UNIVERSITÉ DU QUÉBEC À MONTRÉAL

APPROCHE D'IDENTIFICATION DU NOMBRE OPTIMAL DE CLUSTERS DANS LES ALGORITHMES DE REGROUPEMENT EN COMBINANT K-MEANS ET DES MÉTAHEURISTIQUES

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

CHATOU MOHAMMED REDA

MAI 2024

REMERCIEMENTS

Je tiens à exprimer ma plus profonde gratitude envers ma mère, Labtere Fatima, pilier de ma vie et source inépuisable de soutien et d'amour. Tout au long de ce parcours académique, tu as été mon refuge et ma force, m'encourageant à poursuivre mes rêves et à surmonter les obstacles. Ta confiance inébranlable en mes capacités m'a constamment motivé à me surpasser. Ce mémoire est aussi le tien, car sans ton soutien indéfectible et ta présence réconfortante, je n'aurais jamais pu atteindre cette étape cruciale de ma vie. Merci d'être toujours là, dans les moments de doute comme dans les instants de réussite.

À mon père, Mohammed Belkacem, qui a toujours été un modèle de persévérance et de détermination. Merci de m'avoir enseigné l'importance de la rigueur et du travail acharné. Ta sagesse et tes conseils ont été des guides précieux dans les moments de décision et de doute. Ta présence silencieuse mais constante a été une source de sécurité et de motivation pour moi. Ce diplôme, je le dédie aussi à toi, pour toutes les fois où tu as sacrifié tes propres besoins pour prioriser les miens. Merci d'avoir cru en moi, même quand les défis semblaient insurmontables.

Je suis également reconnaissant à mon directeur de thèse, Professeur Abdoulaye Baniré Diallo, pour avoir accepté de m'encadrer. Sa patience et sa rigueur dans l'enseignement des procédures scientifiques ont été essentielles à ma formation et à la conduite de ce projet depuis son commencement.

Enfin, je souhaite remercier Thomas Gisiger pour le temps consacré à la relecture de mon mémoire et pour ses conseils précieux qui ont permis d'élever la qualité de ce travail à un niveau supérieur.

À tous, je vous suis infiniment reconnaissant.

Chatou Mohammed Réda

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

Le Projet traite la complexité NP-difficile associée à la détermination du nombre optimal de clusters pour l'algorithme K-means, une méthode courante en apprentissage automatique non supervisé. L'algorithme K-means nécessite la spécification préalable du nombre de clusters (k), et le défi réside dans le choix de k qui capture le mieux la structure sous-jacente des données. Ce choix est crucial, car il influence la qualité du regroupement des données, généralement évaluée en fonction de la proximité des points de données par rapport aux centres des clusters. Le problème se complique et devient NP-difficile en raison de l'aspect combinatoire de la sélection des clusters, où le nombre de combinaisons possibles croît de manière exponentielle avec la taille des données. Ceci rend la quête d'une solution optimale non seulement exigeante en termes de temps de calcul, mais aussi complexe en raison de l'absence d'une méthode déterministe fiable. De plus, l'obtention du résultat optimal dépend étroitement des données spécifiques utilisées et de la métrique de distance employée. Mon projet cherche à intégrer K-means avec des métaheuristiques telles que la Differential Evolution, le Dual Annealing, le Direct, et le Basin Hopping pour développer une méthode capable de rivaliser avec la méthode du coude tout en réduisant le temps d'exécution. Cette approche hybride est évaluée à l'aide de trois indices de validité (Davies-Bouldin, Calinski-Harabasz, Silhouette), permettant une comparaison de ces indices en termes de précision et de temps d'exécution. Les expériences menées montrent que les algorithmes hybrides peuvent être compétitifs par rapport à la méthode du coude, avec des performances variables selon l'indice de validité utilisé. Les résultats indiquent également que le choix de l'indice de validation peut influencer significativement les performances des algorithmes, et que la stabilité des résultats peut varier en fonction de la taille de l'échantillon et du type de données utilisé. En conclusion, le projet illustre l'efficacité de l'approche hybride pour résoudre le problème du nombre optimal de clusters, offrant une alternative viable qui combine la précision et l'efficacité temporelle. La recherche suggère des directions futures pour l'application de ces algorithmes à divers ensembles de données réels et pour l'exploration d'autres métaheuristiques et indices de validation, ouvrant ainsi la voie à des améliorations supplémentaires dans le domaine de l'analyse de données.

INTRODUCTION

Le problème NP-difficile lié à la sélection du nombre optimal de clusters pour les k-means est une question cruciale en apprentissage automatique non supervisé. Les k-means sont un algorithme de regroupement qui vise à partitionner un ensemble de données en k groupes ou clusters, où k est un paramètre que l'on doit spécifier à l'avance. Le défi consiste à déterminer quel nombre de clusters, k, est le plus approprié pour décrire la structure sous-jacente des données.

0.1 Le problème se pose de la manière suivante :

Étant donné un ensemble de données, nous voulons trouver la valeur optimale de k qui minimise une certaine mesure de la qualité du regroupement, généralement basée sur la distance entre les points de données et les centres de clusters. Une approche courante pour résoudre ce problème est d'utiliser la méthode du coude (elbow method) ou d'autres critères de similarité.

Cependant, le problème devient NP-difficile lorsque nous considérons toutes les combinaisons possibles de k clusters et que nous cherchons à déterminer la meilleure configuration en utilisant un algorithme d'optimisation. En d'autres termes, nous cherchons à trouver la valeur de k qui minimise une fonction objectif, ce qui peut être difficile à résoudre efficacement, car le nombre de combinaisons possibles de clusters peut être exponentiel en fonction de la taille des données. Le problème est considéré comme NP-difficile pour plusieurs raisons :

1. **Combinatoire :** Il existe un nombre exponentiel de façons de diviser les données en k clusters différents. Pour chaque valeur de k, il faut évaluer la qualité de la partition, ce qui peut être coûteux en termes de temps de calcul.
2. **Pas de solution unique :** Il n'y a pas de méthode déterministe pour trouver la solution optimale, car la qualité de la partition dépend souvent des données et de la métrique utilisées (par exemple, la distance euclidienne). Différentes métriques peuvent donner des résultats différents.
3. **Dépendance aux données :** Le nombre optimal de clusters peut varier en fonction des données. Ce qui est optimal pour un ensemble de données peut ne pas l'être pour un autre ensemble de données similaire.

Pour résoudre ce problème NP-difficile, plusieurs approches sont couramment utilisées :

1. **Méthodes heuristiques :** Des méthodes telles que la méthode du coude (*elbow method*), la méthode de la silhouette (*silhouette method*), et la méthode de la validation croisée (*cross-validation*) sont souvent utilisées pour estimer le nombre optimal de clusters en évaluant la qualité des partitions pour différentes valeurs de k.
2. **Méthodes basées sur des indices :** Divers indices, tels que l'indice de Davies-Bouldin, l'indice de Calinski-Harabasz, et l'indice de Dunn, peuvent être utilisés pour mesurer la qualité des clusters et guider le choix de k.
3. Algorithmes de sélection automatique : Certains algorithmes, comme le Gap Statistic et le modèle de mélange gaussien, tentent de trouver automatiquement le nombre optimal de clusters en utilisant des critères statistiques.

0.2 Objectif du projet :

Dans le cadre de notre projet, nous envisageons de combiner l'algorithme K-means avec des métaheuristiques telles que la Differential Evolution, le Dual Annealing, le Direct et le Basin Hopping. L'objectif est de développer une approche capable de concurrencer la méthode du coude en trouvant de manière approximative le nombre optimal de clusters K dans un délai d'exécution raisonnable. Pour évaluer la qualité du partitionnement, nous utiliserons trois indices de validité (Davies-Bouldin, Calinski-Harabasz, Silhouette), ce qui nous permettra également de comparer ces indices en termes de précision et de temps d'exécution.

CHAPITRE 1
CADRE THÉORIQUE : NOTIONS SUR LA THÉORIE DE LA COMPLEXITÉ ALGORITHMIQUE ET L'APPRENTISSAGE AUTOMATIQUE

Dans ce chapitre introductif, nous plongeons dans l'univers fascinant de la théorie de la complexité, un domaine qui explore les limites de ce qui peut être résolu dans des délais raisonnables par les ordinateurs. Nous aborderons des concepts fondamentaux tels que la complexité temporelle, la classe P, la classe NP, ainsi que les problèmes NP-complets et NP-difficiles, qui sont au cœur de l'informatique théorique et ont des implications profondes dans le domaine de l'intelligence artificielle.

Par ailleurs, nous examinerons les diverses catégories de l'apprentissage automatique, en détaillant leurs spécificités et applications : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Chaque catégorie sera étudiée pour sa contribution unique au développement de modèles intelligents capables d'apprendre et de s'adapter.

En particulier, nous consacrerons une attention particulière à l'algorithme k-means, un outil puissant pour l'analyse de regroupement. Nous explorerons également des indices d'évaluation clés comme l'indice de Davies-Bouldin et l'indice de Calinski–Harabasz, qui permettent de jauger la qualité des clusters formés.

Enfin, nous aborderons les métriques d'évaluation essentielles qui mesurent la performance des algorithmes d'apprentissage automatique, tels que le taux d'exactitude, la précision et la racine de l'erreur quadratique moyenne. Ces métriques sont vitales pour évaluer et comparer les modèles de prédiction, constituant ainsi la pierre angulaire de la validation expérimentale en apprentissage automatique.

1.1 La théorie de la complexité :

La théorie de la complexité algorithmique est une branche des sciences informatiques qui étudie la quantité de ressources (comme le temps d'exécution ou la mémoire utilisée) nécessaires pour exécuter des algorithmes en fonction de la taille de l'entrée.Elle est essentielle pour comprendre comment et pourquoi certains problèmes sont plus difficiles à résoudre que d'autres (Sipser, 2005).

1.1.1 **Complexité temporelle :**

La complexité temporelle est un concept fondamental en informatique théorique et en algorithmique. Elle donne une mesure de la quantité de temps qu'un algorithme prend pour résoudre un problème en fonction de la taille de l'entrée.

- La complexité temporelle d'un algorithme décrit le taux de croissance du temps qu'il faut à cet algorithme pour exécuter une tâche en fonction de la taille de l'entrée.
- Habituellement, cette complexité est exprimée en termes de "Big O notation" (notation O grand), qui fournit une borne supérieure sur la croissance du temps nécessaire. Par exemple :
  - $O(1)$: Complexité constante, c'est-à-dire que l'algorithme s'exécute en un temps constant, quel que soit la taille de l'entrée.
  - $O(\log n)$: Complexité logarithmique. Typiquement observée dans les algorithmes de recherche comme la recherche binaire.
  - $O(n)$: Complexité linéaire. Si vous devez parcourir une liste une fois, c'est une opération de complexité linéaire.
  - $O(n \log n)$: Complexité linéarithmique. De nombreux algorithmes de tri efficaces, comme le tri fusion, ont cette complexité.
  - $O(n^2)$: Complexité quadratique. Des algorithmes comme le tri à bulles ou une recherche dans une liste avec une boucle imbriquée ont cette complexité.
- Il est important de noter que la notation O grand ne fournit qu'une estimation approximative de la performance ; elle élimine les termes constants et moins significatifs. Elle est principalement préoccupée par le comportement de croissance pour de grandes entrées (Cormen *et al.*, 2009).

  Comprendre la complexité temporelle est essentiel pour évaluer la performance relative des algorithmes, en particulier lorsque l'on travaille avec de grandes quantités de données. Un algorithme avec une meilleure complexité temporelle sera généralement plus rapide qu'un algorithme avec une moins bonne complexité, surtout lorsque la taille de l'entrée augmente (Sedgewick et Wayne, 2011).

1.1.2 **La classe P :**

La classe **P** est centrale en théorie de la complexité algorithmique. Elle représente l'ensemble des problèmes de décision pour lesquels une solution peut être trouvée en un temps polynomial par un algorithme déterministe (Sipser, 2005).

