces plantes sont toxiques à forte concentration pour l'être humain et sont directement liées à l'élevage et la culture agricole intensive dans cette région. Ces deux activités relâchent du nitrate dans les sols et les nappes phréatiques, c'est ce composant qui est responsable de l'apparition des algues vertes. Le simulateur TNT2 permet d'aider les scientifiques à étudier des combinaisons de facteurs pouvant influencer la concentration d'azote dans les sols. Le principal désavantage de ce simulateur est le temps de calcul avoisinant l'heure pour simuler un scénario (ensemble de variables collecté sur une parcelle de terrain). L'objectif de cette étude est de fournir aux scientifiques un méta-modèle sur le simulateur TNT2. Ce méta-modèle serait construit à partir de règles d'apprentissage et permettrait de réduire drastiquement le temps de calcul d'une nouvelle simulation. Pour déduire ces règles, un ensemble de simulations minimal couvrant le plus d'espace possible de l'ensemble de simulations total doit être construit.

6.1 Bilan

Une importante partie sur l'état de l'art de la problématique (apprentissage automatique, interprétabilité, approches "post-hoc", optimisation mutliobjectif) a été exposée en début de mémoire. Cette partie introduit certaines notions qui ont été utilisées dans l'élaboration de trois fonctions objectives mais elle pourrait aussi servir de recueil théorique pour de futures études qui approfondiraient le sujet abordé. Les objectifs fixés au début de l'étude ont été atteints : trois fonctions objectives ont été élaborées et implémentées. Ces fonctions servent à construire un ensemble d'apprentissage de taille minimale qui est représentatif de l'espace de recherche. Une preuve de concept a été développée sur des données fictives à la section 5 de ce mémoire.

Difficultés rencontrées : La durée prévue pour cette étude exploratoire était assez réduite (4 mois dans un laboratoire qui propose habituellement des stages s'étendant sur 5 mois minimum) et, de ce fait, certaines pistes de réflexion n'ont pas pu mener à des résultats concrets. De plus, la crise sanitaire de la COVID-19 n'aidant pas, une importante partie de l'investigation s'est déroulée en distanciel, ce qui n'a pas été avantageux pour travailler avec les experts du domaine agro-écologique.

6.2 Perspectives

L'implémentation des fonctions objectives définies au point 4.2 de ce mémoire s'est effectuée sur des données fictives ne reflétant pas la réalité de la problématique. Ce choix est motivé par la durée restreinte de cette étude exploratoire. La faisabilité des différentes contributions s'est donc arrêtée à une preuve de concept. Une piste à explorer pour un travail futur est d'optimiser le code disponible à l'adresse *https://github.com/SimonMattens/Memoire.git* afin de le rendre plus général et qu'il puisse, par après, s'adapter aux données réelles collectées par les experts du domaine agro-écologique. Les fonctions objectives devront sûrement elles aussi être perfectionnées une fois les expérimentations sur les

données réelles opérées mais aussi après les échanges avec les experts du domaine de l'agro-écologie. Une fois la méthode retournant un sous-ensemble d'instances représentatif de la population de départ peaufinée, il serait pertinent d'apprendre des règles d'apprentissage, à partir de cet ensemble, pour ensuite créer un méta-modèle sur le simulateur TNT2. Il a été supposé que l'interprétabilité se ferait au moyen de règles d'apprentissage mais il se pourrait que d'autres méthodes telles que les arbres de décision, les fonctions linéaires... soient plus pertinentes. C'est pourquoi il serait intéressant d'essayer plusieurs méthodes d'interprétabilité. Deux approches d'interprétabilité pourraient être envisagées selon la volonté des experts du domaine agro-écologique :

— Une approche globale en utilisant des méthodes intrinsèques d'interprétabilité telles que des règles d'apprentissage ou des arbres de décision. Ces méthodes permettraient aux humains de comprendre le plus possible le fonctionnement intégral du simulateur TNT2.

— Une approche locale en utilisant des méthodes "post-hoc" telles que LIME ou Anchors par exemple. Cette approche pourrait être envisagée si les scientifiques souhaitent avoir plus de renseignements sur un scénario précis.

Bibliographie

[1] Wikipedia. Data mining. *https://en.wikipedia.org/wiki/Data_mining*.

[2] Christoph Molnar. Interpretable machine learning. *https://christophm.github.io/interpretable-ml-book/index.html*, 2021.

[3] Goeffrey I.Webb Petra Kralj Novak, Nada Lavrac. Supervised descriptive rule discovery : A unifying survey of contrast set, emerging pattern and subgroup mining. *https://www.researchgate.net/publication/220320802_Supervised_Descriptive_Rule_Discovery_A_Unifying_Survey_of_Contrast_Set_Emerging_Pattern_and_Subgroup_Mining*, 2009.

[4] Wikipedia. Règle d'association. *https://fr.wikipedia.org/wiki/R%C3%A8gle_d%27association*.

[5] Carlos Guestrin Marco Tulio Ribeiro, Sameer Singh. "why should i trust you ?" explaining the predictions of any classifier. *https://www.researchgate.net/publication/305342147_Why_Should_I_Trust_You_Explaining_the_Predictions_of_Any_Classifier*, 2016.

[6] Carlos Guestrin Marco Tulio Ribeiro, Sameer Singh. Anchors : High-precision model-agnostic explanations. *https://homes.cs.washington.edu/~marcotcr/aaai18.pdf*, 2018.

[7] C. Zhang S. Shamasunder M. Burnett W.-K. Wong S. Stumpf S. Das A. Shinsel F. Bice K. McIntosh A. Groce, T. Kulesza. You are the only possible oracle : Effective test selection for end users of interactive machine learning systems. *https://agroce.github.io/tse14.pdf*, 2014.