1. **Définition** :
   — **P** est l'ensemble des problèmes de décision qui peuvent être résolus en temps polynomial par une machine de Turing déterministe. Plus formellement, un problème est dans **P** si et seulement s'il existe un algorithme déterministe qui le résout en temps $O(n^k)$ pour un certain constant k, où n est la taille de l'entrée.
2. **Signification** :
   — Les problèmes dans **P** sont souvent considérés comme "efficacement solvables", car les algorithmes polynomiaux sont généralement réalisables en pratique pour des tailles d'entrée raisonnables.
3. **Exemples de problèmes dans P** :
   — Vérifier si un nombre est premier ou non.
   — Trouver le plus court chemin entre deux sommets dans un graphe.
   — La plupart des algorithmes classiques que nous apprenons en algorithmique, telle que le tri ou la recherche, résolvent des problèmes qui sont dans **P** (Arora et Barak, 2009), (Papadimitriou, 1994).

1.1.3 La classe NP :

La classe **NP** (Problèmes non déterministes polynomiaux) est un autre concept fondamental en théorie de la complexité algorithmique. Elle contient des problèmes pour lesquels une solution donnée peut être vérifiée comme étant corrects ou non en un temps polynomial, même si la solution elle-même peut ne pas être trouvée en temps polynomial (Sipser, 2005).

1. **Définition** :
   — **NP** est l'ensemble des problèmes de décision pour lesquels, si la réponse est "oui", une preuve (ou certificat) de cette réponse peut être vérifiée en temps polynomial par une machine de Turing déterministe.
2. **Signification** :
   — Cela ne signifie pas nécessairement que le problème peut être *résolu* en temps polynomial. Mais si on vous donne une solution potentielle, vous pouvez vérifier sa justesse en temps polynomial.
3. **Exemples de problèmes dans NP** :
   — **Problème du voyageur de commerce (TSP)** : Étant donné une liste de villes et les distances entre

chaque paire de villes, le problème est de trouver le chemin le plus court possible qui visite chaque ville exactement une fois et revient à la ville d'origine.

— **Problème du sac à dos** : Étant donné des poids et des valeurs de n articles, déterminer le nombre d'articles les plus précieux à emporter dans un sac à dos qui a une capacité maximale (Arora et Barak, 2009).

1.1.3.1 Débats et questions ouvertes :

L'une des questions les plus célèbres en informatique théorique est "P = NP ?". Cela demande si chaque problème dont la solution peut être vérifiée en temps polynomial (c'est-à-dire les problèmes dans **NP**) peut également être résolu en temps polynomial (c'est-à-dire les problèmes dans **P**). Jusqu'à présent, cette question reste sans réponse (Papadimitriou, 1994).

1.1.4 Problèmes NP-complets :

Les problèmes **NP-complets** représentent une classe spécifique de problèmes au sein de la classe **NP**, possédant des propriétés intéressantes et essentielles en théorie de la complexité (Sipser, 2005).

1. **Définition** :
   — Un problème est dit **NP-complet** s'il est à la fois dans **NP** et aussi "aussi difficile que" tous les problèmes de **NP**. Cela signifie que si un algorithme polynomial pour un problème NP-complet était trouvé, alors tous les problèmes de NP pourraient également être résolus en temps polynomial, c'est-à-dire P = NP.
   — Plus formellement, un problème est NP-complet si tout problème dans NP peut être réduit à lui en temps polynomial (Papadimitriou, 1994).
2. **Signification** :
   — Les problèmes NP-complets sont considérés comme les problèmes les plus difficiles de la classe NP. Si un problème NP-complet peut être résolu en temps polynomial, alors tous les problèmes de NP peuvent également l'être.
   — Les chercheurs ont identifié une grande variété de problèmes NP-complets dans de nombreux domaines différents (Arora et Barak, 2009).
3. **Exemples de problèmes NP-complets :**

— **Problème de la coloration des graphes** : Étant donné un graphe et un nombre k, le problème consiste à déterminer s'il est possible de colorer le graphe avec k couleurs de manière à ce que deux sommets adjacents n'aient pas la même couleur.

1.1.5 Problèmes NP-difficiles :

Les problèmes **NP-difficiles** occupent une place fondamentale en théorie de la complexité algorithmique. Ils capturent l'essence de la difficulté inhérente à la classe **NP** tout en s'étendant au-delà de celle-ci (Sipser, 2005).

1. **Définition** :
   — Un problème est dit **NP-difficile** s'il est "au moins aussi difficile que" le plus difficile des problèmes dans **NP**. Plus formellement, si un problème A est NP-difficile, alors tout problème dans **NP** peut être réduit à A en temps polynomial.
   — Il est important de noter que contrairement aux problèmes NP-complets, les problèmes NP-difficiles ne sont pas nécessairement des problèmes de décision. Cela signifie qu'ils peuvent être des problèmes d'optimisation ou d'autres types de problèmes qui ne rentrent pas directement dans la catégorie des problèmes de décision (Arora et Barak, 2009), (Papadimitriou, 1994).
2. **Relation avec NP-complet** :
   — Tout problème NP-complet est également NP-difficile, mais l'inverse n'est pas nécessairement vrai. Un problème peut être NP-difficile sans être NP-complet (Garey et Johnson, 1979).
3. **Exemples de problèmes NP-difficiles** :
   — Le problème de décision associé au **problème du voyageur de commerce** : existe-t-il un circuit qui visite toutes les villes une fois et dont la longueur totale est inférieure à k ?
   — **Problème de l'ordonnancement des travaux** : étant donné un ensemble de travaux avec des durées et des échéances spécifiques, peut-on ordonner les travaux de telle manière que tous soient terminés avant leurs échéances respectives ?

1.2 Notions d'apprentissage automatique

1.2.1 Définitions :

L'apprentissage automatique, souvent désigné par son terme anglais "machine learning", est un sous-domaine de l'intelligence artificielle (IA) qui se concentre sur la conception de systèmes qui peuvent apprendre à partir de données. Au lieu d'être explicitement programmé pour effectuer une tâche, ces systèmes utilisent des algorithmes pour analyser des données, apprendre des patterns et prendre des décisions basées sur ces informations.

La base de l'apprentissage automatique est la notion d'apprendre à partir d'un ensemble de données ou d'expériences. Les algorithmes d'apprentissage automatique utilisent des données pour identifier des modèles ou des relations. Une fois qu'un algorithme a été "entraîné" sur un certain ensemble de données, il peut commencer à faire des prédictions ou des décisions sans qu'un humain lui donne des instructions explicites.

En termes plus techniques, l'apprentissage automatique est une méthode d'analyse de données qui automatise le processus de construction de modèles analytiques. Il s'agit d'un ensemble d'algorithmes qui utilisent des statistiques pour trouver des structures dans les données. Une fois que ces structures sont trouvées, le système peut faire des prédictions ou prendre des décisions sans être explicitement programmé pour effectuer la tâche (Bishop, 2006).

1.2.2 Catégories de l'apprentissage automatique

1.2.2.1 Apprentissage supervisé :

L'apprentissage supervisé est l'une des méthodes les plus courantes et les plus simples à comprendre dans le domaine de l'apprentissage automatique. Comme son nom l'indique, il fonctionne sous la "supervision" d'un enseignant. En termes simples, nous fournissons à l'algorithme de l'apprentissage automatique une entrée, et pour chaque entrée, la sortie correcte (également appelée étiquette ou label). L'algorithme apprend progressivement à partir de ces données d'entraînement et tente de produire la bonne sortie pour les nouvelles entrées.

Caractéristiques de l'apprentissage supervisé :

1. *Ensemble d'entraînement* : L'ensemble de données utilisé pour l'apprentissage supervisé est appelé ensemble d'entraînement. Chaque exemple d'entraînement est composé d'une entrée et d'une sortie correspondante.
2. *Objectif* : L'objectif principal est de permettre à l'algorithme de généraliser à partir de cet ensemble d'entraînement pour produire la bonne sortie pour de nouvelles entrées.
3. *Types de tâches* :
   — **Classification** : Lorsque la sortie est une catégorie (par exemple, "spam" ou "non-spam", "chat", "chien", etc.)
   — **Régression** : Lorsque la sortie est une quantité continue (par exemple, prédire le prix d'une maison).

Évaluation : Une fois que le modèle est formé, il est évalué à l'aide d'un ensemble de tests distinct pour mesurer sa précision. Si la précision est jugée satisfaisante, le modèle peut être déployé pour faire des prédictions sur de nouvelles données.

L'apprentissage supervisé est largement utilisé dans diverses applications telles que la détection de fraudes, la reconnaissance vocale, la reconnaissance d'images et la prédiction de tendances du marché (Hastie *et al.*, 2009), (Bishop, 2006).

1.2.2.2 **Apprentissage non supervisé :**

L'apprentissage non supervisé est une méthode d'apprentissage automatique où les algorithmes sont formés en utilisant des données non étiquetées. Contrairement à l'apprentissage supervisé, où chaque exemple d'entraînement est associé à une étiquette, dans l'apprentissage non supervisé, l'objectif principal est de détecter des structures intrinsèques et des modèles sous-jacents à partir des données elles-mêmes.

Caractéristiques de l'apprentissage non supervisé :

1. *Données d'entrée* : Les algorithmes reçoivent des données d'entrée sans étiquettes correspondantes.
2. *Objectif* : Découvrir la structure cachée à partir de données non étiquetées.
3. *Types de tâches* :
   — **Clustering** : Il s'agit de regrouper des points de données similaires en clusters. Exemple : Regrou-

per des articles de journaux sur des sujets similaires sans les lire, avec des algorithmes comme le K-Means ou DBSCAN à titre d'exemple.

— **Réduction de dimensionnalité** : Réduire le nombre de variables en jeu tout en conservant l'essentiel de l'information, comme l'analyse en composantes principales (PCA).

Applications : Cette méthode est souvent utilisée pour la segmentation de la clientèle, la recommandation de produits, l'analyse de grands ensembles de données pour identifier des modèles, etc.

L'apprentissage non supervisé offre un moyen d'explorer des ensembles de données complexes et volumineux pour découvrir des informations qui ne sont pas immédiatement évidentes. Il est souvent utilisé lorsque nous n'avons pas une idée claire de ce que nous cherchons (Hastie *et al.*, 2009), (Bishop, 2006).

1.2.3 **L'algorithme k-means :**

Le k-means est une technique de classification vectorielle utilisée en traitement du signal, qui a pour but de diviser un ensemble de données en k groupes, de sorte que chaque donnée soit associée au groupe dont le centre (ou moyenne) est le plus proche. L'objectif principal du k-means est de regrouper des données similaires et de révéler des structures cachées dans les données. Pour cela, le k-means détermine un nombre prédéfini (k) de groupes dans un ensemble de données (MacQueen, 1967).

La première étape consiste à choisir le nombre de groupes k, ce qui correspond au nombre de centres (ou centroïdes) nécessaires. Un centroïde est un point fictif ou réel qui représente le centre d'un groupe. Ensuite, chaque donnée est affectée à un groupe de manière à minimiser la somme des carrés des distances à l'intérieur de chaque groupe.

En d'autres termes, l'algorithme k-means sélectionne k centroïdes, puis attribue chaque donnée au groupe le plus proche, tout en minimisant la taille des centroïdes. L'initialisation de l'algorithme se fait en choisissant au hasard un premier ensemble de centroïdes, qui servent de points de départ pour chaque groupe. L'algorithme procède ensuite par des itérations successives pour ajuster la position des centroïdes. Le k-means s'arrête lorsque les centroïdes ont atteint une position stable ou lorsque le nombre maximal d'itérations a été réalisé (Lloyd, 1982).

Algorithm 1 L'algorithme K-means

Input : k le nombre de clusters, les données à classer

Output : Un ensemble de k clusters

while le nombre d'itérations n'est pas atteint ou les centroïdes ne changent pas de position **do**

1. *Attribuer chaque point de données au cluster dont le centroïde est le plus proche. La proximité est souvent mesurée à l'aide de la distance euclidienne.*

2. *Mettre à jour les centroïdes, c'est-à-dire, chaque centroïde va prendre la valeur moyenne du cluster qui le représente.*

end while

L'algorithme K-means se concentre sur la minimisation des variances intra-cluster, c'est-à-dire les sommes des carrés des distances euclidiennes, plutôt que sur les distances euclidiennes directes. Cette approche soulève un problème bien connu en théorie de la localisation, le problème de Weber. Ce dernier consiste à trouver un point dans un espace qui minimise la somme des coûts de transport depuis ce point vers n destinations, où chaque destination a un coût associé par unité de distance. Le problème de Weber est réputé pour sa complexité NP-difficile et son coût élevé en temps de calcul. Les approches heuristiques classiques tendent généralement à converger vers un optimum local. Pour améliorer les résultats, on peut recourir aux métaheuristiques, des stratégies d'optimisation destinées à résoudre des problèmes pour lesquels aucune méthode classique efficace n'est connue.

1.2.3.1 Apprentissage par renforcement :

L'apprentissage par renforcement est une méthode d'apprentissage automatique où un agent apprend à prendre des décisions en effectuant des actions dans un environnement de manière à maximiser une certaine récompense cumulative. Contrairement à l'apprentissage supervisé et non supervisé, où l'objectif est de faire correspondre des entrées à des sorties ou de trouver des structures dans des données, l'apprentissage par renforcement est axé sur la prise de décisions séquentielles en explorant et exploitant des connaissances pour maximiser un gain à long terme.

Caractéristiques de l'apprentissage par renforcement :

1. *Agent et environnement* : L'agent interagit avec l'environnement en prenant des actions et reçoit des

récompenses en retour.

2. *Politique* : C'est la stratégie adoptée par l'agent pour prendre une action à un état donné.
3. *Récompense* : Une signalisation immédiate reçue après avoir pris une action dans un état particulier.
4. *Fonction de valeur* : Estime la récompense attendue à partir d'un état ou d'une paire état-action.
5. *Exploitation vs Exploration* : L'agent doit choisir entre exploiter ses connaissances actuelles ou explorer de nouvelles actions.

Applications : L'apprentissage par renforcement a été utilisé avec succès dans divers domaines, tels que les jeux (par exemple, AlphaGo de DeepMind), la robotique, la finance, et bien d'autres (Sutton et Barto, 2018).

1.3 **Hyperparamètres :**

En apprentissage automatique, un hyperparamètre est un paramètre configuré avant le processus d'apprentissage pour guider celui-ci. Ce terme distingue les hyperparamètres des paramètres du modèle, qui sont appris durant l'entraînement. Les hyperparamètres, tels que le taux d'apprentissage ou le choix de l'optimiseur, spécifient les détails du processus d'apprentissage, soulignant ainsi leur rôle crucial dans la performance du modèle (Yang et Shami, 2020).

Contrairement aux paramètres du modèle, les hyperparamètres ne sont pas dérivés des données, mais doivent être définis par l'utilisateur. Leur choix adéquat est essentiel pour optimiser la performance d'un modèle. Des techniques telles que l'optimisation des hyperparamètres vise à trouver la meilleure combinaison d'hyperparamètres pour minimiser une fonction de perte prédéfinie sur des données de test, bien que ces méthodes ne soient généralement pas basées sur le gradient (Claesen et Moor, 2015).

La principale différence entre les hyperparamètres et les paramètres réside dans le fait que les premiers sont fixés avant l'apprentissage et ne changent pas, tandis que les seconds sont appris pendant l'apprentissage. Les hyperparamètres influencent le modèle, mais ne font pas partie intégrante de celui-ci, ce qui les distingue clairement des paramètres qui sont ajustés pour réduire l'erreur de prédiction du modèle.

Les paramètres sont des composants internes du modèle ajustés au cours du processus d'apprentissage pour mapper précisément les entrées aux sorties. Ils sont initialement définis à des valeurs spécifiques et modifiés à travers des algorithmes d'optimisation pour améliorer la performance du modèle. À la fin de

l'apprentissage, ces paramètres constituent le cœur du modèle.

1.4 **La fonction objective :**

La fonction objectif est un moyen de maximiser (ou de minimiser) une valeur numérique. Dans le monde réel, il peut s'agir du coût d'un projet, d'une quantité de production, d'une valeur de profit ou même des matériaux économisés grâce à un processus rationalisé. Avec la fonction objectif, nous essayons d'arriver à une cible de production, de profit, d'utilisation des ressources, etc. La fonction objectif tente de maximiser les profits ou de minimiser les pertes en fonction d'un ensemble de contraintes et de la relation entre une ou plusieurs variables de décision. Les contraintes peuvent faire référence à la capacité, la disponibilité, les ressources, la technologie, etc.

Dans le cadre de regroupement (clustering), la fonction objectif est appelée l'indice de validité. L'indice de validité de cluster nous renseigne sur la qualité de la partition qui a été trouvée et permet de déterminer la partition optimale. Par conséquent, l'indice de validité des grappes peut également être utilisé pour rechercher le nombre optimal de grappes lorsque le nombre de grappes dans l'ensemble de données n'est pas connu à l'avance (Zhang *et al.*, 2008).

La plupart des indices de validation proposés au cours des dernières décennies se sont concentrés sur la compacité et la séparation. La séparation est une mesure de l'isolement des clusters les uns par rapport aux autres et la compacité est une mesure de la proximité des objets de données au sein d'un cluster. Une faible valeur de variance est un indicateur de proximité. Les membres de chaque cluster doivent être aussi proches les uns des autres que possible et les clusters doivent être largement séparés (Žalik et Žalik, 2011).

De nombreux indices de validité de cluster ont été développés pour évaluer la qualité des partitions dans le but de trouver un partitionnement optimal composé de clusters compacts et bien séparés, dans le cadre de notre projet en va se concentrer sur deux indices, le Davies-Bouldin et Calinski-Harabasz.

1.4.1 **L'indice de Davies-Bouldin :**

Proposer par Davies et Bouldin en 1979 (Davies et Bouldin, 1979). L'indice de Davies-Bouldin est basé sur l'estimation approximative des distances entre les clusters et leurs dispersions pour obtenir une valeur finale représentative de la qualité de la partition. Certaines statistiques sont utilisées pour estimer les distances entre les clusters, telles que la distance euclidienne, la distance cylindrique et la distance des hyper rectangles (Karo *et al.*, 2017).Le but de l'indice Davis-Bouldin est de maximiser la distance inter-cluster et en même temps minimiser la distance entre les points du même cluster. La valeur de DBI est utilisée pour mesurer la qualité du partitionnement et pour sélectionner le numéro de cluster optimal.Pour définir l'indice DB, nous devons définir la mesure de dispersion et la mesure de similarité des clusters. Dans [4.1], la dispersion S_i du cluster C_i (1) et la séparation D_{ij} entre le ième et le jème clusters (2) sont définies comme suit :

$$S_i = \left(\frac{1}{|C_i|} \sum_{x \in C_i} D^p(x, c_i) \right)^{\frac{1}{p}}, p > 0 \tag{1.1}$$

Où $|C_i|$ est le nombre de points de données dans le cluster C_i et c_i est le centre du cluster C_i, et :

$$D_{ij} = \left(\sum_{l=1}^{d} |v_{il} - v_{jl}|^t \right)^{\frac{1}{t}}, t > 1 \tag{1.2}$$

Où v_i et v_j sont les centroïdes des clusters C_i et C_j, respectivement. Ensuite, l'index DB est défini comme suit :

$$V_{DB} = \frac{1}{k} \sum_{i=1}^{k} R_i \tag{1.3}$$

Où k est le nombre de clusters et R_i est défini comme :

$$R_i = \max_{i \neq j} R_{ij} \tag{1.4}$$

Où R_{ij} est la mesure de similarité entre les clusters C_i et C_j, et elle est défini comme :

$$R_{ij} = \frac{S_i + S_j}{D_{ij}} \tag{1.5}$$

La principale limite du DBI c'est qu'aucun de ses termes ne considère la géométrie de la distribution spatiale des clusters. Cette limitation est directement liée à l'utilisation des distances moyennes entre les clusters. Plus précisément, l'utilisation des moyennes comme mesure pour calculer la distance entre les grappes échoue lorsqu'elles présentent des variances différentes ou que les variances ne sont pas égales sur tous les axes. L'utilisation de la distance euclidienne pour calculer les distances entre les clusters peut conduire à des situations où deux clusters, dont les géométries, sont proches, mais ses moyennes sont éloignées l'un de l'autre, alors elles sont considérées comme plus éloignées que deux clusters dont les géométries sont plus éloignées, mais ses moyennes sont plus proches (Thomas *et al.*, 2013).

1.4.2 **L'indice de Calinski–Harabasz (CH) :**

CH est une mesure pour évaluer la séparation entre les clusters et la compacité qui existe à l'intérieur du même cluster. L'avantage de cet indice c'est qu'il peut être utilisé avec n'importe quel algorithme de clustering, et contrairement à d'autres indices d'évaluation, il n'a pas le problème connu sous le nom de Small Sample Size Problem (Niijima et Okuno, 2008) cela se produit lorsque le nombre de caractéristiques est supérieur au nombre d'instances, ou lorsque deux caractéristiques ou plus sont identiques (Solorio-Fernández *et al.*, 2016).

Le CH évalue la cohésion par la somme des distances des éléments du cluster par rapport à leurs barycentres respectifs. Le critère de séparation est calculé à partir de la somme des distances entre le centroïde de chaque cluster et le centroïde global du jeu de données. Le coût de calcul (temps d'exécution) de cette méthode n'est pas élevé et surpasse, généralement, les autres indices de validité. Sa complexité est égale à $O(n)$ (Lima et Cruz, 2020). L'indice Calinski–Harabasz est calculé comme suit :

$$I_{CH} = \frac{N-C}{C-1} \frac{\sum_{i=1}^{C} d(u_i, U)}{\sum_{i=1}^{c} \sum_{x_j \in CL_i} d(x_j, u_i)} \tag{1.6}$$

Où $u_i \in R^N$ pour le cluster non vide et CL_i correspond au centre du cluster et elle est défini par :

$$u_i = \frac{1}{M_i} \sum_{y_j \in CL_i} y_j,\ i = 1, \ldots, C \tag{1.7}$$

Ou M_i correspond à la cardinalité du cluster i et U correspond au centre de gravité du jeu de données :

$$U = \frac{1}{M} \sum_{j=1}^{M} y_j. \tag{1.8}$$

L'indice CH a démontré dans plusieurs études son potentiel solide pour valider les solutions de clustering (Łukasik *et al.*, 2017).

1.5 Les métriques d'évaluation

1.5.1 Taux d'exactitude :

Le taux d'exactitude (ou "accuracy" en anglais) est une mesure utilisée en apprentissage automatique pour évaluer la performance d'un modèle de classification. Elle est définie comme le rapport entre le nombre de prédictions correctes et le nombre total de prédictions.

Formule :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}} \tag{1.9}$$

Dans le contexte de la classification binaire (c'est-à-dire, lorsque vous avez seulement deux classes), l'exactitude peut être détaillée comme suit :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.10}$$

Où :

— TP est le nombre de vrais positifs (true positives).
— TN est le nombre de vrais négatifs (true negatives).
— FP est le nombre de faux positifs (false positives).
— FN est le nombre de faux négatifs (false negatives).

Il est important de noter que, bien que l'exactitude soit une mesure courante, elle peut être trompeuse, en particulier dans les situations où les classes sont très déséquilibrées. Par exemple, imaginez une situation où 95% des échantillons appartiennent à la classe A et seulement 5% à la classe B. Un modèle qui prédit toujours la classe A aura une exactitude de 95%, mais il ne sera d'aucune utilité pour identifier la classe B.

Pour cette raison, il est souvent utile de considérer d'autres métriques (comme la précision, le rappel, et le F1-score) pour avoir une image plus complète des performances d'un modèle (Hastie *et al.*, 2009).

1.5.2 **La précision :**

La précision est une mesure utilisée en apprentissage automatique pour évaluer la performance des modèles de classification, en particulier dans des contextes où les faux positifs ont des conséquences significatives. Elle est particulièrement utile lorsque les classes sont déséquilibrées.

La précision est définie comme le rapport entre le nombre de vrais positifs et la somme des vrais positifs et des faux positifs. C'est une mesure de la qualité des prédictions positives faites par un modèle.

Formule :

$$\text{Précision} = \frac{TP}{TP + FP} \tag{1.11}$$

Où :

— TP est le nombre de vrais positifs (*true positives*).
— FP est le nombre de faux positifs (*false positives*).

Intuitivement, la précision répond à la question : "Parmi toutes les instances que le modèle a classées comme positives, combien sont réellement positives ?"