[8] D. Golovin A. Krause. Submodular function maximization. *https://viterbi-web.usc.edu/~shanghua/teaching/Fall2019-670/krause12survey.pdf*, 2014.

[9] André H. Deutz Michael T. M. Emmerich. A tutorial on multiobjective optimization : fundamentals and evolutionary methods. *https://link.springer.com/article/10.1007/s11047-018-9685-y*, 2018.

[10] Joshua Knowles. Multiobjective optimization - concepts, algorithms and performance measures. *http://syllabus.cs.manchester.ac.uk/pgt/2018/COMP60342/COMP60342-2014-MOO.pdf*, 2014.

[11] Andrzej P. Wierzbicki Kaisa Miettinen, Francisco Ruiz. Introduction to multiobjective optimization : Interactive approaches. *https://www.researchgate.net/publication/221024595_Introduction_to_Multiobjective_Optimization_Interactive_Approaches*, 2008.

[12] J. E. Dennis Indraneel Das. Normal-boundary intersection : A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *https://www.researchgate.net/publication/2460803_Normal-Boundary_Intersection_A_New_Method_for_Generating_the_Pareto_Surface_in_Nonlinear_Multicriteria_Optimization_Problems*, 2000.

[13] Fatima-Zahra Oujebbour. Methods and industrial applications in multicriteria optimization of process parameters in sheet metal forming. *https://www.researchgate.net/publication/278645002_Methods_and_industrial_applications_in_multicriteria_optimization_of_process_parameters_in_sheet_metal_forming*, 2014.

[14] Agarwal S Meyarivan T Deb K, Pratap A. A fast and elitist multiobjective genetic algorithm : Nsga-ıı. *https://www.iitk.ac.in/kangal/Deb_NSGA-II.pdf*, 2002.

[15] Qingfu Zhang Hui Li. Moea/d : A multiobjective evolutionary algorithm based on decomposition. *https://www.researchgate.net/publication/3418989_MOEAD_A_Multiobjective_Evolutionary_Algorithm_Based_on_Decomposition*, 2008.

[16] Michael Emmerich Nicola Beume, Boris Naujoks. Sms-emoa : Multiobjective selection based on dominated hypervolume. *https://www.sciencedirect.com/science/article/abs/pii/S0377221706005443*, 2006.

[17] Wikipedia. Mesure de lebesgue. *https://fr.wikipedia.org/wiki/Mesure_de_Lebesgue#:~:text=Th%C3%A9or%C3%A8me%20et%20d%C3%A9finition%20%E2%80%94%20Il%20existe,d%C3%A9finition%20la%20tribu%20de%20Lebesgue.*

[18] Krzysztof Michalak. The effects of asymmetric neighborhood assignment in the moea/d algorithm. *https://www.sciencedirect.com/science/article/abs/pii/S1568494614004463*, 2014.

[19] Guixu Zhang Aimin Zhou, Qingfu Zhang. A multiobjective evolutionary algorithm based on decomposition and probability model. *https://www.researchgate.net/publication/261165962_A_multiobjective_evolutionary_algorithm_based_on_decomposition_and_probability_model*, 2012.

[20] Pierre Moreau Rene Quiniou Jordy Salmon-Monviola Chantal Gascuel-Odoux Tassadit Bouadi, Marie-Odile Cordier. A data warehouse to explore multidimensional simulated data from a spatially distributed agro-hydrological model to improve catchment nitrogen management. *https://www.sciencedirect.com/science/article/pii/S1364815216305655*, 2017.

[21] Ripoche D. Jeuffroy M. Ruget F. Nicoullaud B. Gate P.-Devienne-Barret F. Antonioletti R. Durr C. Richard G. Beaudoin N. Recous S. Tayot X. Plenet D. Cellier P. Machet J. Meynard J. Delecolle R. Brisson N., Mary B. Stics : a generic model for the simulation of crops and their water and nitrogen balances. 1. theory and parameterization applied to wheat and corn. *https://www.agronomy-journal.org/articles/agro/abs/1998/05/Agronomie_0249-5627_1998_18_5-6_ART0001/Agronomie_0249-5627_1998_18_5-6_ART0001.html*, 1998.

[22] Wikipedia. Digital elevation model. *https://en.wikipedia.org/wiki/Digital_elevation_model*.

[23] Keith Beven. Topmodel : A critique. *https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1099-1085(199707)11:9%3C1069::AID-HYP545%3E3.0.CO;2-O*, 1997.

[24] Wikipedia. Loi de darcy. *https://fr.wikipedia.org/wiki/Loi_de_Darcy*.

[25] Laurent Ruiz Pierre Aurousseau Gilles Cotteret Veronique Beaujouan, Patrick Durand. A hydrological model dedicated to topography-based simulation of nitrogen transfer and transformation : rationale and application to the geomorphology–denitrification relationship. *https://www.oieau.org/eaudoc/system/files/documents/34/174615/174615_doc.pdf*, 2002.

[26] Wikipedia. tf-idf. *https://en.wikipedia.org/wiki/Tf%E2%80%93idf*.

[27] SAX-VSM. Z-normalization. *https://jmotif.github.io/sax-vsm_site/morea/algorithm/znorm.html*.

[28] Wikipedia. Distance de hamming. *https://fr.wikipedia.org/wiki/Distance_de_Hamming*.

[29] Liang Lyu Kamesh Munagala Xingyu Chen, Brandon Fain. Proportionally fair clustering. *https://arxiv.org/abs/1905.03674*, 2019.

[30] Robert Tibshirani Jacob Bien. Prototype selection for interpretable classification. *https://arxiv.org/abs/1202.5933*, 2012.