Il est à noter que la précision est souvent utilisée en conjonction avec le rappel (ou sensibilité). Le rappel est le ratio des vrais positifs par rapport à la somme des vrais positifs et des faux négatifs. Ensemble, la précision et le rappel fournissent une image plus complète des performances d'un modèle que l'exactitude

seule, en particulier dans les situations de classes déséquilibrées.Pour équilibrer la précision et le rappel, le score F1 est souvent utilisé, qui est la moyenne harmonique des deux (Sokolova et Lapalme, 2009).

1.5.3 La Racine de l'Erreur Quadratique Moyenne (Root Mean Squared Error, RMSE :)

La Racine de l'Erreur Quadratique Moyenne (Root Mean Squared Error, RMSE) est une métrique couramment utilisée pour évaluer la précision d'un modèle de prédiction. Plus spécifiquement, elle mesure l'écart moyen entre les prédictions d'un modèle et les observations réelles.

Étant donné une série de n observations réelles $y_1, y_2, \ldots, y_n$ et leurs prédictions correspondantes $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$, l'erreur quadratique pour chaque observation est donnée par $(y_i - \hat{y}_i)^2$. Pour obtenir l'erreur quadratique moyenne (MSE, pour Mean Squared Error), nous prenons la moyenne de ces erreurs sur toutes les observations :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1.12}$$

La RMSE est simplement la racine carrée de la MSE :

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1.13}$$

La RMSE donne une idée de la magnitude des erreurs commises par le modèle. Une RMSE de zéro indique une prédiction parfaite. Plus la RMSE est élevée, moins le modèle est précis. L'avantage de la RMSE par rapport à la MSE est qu'elle est dans les mêmes unités que les observations, ce qui la rend souvent plus interprétable (Hastie *et al.*, 2009).

1.5.4 Erreur Absolue Moyenne (MAE) :

La métrique d'erreur absolue moyenne (MAE, pour *Mean Absolute Error*) est une mesure couramment utilisée pour évaluer la précision de modèles de prédiction. Contrairement à d'autres métriques telles que l'erreur quadratique moyenne, le MAE fournit une évaluation directe de l'ampleur de l'erreur sans la pondérer, ce qui le rend particulièrement utile lorsque vous souhaitez éviter de donner trop d'importance aux grandes erreurs dans le contexte de votre analyse.

Formellement, le MAE est défini comme la moyenne des valeurs absolues des écarts entre les valeurs prédites et les valeurs réelles. Mathématiquement, il est exprimé comme suit :

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

où y_i sont les valeurs réelles, $\hat{y}_i$ sont les valeurs prédites, et n est le nombre total de prédictions effectuées.

Le MAE est particulièrement intéressant, car il est facile à interpréter ; une valeur de MAE de 0 indique une prédiction parfaite, et des valeurs plus élevées indiquent des erreurs plus importantes. En outre, étant donné que chaque erreur est pondérée également, cette métrique est robuste aux anomalies et peut être plus représentative de l'erreur typique lorsque le jeu de données contient des valeurs aberrantes.

Pour une introduction et une analyse détaillée de l'erreur absolue moyenne, je vous recommande de consulter les ouvrages suivants (Hyndman et Athanasopoulos, 2018), (Willmott et Matsuura, 2005)

CHAPITRE 2
ÉTAT DE L'ART

Le chapitre 2 de cette thèse explore en profondeur les diverses méthodologies et techniques avancées pour déterminer le nombre optimal de clusters dans les ensembles de données, une étape cruciale dans l'amélioration de la précision des résultats du clustering. Nous débutons par examiner des méthodes traditionnelles telles que la visualisation pour identifier des valeurs appropriées de k, la méthode du coude, l'approche basée sur le coefficient de silhouette, et la méthode de la statistique d'écart (*gap statistic*), qui offrent des perspectives fondamentales sur la segmentation optimale.

Nous explorons également des algorithmes spécifiques comme l'algorithme canopée et le Bisecting k-means (BKM), en discutant de leur fonctionnement ainsi que de leurs avantages et inconvénients respectifs. Le regroupement automatique est ensuite traité à travers des techniques innovantes telles que l'Évolution Différentielle (ED) et l'Optimisation par Essaims de Particules pour le Clustering Dynamique (DCPSO), enrichissant notre compréhension des méthodes de clustering avancées.

Le cœur de ce chapitre se concentre sur les algorithmes métaheuristiques inspirés de la nature, qui sont intégrés avec le k-means pour résoudre les problèmes de clustering automatique de données. Parmi ceux-ci, nous abordons DIRECT (DIvide a REctangle), le Basin Hopping, le Sequential Least SQuares Programming (SLSQP), ainsi que les célèbres algorithmes Broyden–Fletcher–Goldfarb–Shanno (BFGS) et Nelder-Mead, en mettant en lumière leur procédure, leurs avantages, et leurs limites.

Ce chapitre vise non seulement à présenter une synthèse détaillée des méthodes existantes mais aussi à introduire des approches novatrices et leurs implications pratiques dans le domaine du clustering, permettant ainsi une meilleure adaptation aux défis posés par des ensembles de données complexes et variés.

2.1 Méthodes pour déterminer le nombre optimal de clusters :

2.1.1 Valeurs de K déterminées par visualisation :

La vérification visuelle est souvent privilégiée pour sa facilité et son aptitude à fournir une interprétation des résultats. Il est courant de recourir à des illustrations visuelles pour mettre en lumière les limites d'un algorithme ou pour démontrer les résultats de regroupement escomptés. La fiabilité de l'évaluation des

résultats de clustering par des méthodes de visualisation varie en fonction de leur applicabilité. Il se peut que certaines techniques de regroupement ne soient pas adaptées à des types spécifiques de données. Les exemples présentés dans la Figure 3 servent à souligner ces situations.

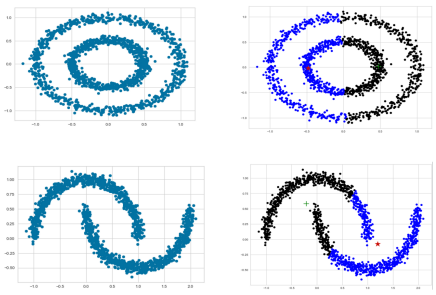

Figure 2.1 Exemple de l'application de k-means sur des données sphériques.

Dans l'illustration de la Figure 3, le premier scénario montre que le cercle intérieur devrait constituer un groupe distinct, tandis que le cercle extérieur devrait former un autre groupe séparé. Toutefois, l'algorithme K-Means ne parvient pas à séparer adéquatement ces ensembles de points en raison de l'impossibilité de gérer les centroïdes superposés. Dans le second scénario, où deux demi-cercles sont censés représenter deux groupes différents, K-Means échoue également à distinguer les deux catégories. Ces exemples démontrent que K-Means n'est pas capable de fournir le regroupement escompté pour de tels cas, révélant que cette méthode n'est pas adaptée à ces types de structures de données. Les ensembles de données réels sont généralement complexes, incluant souvent du bruit et des anomalies. Bien que l'algorithme de clustering K-Means soit reconnu pour sa robustesse, il est important de reconnaître ses limites face à certaines configurations de données.

À ce jour, l'utilisation de la visualisation pour examiner les résultats demeure une approche bénéfique, tant pour la sélection du paramètre K que pour la confirmation de la qualité du clustering obtenu. Cette stratégie est particulièrement préconisée dans les situations où les regroupements escomptés peuvent être clairement définis à l'avance (Pham *et al.*, 2005).

2.1.2 **La méthode du coude :**

La méthode du coude est un procédé heuristique employé pour identifier le nombre idéal de clusters en appliquant l'algorithme K-means sur un jeu de données à travers un éventail de valeurs pour K (typiquement de 1 à 20). Pour chaque valeur de K, on calcule la somme des carrés des erreurs (SSE), puis on trace un graphique de SSE en fonction de K lorsque le graphique forme une inflexion semblable à un coude, cette inflexion indique la valeur optimale de K (Ketchen et Shook, 1996).

L'essence de cette méthode repose sur le principe que l'accroissement du nombre de clusters améliore la précision du modèle jusqu'à un certain point, après lequel les bénéfices diminuent. Le point de "coude" symbolise le moment où l'ajout de clusters supplémentaires ne résulte plus en une amélioration significative, marquant ainsi un équilibre entre précision et simplicité.

Cependant, dans la pratique, l'identification du coude peut s'avérer difficile, particulièrement avec des données peu ou mal regroupées. L'analyse visuelle des graphiques peut révéler des cas où la diminution du SSE se poursuit de manière graduelle sur toute la plage de valeurs de K étudiées, sans présenter de point d'inflexion marqué. Ce phénomène souligne une des limites de la méthode du coude, notamment dans les situations où la structure des données ne permet pas une distinction claire du nombre optimal de clusters (Shi *et al.*, 2021).

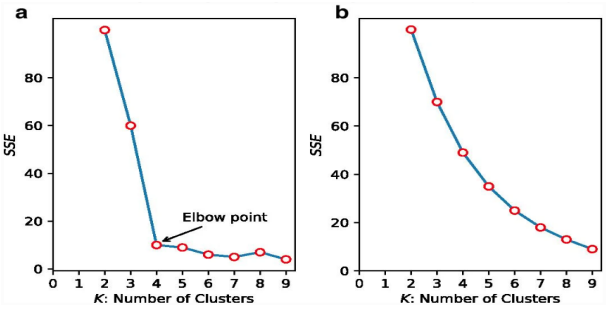

Figure 2.2 Observations de l'évolution de la valeur SSE avec la valeur K (Shi *et al.*, 2021).

2.1.3 **Méthode base sur le coefficient de silhouette :**

La méthode Silhouette, introduite par (Rousseeuw, 1987), est reconnue pour son efficacité dans la détermination du nombre optimal de clusters. Elle évalue la qualité du regroupement en calculant la distance moyenne entre un point et les autres points de son cluster, ainsi que la distance moyenne entre son cluster

et le cluster le plus proche. Le score de silhouette (S) est le critère de mesure de cette méthode, exprimé par la formule :

$$\frac{b-a}{\max(a,b)} \tag{2.1}$$

Désigne la distance moyenne intra-cluster et b la distance moyenne au cluster le plus proche. Un score de silhouette varie de -1 à 1, indiquant une classification très distincte à l'approche de 1 et une classification inappropriée à l'approche de -1 (Shi *et al.*, 2021).

Cette méthode analyse simultanément deux aspects : la séparation entre les clusters et la cohésion à l'intérieur des clusters, offrant une vue d'ensemble sur la proximité et l'isolement des points de données par rapport à leur propre cluster et aux autres clusters (Sudheera *et al.*, 2016).

Pour identifier le nombre idéal de clusters, la méthode Silhouette suit une démarche similaire à celle de la méthode du coude. Elle applique l'algorithme K-means sur une série de valeurs de K, calculant le score de silhouette pour chaque valeur. Le nombre de clusters donnant un score de silhouette maximal, le plus proche de 1, est considéré comme la configuration optimale, reflétant une séparation claire et une cohésion forte au sein des clusters.

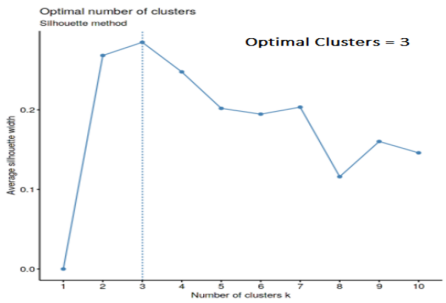

Figure 2.3 Observations de l'évolution de la valeur silhouette avec la valeur K (Androniceanu *et al.*, 2018).

L'exemple montre le résultat de la méthode de silhouette sur un ensemble de données (Abdulhafedh, 2021), la valeur la plus proche de 1 correspond au nombre de clusters optimal k=3. L'avantage de cette méthode par apport a la méthode de coude, c'est qu'on peut distinguer facilement la solution son ambiguïté.

2.1.4 **La méthode de la statistique d'écart (gap statistic)**

La statistique d'écart a été développée par Tibshirani (Tibshirani *et al.*, 2001). C'est une sorte d'algorithme d'exploration de données qui vise à améliorer le processus de clustering par une estimation efficace du

meilleur nombre de clusters. Cette méthode est conçue pour s'appliquer à toutes les techniques de clustering.

L'idée de base de Gap Statistic est d'introduire des mesures de référence, qui peuvent être obtenue par la méthode d'échantillonnage de Monte-Carlo et de calculer la somme des carrés de la distance euclidienne entre deux mesures dans chaque classe.Les résultats de regroupement de la distribution moyenne nulle de référence construite sont comparés pour déterminer le nombre optimal de grappes dans l'ensemble de données (Yuan et Yang, 2019). L'algorithme va procéder comme suite. Premièrement, on calcule la somme des carrés intra-clusters (W_k, k est le nombre de clusters), qui est représentée par

$$W_k = \sum_{i=1}^{k} \frac{1}{2n_i} \sum_{j=1}^{n_i} \sum_{d=1}^{n_i} x^2(x_j, x_d) \in C_i \tag{2.2}$$

Où n_i est le nombre de données dans le cluster r, C_i. L'algorithme de la statistique Gap normalise le graphe de $log(W_k)$ en le comparant à sa valeur attendue sous une distribution de référence nulle appropriée de l'ensemble de données. La distribution de référence nulle est générée pour chaque entité de référence uniformément sur la plage des valeurs observées pour cette entité. La distribution de référence nulle est calculée à l'aide de $\frac{1}{B} \sum_{b=1}^{B} log(W_{kb})$ .L'ensemble de données de référence. Ainsi, $Gap(k)$ est défini comme

$$Gap(k) = \frac{1}{B} \sum_{b=1}^{B} log(W_{kb}) - log(W_k) \tag{2.3}$$

$Gap(k)$ est calculé avec le nombre de clusters $k = 1, 2, \ldots, T$. Ou T est le nombre maximal de clusters choisis arbitrairement.L'erreur de simulation est calculée comme suite

$$s_k = \sqrt{1 + \frac{1}{B}}.sd(k) \tag{2.4}$$

Où sd(k) est l'écart type de la distribution nulle de référence.Alors le minimum k qui satisfait l'équation suivante est défini comme étant le nombre optimal de clusters (Arima *et al.*, 2008)

$$Gap(K) \geq Gap(k+1) - s_{k+1} \tag{2.5}$$

La méthode de la statistique d'écart a démontré qu'elle donne des résultats meilleurs que la méthode de coude et silhouette dans de nombreux projets (Yang *et al.*, 2019). Cependant, la déficience de la méthode des écarts dans la détermination du nombre de grappes a également été démontrée dans des études plus récentes. Par exemple, la méthode des écarts n'a pas réussi à détecter la structure à quatre grappes dans les données simulées qui contiennent des grappes bien séparées générées à partir de distributions exponentielles distinctes. Et il a été démontré que la méthode tend à surestimer le nombre de clusters en application (Yan et Ye, 2007).

2.1.5 **L'algorithme canopée**

L'algorithme de canopée est une méthode préliminaire de regroupement non supervisé, développé par McCallum et ses collaborateurs (McCallum *et al.*, 2000). Cette technique sert fréquemment de prétraitement pour des algorithmes plus complexes comme K-means ou le clustering hiérarchique. L'algorithme opère en définissant deux seuils de proximité, $T1$ et $T2$, et en initiant aléatoirement un centre de cluster. Il évalue ensuite la distance euclidienne entre chaque échantillon et ce centre, affectant les échantillons à un cluster selon ces seuils. Cette étape de préclustering segmente les données en plusieurs groupes préliminaires. Les résultats, y compris les centres des clusters identifiés par l'algorithme de canopée, servent ensuite de points de départ pour affiner le clustering avec des méthodes comme K-means, en facilitant et en optimisant le partitionnement final des données (Zhang *et al.*, 2018).

L'algorithme fonctionne en choisissant aléatoirement un vecteur initial A parmi l'ensemble des données X, puis en calculant la distance D entre A et chaque autre vecteur dans X. Un vecteur est classé dans la même canopée que A si sa distance D est inférieure au seuil $T1$, et est exclu de la sélection future comme centre potentiel si D est inférieure à $T2$. Cette procédure est répétée jusqu'à épuisement des vecteurs candidats dans X, permettant ainsi de former des groupes préliminaires ou "canopées" avant une analyse de clustering plus fine (Yuan et Yang, 2019).

Algorithm 2 L'algorithme Canopy

Input : Ensemble de données D

Output : Le nombre de clusters K et les centres initiaux des clusters de l'ensemble de données

1. *Initialiser la liste de tableaux et définir* $T1, T2$

2. *Sélectionner l'échantillon* S *au hasard*

while il existe un échantillon $a \in D$ **do**

  if ensemble de données $D \neq \emptyset$ **then**

    3. Calculer la distance D

    if $D < T1$ **then**

      Ajouter a à l'ensemble des canopées C_i

    else if $D < T2$ **then**

      Supprimer l'échantillon a

    end if

  else

    Sortir de l'algorithme

  end if

end while

2.1.6 L'algorithme Bisecting k-means (BKM)

L'algorithme bisecting k-means est une variante de l'algorithme de clustering k-means, qui est largement utilisé en apprentissage automatique et en analyse de données pour regrouper des données non étiquetées en k groupes distincts. L'algorithme bisecting k-means est particulièrement utile lorsque le nombre de clusters n'est pas connu à l'avance.

2.1.6.1 Fonctionnement de l'algorithme Bisecting k-means :

1. **Initialisation** :
   - Commencez par traiter l'ensemble de données entier comme un seul cluster.
   - Définissez une liste de clusters avec ce cluster initial.
2. **Bissection** :
   - Pour chaque cluster dans la liste de clusters :
     (a) Divisez le cluster en deux sous-clusters en utilisant l'algorithme k-means standard (avec $k =$

2).

(b) Calculez le SSE (somme des erreurs au carré) pour ces deux sous-clusters.

3. **Sélection du cluster à diviser** :
   — Parmi tous les clusters possibles à diviser, choisissez celui qui donne la plus grande réduction de SSE lorsqu'il est divisé. Remplacez ce cluster dans la liste par les deux sous-clusters résultants.
4. **Répétition** :
   — Répétez les étapes 2 et 3 jusqu'à ce que le nombre désiré de clusters soit atteint.

2.1.6.2 Avantages :

— Le bisecting k-means est souvent plus robuste que le k-means standard, car il est moins sensible aux initialisations.
— Il peut découvrir des clusters de forme plus complexe que le k-means standard.

2.1.6.3 Inconvénients :

— Il peut être plus lent que le k-means standard car il nécessite l'exécution de l'algorithme k-means plusieurs fois.

L'algorithme Bisecting K-Means a été introduit comme une méthode pour améliorer les performances et la qualité du clustering en évitant certaines des faiblesses du K-Means traditionnel. Bien qu'il soit plus lent, l'approche hiérarchique de la division des clusters en sous-clusters peut souvent conduire à des résultats de meilleure qualité (Steinbach *et al.*, 2000), (Zhao *et al.*, 2009).

2.1.7 **Regroupement automatique à l'aide de l'évolution différentielle (ACDE)**

Dans cet article (Das *et al.*, 2007), les auteurs présentent une méthode fondée sur l'évolution différentielle (DE), un type d'algorithme évolutionnaire. La DE opère en maintenant une population de S solutions potentielles au problème d'optimisation. Initialement, ces solutions sont générées de manière aléatoire suivant une distribution uniforme. L'algorithme procède ensuite par itérations successives, chacune comprenant trois phases : la reproduction (qui engendre une population temporaire), l'évaluation de la fonction de fitness pour chaque membre de cette population temporaire, et enfin la sélection des solutions les plus performantes pour la prochaine génération (Kwedlo, 2011).

Soit $u_{i,G}$ le ième membre $(i = 1, ..., S)$ de la population à la Gème itération. Habituellement, $u_{i,G}$ prend la forme d'un vecteur à valeurs réelles de dimension D, c'est-à-dire $u_{i,G} \in \Re$. La reproduction dans DE crée une population temporaire de vecteurs d'essai. Pour chaque solution ui,G, un vecteur d'essai correspondant yi,G est obtenu par les opérateurs de mutation et de croisement. L'opérateur de mutation génère un vecteur mutant selon l'équation :

$$y'_{i,G} = u_{a,G} + F(u_{b,G} - u_{c,G}) \tag{2.6}$$

Où $F \in [0, 1]$ est un paramètre fourni par l'utilisateur appelé facteur d'amplification et $a, b, c \in 1, ..., S$ sont choisis aléatoirement de telle sorte que $a \neq b \neq c \neq i$.

Le vecteur d'essai final $y_{i,G}$ est obtenu par l'opérateur de croisement, qui mélange le vecteur mutant avec le vecteur original $u_{i,G}$. Supposons que $u_{i,G} = (u_{1i,G}, u_{2i,G}, ..., u_{Di,G})$. avec le vecteur original $u_{i,G}$. Chaque élément $y_{ji,G}$ (où $j = 1, ..., D$) du vecteur d'essai $y_{i,G}$ est généré comme suit :

$$y_{ji,G} = \begin{cases} y'_{i,G} & \text{si } r(j) < CR \text{ ou } j = d \\ u_{i,G} & \text{sinon.} \end{cases} \tag{2.7}$$

Où, $CR \in [0, 1]$ représente un paramètre défini par l'utilisateur, nommé facteur de croisement. Le terme $r(j)$ correspond à un nombre aléatoire tiré d'une distribution uniforme sur l'intervalle $[0, 1]$, généré de manière indépendante pour chaque j. Quant à $d \in \{1, \ldots, D\}$, il s'agit d'un indice sélectionné aléatoirement qui assure qu'au moins un élément du vecteur de test $y_{i,G}$ est issu du vecteur mutant $y'_{i,G}$ (Kwedlo, 2011).

L'innovation introduite par les auteurs de cet article réside dans la réduction linéaire du taux de croisement (Cr) au fil du temps, avec Cr variant entre un maximum de $Cr_{\text{max}} = 1.0$ et un minimum de $Cr_{\text{min}} = 0.5$. Cela implique qu'au début, tous les éléments du vecteur parent sont susceptibles d'être remplacés par l'opérateur de différence vectorielle, en utilisant un indice de validité tel que l'indice de Davies-Bouldin. Toutefois, à mesure que le processus d'optimisation avance et que Cr diminue, de nouveaux éléments du vecteur parent sont conservés dans la descendance. Cette stratégie permet une exploration approfondie de l'espace de recherche au début, tout en affinant progressivement les déplacements des solutions d'essai

dans les phases ultérieures, facilitant ainsi l'exploration d'un espace plus restreint où la solution globale optimale est susceptible de se trouver. La variation de Cr en fonction du temps peut être décrit par l'équation suivante :

$$Cr = (Crmax - Crmin) \times \frac{Maxit - iter}{Maxit} \tag{2.8}$$

Où, Cr_{max} et Cr_{min} désignent les valeurs maximales et minimales du taux de croisement Cr, respectivement. Le terme $iter$ représente le numéro de l'itération en cours, tandis que $Maxit$ indique le nombre maximal d'itérations autorisées. Pour plus de détails sur la représentation chromosomique et le pseudo-code de l'algorithme ACDE, veuillez consulter l'article (Das *et al.*, 2007).

2.1.8 **Clustering dynamique avec Optimisation par essaims de particules (DCPSO)**

Le DCPSO est une variante de l'algorithme d'optimisation par essaims de particules (PSO) conçue pour identifier automatiquement le nombre "optimal" de clusters tout en regroupant les données avec une intervention minimale de l'utilisateur. L'algorithme initie le processus en divisant l'ensemble de données en un grand nombre de clusters pour atténuer l'impact des conditions de départ. Par la suite, grâce à l'optimisation par essaims de particules binaires, il sélectionne le nombre approprié de clusters. Finalement, les centres des clusters déterminés sont peaufinés à l'aide de l'algorithme K-means pour affiner le regroupement (Omran *et al.*, 2006).

L'optimisation par essaims de particules (PSO) a été introduite par Kennedy et Eberhart en 1995 (Kennedy et Eberhart, 1995). Comme le souligne l'article original, les sociobiologistes estiment qu'un banc de poissons ou une volée d'oiseaux en mouvement de groupe peut bénéficier de l'expérience collective de ses membres. Autrement dit, pendant qu'un oiseau cherche de la nourriture au hasard, l'ensemble du groupe peut partager ses découvertes et ainsi optimiser la recherche de nourriture. Bien que nous puissions simuler le comportement d'une volée d'oiseaux, nous pouvons aussi envisager que chaque oiseau contribue à la recherche de la solution optimale dans un espace de solutions multidimensionnel. La meilleure solution trouvée par le groupe est considérée comme la meilleure dans l'espace de recherche. Cependant, il est important de noter que la solution obtenue est heuristique, car il est impossible de prouver qu'elle est la véritable solution optimale (globale), ce qui n'est généralement pas le cas. Voici un aperçu de l'algorithme

DCPSO :

Initialisation :

- Définir le nombre de particules dans l'essaim.
- Initialiser chaque particule avec une position aléatoire dans l'espace de recherche. Chaque position représente une solution potentielle de clustering, c'est-à-dire un ensemble de centres de clusters.
- Initialiser la vitesse de chaque particule.
- Évaluer la qualité du regroupement de chaque particule en utilisant une fonction de fitness, comme la somme des distances intra-cluster.

Boucle principale :

1. Pour chaque particule :
   - Mettre à jour la vitesse de la particule en fonction de sa meilleure position passée (meilleure solution personnelle) et de la meilleure position globale trouvée par l'essaim (meilleure solution globale).
   - Mettre à jour la position de la particule en ajoutant la nouvelle vitesse à sa position actuelle.
   - Évaluer la nouvelle solution de clustering et mettre à jour la meilleure position personnelle si nécessaire.
2. Mettre à jour la meilleure position globale si une meilleure solution est trouvée par une particule.
3. Vérifier les critères d'arrêt, tels que le nombre maximal d'itérations ou la convergence vers une solution stable.

Post-traitement :

- Utiliser l'algorithme K-means pour affiner les centres de clusters obtenus par le DCPSO.
- Attribuer chaque point de données au cluster le plus proche.

Résultat :

- Retourner les centres de clusters finaux et les affectations de cluster pour chaque point de données.

2.1.9 **Amélioration du clustering k-means par la recherche d'organismes symbiotiques pour les problèmes de clustering automatique**

L'article (Ikotun et Ezugwu, 2022), propose une nouvelle méthode de *clustering* hybride combinant l'algorithme de recherche d'organismes symbiotiques (SOS) avec l'algorithme K-means. Cette méthode vise à améliorer le *clustering* automatique des données. La méthode K-means est généralement sensible aux

centres des clusters initiaux et nécessite la spécification du nombre de clusters, ce qui peut conduire à une convergence vers des optima locaux. L'approche hybride cherche à surmonter ces défis en utilisant SOS pour une recherche globale, afin de générer des centres de clusters optimaux initiaux pour K-means.

L'algorithme SOS est choisi pour sa capacité à opérer sans paramètres extensifs, offrant ainsi une recherche de qualité supérieure sans nécessiter un ajustement fin. L'efficacité de l'algorithme hybride proposé est évaluée à l'aide de plusieurs ensembles de données de l'UCI Machine Learning Repository et d'un ensemble de données artificiel, montrant des améliorations dans la performance de *clustering* par rapport à celle des méthodes standards K-means et d'autres approches hybrides existantes.

La recherche met en évidence que, bien que les méthodes métaheuristiques aient été traditionnellement utilisées pour aborder les défis du *clustering* automatique, elles nécessitent souvent des ajustements de paramètres rigoureux. L'approche hybride proposée ici réduit ce besoin tout en améliorant la qualité de la recherche et la stabilité des performances, facilitant ainsi la résolution de problèmes complexes de *clustering* de manière plus efficace. Pour plus de détails, veuillez consulter l'article suivant (Ikotun et Ezugwu, 2022).

2.1.10 Algorithmes métaheuristiques inspirés de la nature basés sur K-means pour les problèmes de clustering automatique de données :

L'article intitulé (Ikotun *et al.*, 2021), passe en revue les avancées récentes dans l'amélioration de l'algorithme de clustering K-means par le biais de techniques d'optimisation métaheuristiques inspirées de la nature pour résoudre des problèmes de clustering automatique de données. L'étude systématique présente les méthodes hybrides développées en combinant K-means avec divers algorithmes métaheuristiques, explorant leurs forces dans l'identification de clusters naturels dans les jeux de données sans information préalable sur les objets de données.

Le clustering automatique, qui ne nécessite pas la spécification du nombre de clusters par l'utilisateur, est mis en avant comme une approche efficace face aux limites de K-means, notamment la convergence vers des minima locaux et la sensibilité à l'initialisation des centres de clusters. Les algorithmes métaheuristiques, tels que les algorithmes génétiques, l'optimisation par essaims de particules et d'autres, sont explorés pour leur capacité à trouver des solutions optimales ou proches de l'optimal avec moins d'effort computationnel.

L'article souligne également les directions futures de recherche dans ce domaine, indiquant un besoin continu d'amélioration des techniques existantes et de développement de nouvelles approches pour le clustering automatique des données volumineuses et complexes d'aujourd'hui. Pour plus de détails, veuillez consulter l'article suivant (Ikotun *et al.*, 2021).

2.2 **Algorithmes métaheuristique :**

2.2.1 **Évolution Différentielle (ED) :**

L'Évolution Différentielle (ED) est une méthode d'optimisation numérique qui fait partie de la famille des algorithmes évolutionnaires. Elle est principalement utilisée pour optimiser des fonctions continues et est particulièrement efficace pour les problèmes d'optimisation globale. L'ED est simple à comprendre et à mettre en œuvre, mais elle est aussi très puissante. L'algorithme a été initialement proposé par (Storn et Price, 1997).

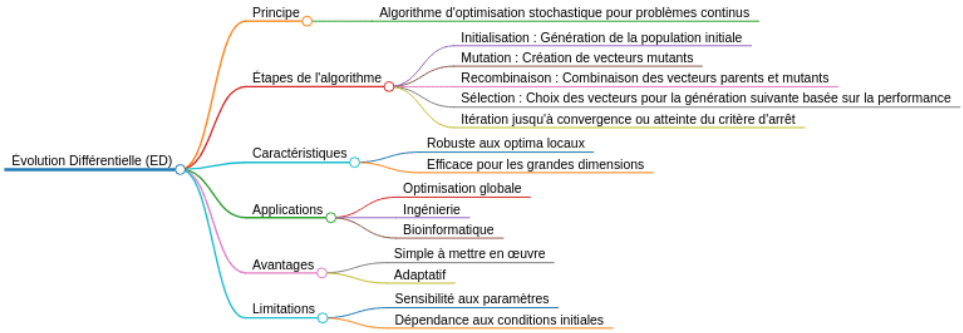

Figure 2.4 Diagramme qui récapitule l'algorithme ED

Voici une explication détaillée de l'algorithme Évolution Différentielle :

1. **Initialisation :**

- Choisissez les paramètres de contrôle :
  - NP (taille de la population)
  - F (facteur de différenciation, généralement entre [0.4, 1.0])
  - CR (probabilité de croisement, généralement entre [0.0, 1.0])
- Initialisez une population de NP vecteurs x dans l'espace de recherche. Chaque vecteur est une solution potentielle.

2. **Évaluation de la Fitness :**
   - Évaluez la fitness de chaque vecteur dans la population en utilisant la fonction objectif que vous cherchez à optimiser.
3. **Boucle générationnelle :** Répétez les étapes suivantes jusqu'à ce qu'une condition d'arrêt soit satisfaite (par exemple, un nombre maximum de générations, une tolérance sur la fitness, etc.) :
   (a) **Mutation :**
   - Pour chaque vecteur cible x_i de la population, sélectionnez trois vecteurs a, b et c distincts de x_i et de l'ensemble de la population.
   - Calculez le vecteur mutant v selon la formule :

$$v = a + F \cdot (b - c) \tag{2.9}$$

   (b) **Recombinaison (Croisement) :**
   - Pour chaque composant j du vecteur x_i et du vecteur mutant v, décidez si v_j doit remplacer x_{ij} dans le vecteur enfant u. Ceci est fait en tirant un nombre aléatoire r dans [0, 1] et en le comparant à CR. Si r est inférieur à CR, alors $u_j = v_j$, sinon $u_j = x_{ij}$.

   (c) **Sélection :**
   - Évaluez la fitness du vecteur enfant u en utilisant la fonction objectif.
   - Si la fitness de u est meilleure que celle de x_i, alors remplacez x_i par u dans la population de la prochaine génération.
4. **Solution :**
   - Après que la condition d'arrêt soit satisfaite, sélectionnez le vecteur dans la population actuelle ayant la meilleure fitness comme solution du problème d'optimisation (Lampinen *et al.*, 2005).

Notez que depuis sa proposition initiale, plusieurs variantes de l'Évolution Différentielle ont été développées pour améliorer la performance de l'algorithme dans différents types de problèmes d'optimisation. Ces variantes modifient souvent les stratégies de mutation ou de recombinaison.

2.2.2 **Simplicial Homology Global Optimization (SHGO)**

L'algorithme shgo (Simplicial Homology Global Optimization) est une méthode d'optimisation globale qui cherche à identifier l'ensemble de toutes les solutions optimales pour des problèmes d'optimisation sans contraintes ou avec contraintes. Cette méthode est basée sur les propriétés mathématiques des homologies simpliciales. L'avantage principal de shgo est qu'il fournit non seulement une estimation du minimum global, mais aussi des minima locaux et qu'il garantit la convergence vers ces solutions (Endres et Sandholm, 2008).

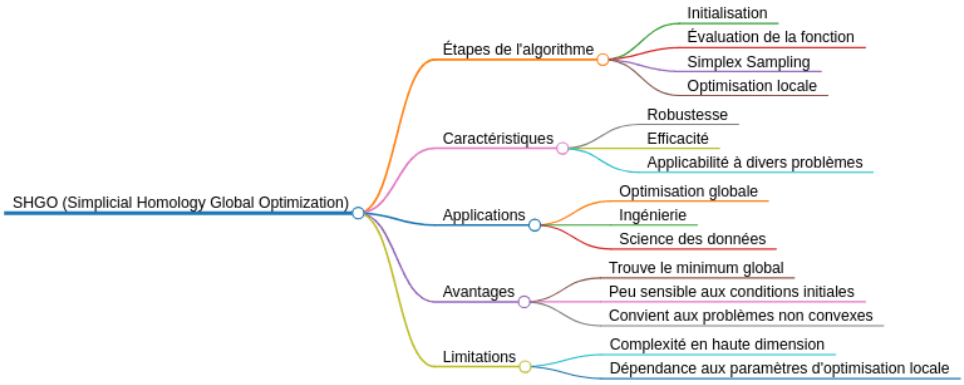

Figure 2.5 Diagramme qui récapitule l'algorithme SHGO

Voici une explication détaillée de l'algorithme SHGO :

1. **Homologie simpliciale :** SHGO utilise l'homologie simpliciale pour caractériser la topologie de l'espace de fonctionnement. Cela implique de décomposer le domaine en un ensemble de simplices (généralisation d'un triangle à des dimensions supérieures). L'homologie simpliciale est utilisée pour identifier et distinguer les caractéristiques topologiques, telles que les trous, des domaines de contrainte.
2. **Échantillonnage :** L'algorithme débute par un échantillonnage du domaine de contrainte. Cela peut être réalisé avec une méthode d'échantillonnage existante, telle que le Latin Hypercube Sampling (LHS).

3. **Localisation des minima locaux :** Une fois l'échantillonnage effectué, SHGO utilise ces échantillons pour localiser les régions qui contiennent potentiellement des minima locaux.
4. **Raffinement :** Après avoir identifié les régions d'intérêt, l'algorithme effectue un raffinement en utilisant des méthodes d'optimisation locales pour localiser précisément les minima locaux.
5. **Verification :** Une fois tous les minima locaux identifiés, SHGO compare les valeurs de la fonction objective pour chaque minimum local pour déterminer la solution globale (Endres et Sandgren, 2008), (Stillinger et Weber, 1985).

Le principal avantage de SHGO par rapport à d'autres méthodes d'optimisation globale est qu'il peut garantir la découverte de toutes les solutions globales et locales d'une fonction, étant donné que certaines conditions sont remplies.Notez que comme pour toute méthode d'optimisation globale, la performance de shgo dépend fortement de la nature du problème à résoudre. Dans certains cas, il peut être plus rapide et plus efficace que d'autres méthodes, tandis que dans d'autres, il peut nécessiter plus de temps ou d'évaluations de la fonction (Stillinger *et al.*, 1995), (Endres, 2010).

2.2.3 **DIRECT (DIvide a REctangle) :**

L'algorithme DIRECT (DIvide a REctangle) est une méthode déterministe d'optimisation globale pour des fonctions généralement définies sur des espaces bornés. Il a été introduit par Jones, Perttunen, et Stuckman en 1993. L'idée principale de DIRECT est de diviser systématiquement l'hyper-rectangle d'intérêt en sous-rectangles plus petits. À chaque itération, l'algorithme évalue la fonction objectif au centre de ces sous-rectangles, puis décide de les diviser ou non en fonction de certaines conditions (Jones *et al.*, 1993).

2.2.3.1 Procédure de base

1. **Initialisation** : Commencez par un hyperrectangle englobant tout l'espace de recherche.
2. **Évaluation** : Pour chaque hyperrectangle, évaluez la fonction objectif en son centre.
3. **Sélection** : Identifiez l'hyperrectangle qui a la plus petite valeur de fonction objectif.
4. **Division** : Divisez cet hyperrectangle en sous-rectangles plus petits. Le nombre et la taille de ces divisions dépendent de la dimensionnalité de l'espace de recherche.
5. **Itération** : Répétez les étapes 2 à 4 jusqu'à ce qu'un critère d'arrêt soit atteint (par exemple, un nombre maximum d'itérations, ou une tolérance de convergence).

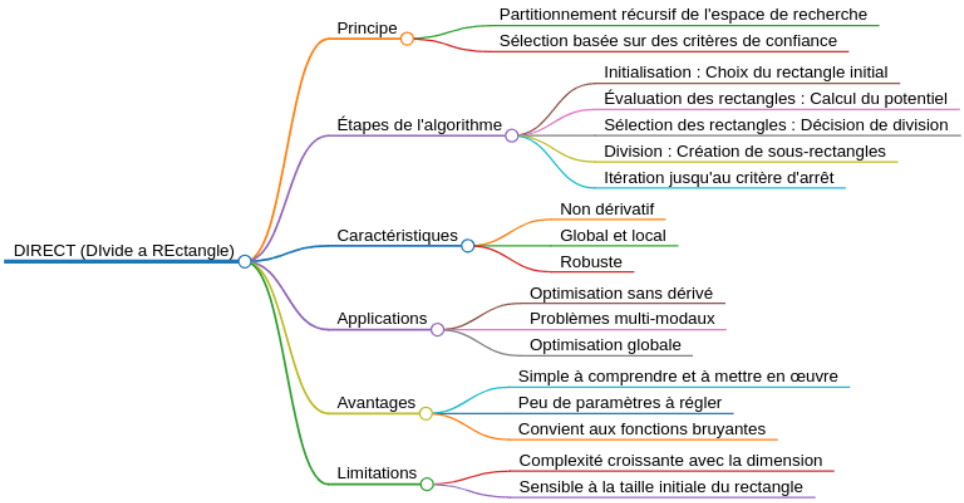

Figure 2.6 Diagramme qui récapitule l'algorithme DIRECT

2.2.3.2 Critères de division

L'un des aspects clés de l'algorithme DIRECT est de déterminer quels hyperrectangles diviser. La décision est généralement basée sur une comparaison entre la valeur de la fonction au centre de l'hyperrectangle et les valeurs sur ses bords. Si la valeur du centre est "proche" des valeurs des bords, cela suggère qu'il pourrait y avoir un minimum à l'intérieur de l'hyperrectangle, et il est donc divisé pour une investigation plus approfondie.

2.2.3.3 Avantages

- **Globalité** : DIRECT peut identifier des minima globaux, pas seulement locaux.
- **Pas de dérivées** : Convient aux fonctions black-box et non différenciables.
- **Déterministe** : Contrairement à de nombreuses autres méthodes d'optimisation globale, DIRECT est déterministe.

2.2.3.4 Inconvénients

— **Efficacité** : Sur des espaces de grande dimension, DIRECT peut nécessiter un grand nombre d'évaluations de fonction.

— **Simplexité** : Ne gère pas directement les contraintes (Jones *et al.*, 1993).

2.2.4 **Basin Hopping :**

L'algorithme de *Basin-hopping* (ou "saut de bassin") est une méthode stochastique utilisée pour trouver le minimum global d'une fonction mathématique qui présente plusieurs minima locaux. Il est souvent utilisé dans le contexte de l'optimisation des paysages énergétiques complexes, comme ceux rencontrés en chimie quantique ou en biologie moléculaire.

Le concept derrière le *Basin-hopping* est simple en théorie : à partir d'un point initial x_0, l'algorithme saute aléatoirement à un autre point, puis tente de trouver le minimum local à partir de ce nouveau point. Si ce nouveau minimum est plus bas que le précédent, le saut est accepté. Sinon, il peut être accepté avec une certaine probabilité ou rejeté (Leary et Doye, 2001).

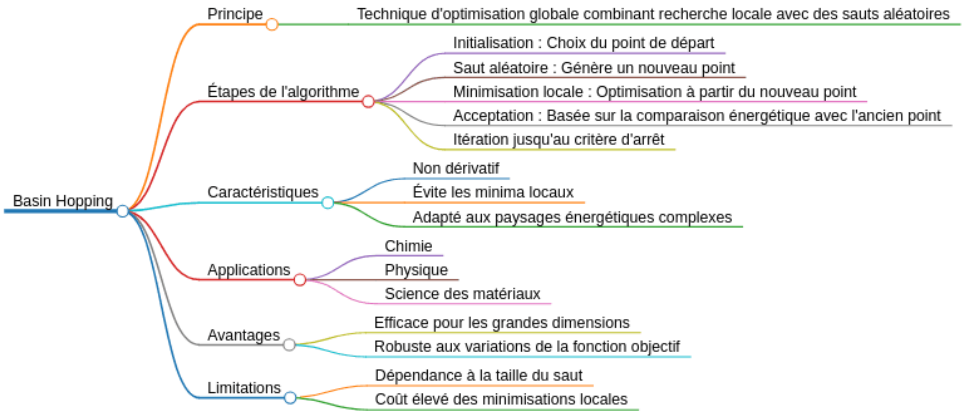

Figure 2.7 Diagramme qui récapitule l'algorithme Basin Hopping

Fonctionnement détaillé :

1. **Initialisation** : Choisir un point de départ x_0 et évaluer la fonction $f(x_0)$.
2. **Étape de saut** : Perturber le point actuel pour obtenir un nouveau point x' en utilisant une certaine distribution (par exemple, une distribution gaussienne autour du point actuel).
3. **Recherche locale** : À partir du point x', utiliser une méthode d'optimisation locale (comme la descente de gradient) pour trouver le minimum local x''.
4. **Critère d'acceptation** : Si $f(x'') < f(x_0)$, alors accepter le saut et définir $x_0 = x''$. Si $f(x'')$ est plus grand, alors accepter le saut avec une probabilité

$$e^{-\frac{(f(x'')-f(x_0))}{T}} \tag{2.10}$$

, Où T est une "température" qui décroît lentement au fil des itérations. Ce critère est similaire à celui utilisé dans le recuit simulé.

5. **Répétition** : Répéter les étapes 2-4 un certain nombre de fois ou jusqu'à ce qu'un critère d'arrêt soit satisfait (Wales et Doye, 1997).

2.2.4.1 Paramètres importants

— **Taille du saut** : Contrôle l'amplitude des sauts aléatoires.
— **Température fictive (T)** : Utilisée pour déterminer la probabilité d'accepter un saut vers un minimum de plus haute énergie.

2.2.4.2 Avantages :

— Capable de trouver des minima globaux là où les méthodes d'optimisation conventionnelles pourraient rester bloquées dans des minima locaux.
— Facile à implémenter avec n'importe quelle méthode d'optimisation locale.

2.2.4.3 Limitations :

— Il n'y a aucune garantie de trouver le minimum global.
— Le choix des paramètres (taille du saut, fonction d'acceptation, etc.) peut influencer considérablement les performances.

Le *Basin-hopping* combine des éléments d'exploration globale (à travers les sauts) et d'optimisation locale pour éviter de rester coincé dans des minima locaux (Wales et Doye, 1997).

2.2.5 **Sequential Least SQuares Programming (SLSQP)**

Le Sequential Least SQuares Programming (SLSQP) est un algorithme d'optimisation non linéaire qui peut être utilisé pour résoudre des problèmes d'optimisation avec ou sans contraintes. Il s'agit d'une méthode de type quasi-Newton, ce qui signifie qu'elle utilise des approximations de la matrice hessienne (ou de son inverse) pour effectuer ses itérations (Kraft, 1988).

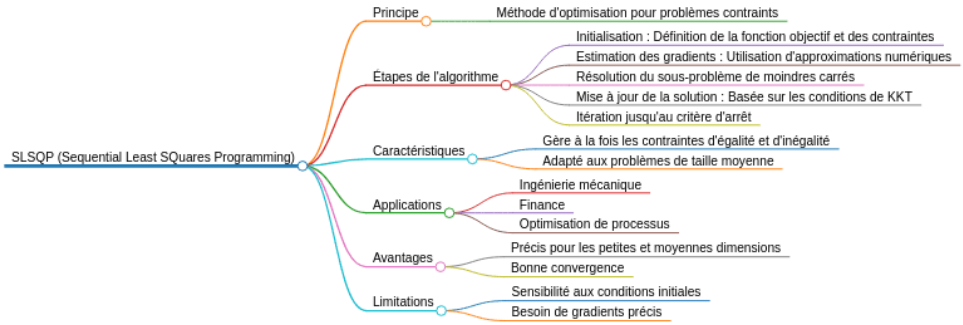

Figure 2.8 Diagramme qui récapitule l'algorithme SLSQP

2.2.5.1 Algorithme :

1. **Initialisation** : Commencez avec une estimation initiale de la solution, x_0, et définissez une approximation initiale de la matrice hessienne (ou de son inverse).
2. **Évaluation de la fonction et de ses dérivées** : À chaque étape, évaluez la fonction objectif, son gradient et, si nécessaire, son hessienne (ou une approximation de celle-ci).
3. **Résolution d'un sous-problème quadratique** : À partir des informations actuelles sur la fonction (et

possiblement ses contraintes), formulez et résolvez un sous-problème quadratique pour obtenir une direction de recherche.

4. **Mise à jour** : Mettez à jour la solution courante dans la direction trouvée à l'étape précédente. Cette mise à jour peut nécessiter une étape de recherche linéaire ou d'autres stratégies pour garantir la convergence.
5. **Mise à jour de l'approximation de l'hessienne** : Si vous n'utilisez pas l'hessienne exacte, mettez à jour l'approximation de l'hessienne (ou de son inverse) à l'aide d'une méthode appropriée (par exemple, la méthode BFGS).
6. **Vérification des critères d'arrêt** : Vérifiez si les critères d'arrêt sont satisfaits (par exemple, si le gradient est suffisamment proche de zéro ou si le changement de la fonction objectif est suffisamment petit).
7. **Répétition** : Si les critères d'arrêt ne sont pas satisfaits, revenez à l'étape 2.

2.2.5.2 Avantages et inconvénients :

Avantages :

— Capable de traiter des problèmes avec des contraintes d'égalité et d'inégalité.

— Ne nécessite pas la connaissance de l'hessienne exacte.

Inconvénients :

— Peut nécessiter de nombreuses évaluations de la fonction et de ses dérivées.

— L'efficacité dépend de la qualité des approximations de l'hessienne (Nocedal et Wright, 2006).

2.2.6 **Broyden–Fletcher–Goldfarb–Shanno (BFGS) :**

L'algorithme Broyden–Fletcher–Goldfarb–Shanno (BFGS) est une méthode itérative pour résoudre des problèmes d'optimisation sans contrainte. Elle appartient à la famille des méthodes quasi-Newton, qui sont conçues pour approximer l'inverse de la matrice hessienne d'une fonction. La particularité de BFGS est qu'elle met à jour cette approximation à chaque étape, plutôt que de recalculer l'hessienne entière (Broyden, 1970), (Fletcher, 1970).

L'idée principale derrière les méthodes quasi-Newton est de construire une séquence d'approximations de l'inverse de l'hessienne, tout en garantissant la convergence vers le minimum de la fonction (Goldfarb, 1970).

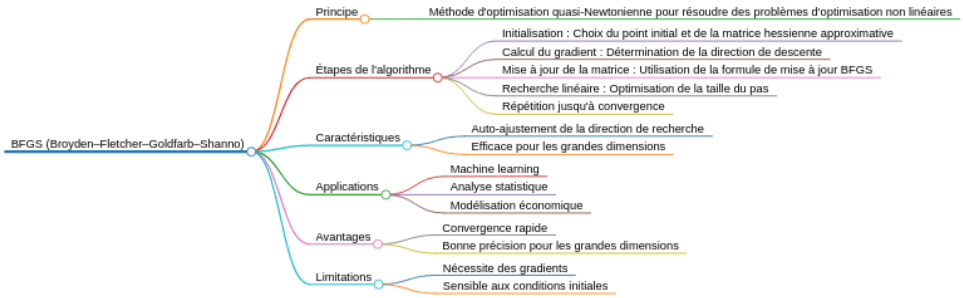

Figure 2.9 Diagramme qui récapitule l'algorithme BFGS

Explication de BFGS :

1. **Initialisation :** Choisissez un point de départ x_0 et une matrice initiale B_0, souvent la matrice identité. L'approximation initiale de l'hessienne inverse est donnée par cette matrice.
2. **Boucle d'optimisation :** Pour chaque itération k :

   (a) **Recherche linéaire :** Calculez une direction de descente p_k par la formule :

   $$p_k = -B_k \nabla f(x_k) \tag{2.11}$$

   Où ∇f est le gradient de la fonction objectif.

   (b) En utilisant cette direction, trouvez une taille de pas α_k qui réduit suffisamment la fonction selon un critère donné (comme la condition de Wolfe).

   (c) Mettez à jour le point courant :

   $$x_{k+1} = x_k + \alpha_k p_k \tag{2.12}$$

(d) Calculez la différence entre les points successifs et les gradients :

$$s_k = x_{k+1} - x_k \tag{2.13}$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \tag{2.14}$$

(e) **Mise à jour de la matrice :** Mettez à jour l'approximation de l'hessienne inverse en utilisant la formule BFGS :

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \tag{2.15}$$

3. **Critère d'arrêt :** Continuez la boucle d'optimisation jusqu'à ce qu'un critère d'arrêt soit satisfait, par exemple, quand la norme du gradient est suffisamment petite ou après un certain nombre d'itérations.

2.2.6.1 Avantages et inconvénients

Avantages :

— Plus rapide que la méthode de Newton pour de nombreuses fonctions, car elle ne nécessite pas l'évaluation de l'hessienne.

— Converge souvent en moins d'itérations que d'autres méthodes quasi-Newtoniennes.

Inconvénients :

— La convergence superlinéaire (mais pas quadratique comme la méthode de Newton) est obtenue sous certaines conditions.

— Peut nécessiter beaucoup de mémoire pour stocker les matrices, en particulier pour des problèmes à haute dimension (Shanno, 1970), (Nocedal et Wright, 2006).

2.2.7 **L'algorithme Nelder-Mead**

L'algorithme Nelder-Mead, souvent appelé "méthode du simplex", est une technique d'optimisation non linéaire qui est utilisée pour trouver un minimum ou un maximum d'une fonction dans un espace à plusieurs dimensions. Il est particulièrement populaire dans les cas où la fonction à optimiser est complexe, non différentiable ou non continue.

Principe de base :

L'algorithme utilise un simplex, qui est un polyèdre à $n+1$ sommets dans un espace n-dimensionnel. Par exemple, dans un espace 2D, un simplex serait un triangle, tandis que dans un espace 3D, il serait un tétraèdre.

L'algorithme procède par une série de mouvements du simplex pour explorer l'espace de la fonction. À chaque étape, il évalue la fonction à tous les sommets du simplex et essaie de remplacer le pire sommet (c'est-à-dire celui ayant la plus haute valeur pour un problème de minimisation) par un meilleur sommet (Nelder et Mead, 1965).

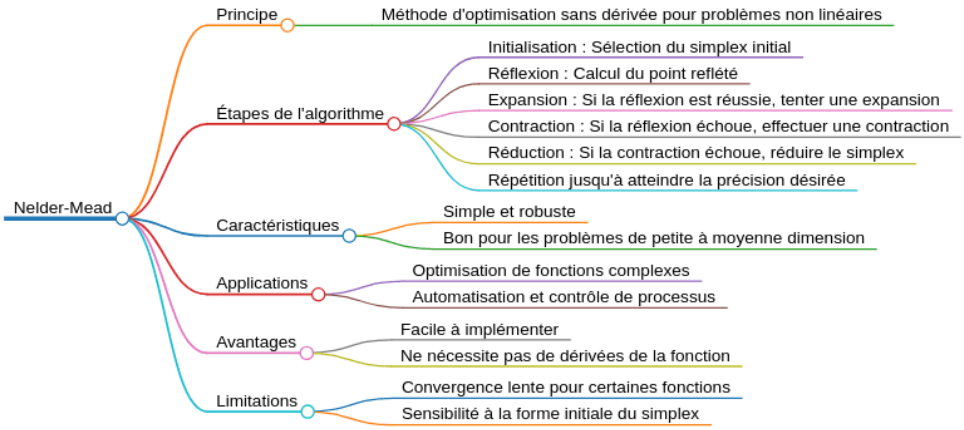

Figure 2.10 Diagramme qui récapitule l'algorithme Nelder-Mead

Étapes de l'algorithme :

1. **Initialisation :** Commencez avec un simplex initial de $n+1$ points.
2. **Évaluation :** Évaluez la fonction à optimiser en chaque point du simplex.
3. **Tri :** Triez les points du simplex en fonction de la valeur de la fonction.
4. **Transformation :** Remplacez le pire point par un nouveau point en utilisant l'une des méthodes suivantes :

- **Réflexion :** Calculez un point réfléchi en “reflétant” le pire point par rapport au centre de masse des autres points. Si ce point réfléchi est meilleur que le pire point actuel, mais pas mieux que les autres, remplacez le pire point par le point réfléchi.
- **Expansion :** Si le point réfléchi est le meilleur point jusqu’à présent, essayez de “dilater” le simplex dans cette direction pour voir si vous pouvez obtenir un point encore meilleur.
- **Contraction :** Si le point réfléchi est toujours pire que la plupart des autres points, essayez de “contracter” le simplex entre le pire point et le centre de masse.
- **Réduction :** Si la contraction ne fonctionne pas, réduisez la taille du simplex autour du meilleur point.

5. **Convergence :** Si le simplex satisfait un certain critère d’arrêt (par exemple, la variation des valeurs de la fonction est inférieure à un certain seuil), arrêtez l’algorithme. Sinon, retournez à l’étape 2 (Lagarias *et al.*, 1998).

2.2.8 L’algorithme COBYLA

L’algorithme COBYLA (Constrained Optimization BY Linear Approximations) est une méthode d’optimisation sans dérivée conçue pour résoudre des problèmes d’optimisation non linéaires avec des contraintes. Contrairement à de nombreuses autres méthodes, COBYLA n’a pas besoin d’évaluations de la fonction dérivée (ni de la fonction elle-même).

L’idée fondamentale de COBYLA est de construire une séquence d’approximations linéaires de la fonction objectif et des contraintes sur la base des évaluations précédentes de la fonction. Ces approximations linéaires sont ensuite utilisées pour définir un modèle d’optimisation linéaire, qui est résolu pour obtenir le prochain point d’évaluation.

2.2.8.1 Étapes de l’algorithme

1. **Initialisation** : On commence par choisir un point initial et définir un ensemble de points autour de ce point initial qui satisferaient toutes les contraintes.
2. **Modélisation** : Pour le point courant, une approximation affine est créée pour la fonction objectif et les contraintes en utilisant les évaluations précédentes.
3. **Optimisation** : On résout le problème d’optimisation linéaire construit à partir des approximations afin de déterminer le prochain point.

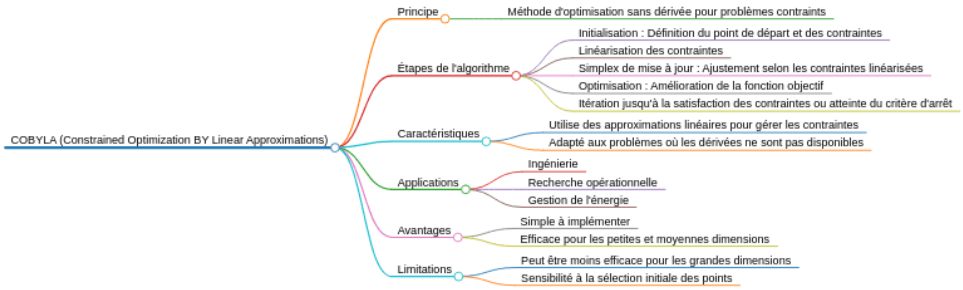

Figure 2.11 Diagramme qui récapitule l'algorithme COBYLA

4. **Mise à jour** : L'ensemble de points est mis à jour en fonction du nouveau point choisi.
5. **Convergence** : Si la différence entre les évaluations de la fonction objectif pour les points consécutifs est inférieure à un seuil prédéfini ou si un autre critère de convergence est satisfait, l'algorithme s'arrête.

2.2.8.2 Avantages

— Comme COBYLA est une méthode sans dérivée, elle est utile pour optimiser des fonctions pour lesquelles les dérivées ne sont pas disponibles ou sont difficiles à calculer.
— Elle est capable de gérer des fonctions non lisses.

2.2.8.3 Inconvénients

— Elle peut être moins efficace que les méthodes qui utilisent des dérivées lorsqu'elles sont disponibles.
— La convergence vers un minimum local est garantie seulement sous certaines conditions, comme c'est souvent le cas pour les algorithmes d'optimisation.

Il est à noter que, comme pour tout algorithme d'optimisation, la performance et la pertinence de COBYLA dépendront fortement de la nature du problème à résoudre. Il est donc recommandé de bien comprendre le problème en question et d'éventuellement essayer plusieurs méthodes pour trouver celle qui convient le mieux (Powell, 1998).

CHAPITRE 3
PROBLÉMATIQUE ET MÉTHODOLOGIE UTILISÉE

Le chapitre 3 de cette thèse se consacre à l'exploration approfondie de l'approche hybride développée pour résoudre le problème NP-difficile de détermination du nombre optimal de clusters dans les algorithmes de clustering, spécifiquement le k-means. Ce chapitre débute par la reformulation de la problématique, suivie par une description détaillée de l'environnement informatique utilisé pour la mise en œuvre des simulations. Nous présenterons ensuite les différentes sources de données utilisées, incluant à la fois les données synthétiques générées pour tester l'efficacité de l'approche proposée et les données réelles sur le cancer du sein, servant de cas d'étude pour évaluer l'applicabilité de l'algorithme dans des scénarios réels.

3.1 **Problématique :**

La problématique centrale de mon projet de recherche concerne l'identification du nombre optimal de clusters dans les algorithmes de clustering, tel que le k-means, un problème reconnu comme étant NP-difficile. L'approche proposée pour surmonter cette difficulté repose sur la conception d'une méthode hybride innovante qui intègre le k-means avec diverses techniques métaheuristiques telles que l'Évolution Différentielle, Direct, Shgo, Basin Hopping, SLSQP, BFGS, COBYLA et Nelder. Ces méthodes sont détaillées dans le chapitre deux de mon étude.

Pour évaluer l'efficacité de cette approche hybride, deux séries de simulations ont été entreprises. Initialement, j'ai généré des données synthétiques variées, ajustant le nombre de clusters de 2 à 100 et fixant la taille d'échantillon à 2000 points. Cette simulation a été évaluée à l'aide de trois indices de validation bien établis : Davies-Bouldin, Calinski-Harabasz, et Silhouette. Les résultats ont été comparés à ceux obtenus par la méthode de référence Elbow, permettant de mesurer les avancées de l'approche hybride. Dans une seconde simulation, j'ai conservé un nombre de clusters fixe à 10 tout en variant la taille de l'échantillon de 2000 à 11000 points, afin d'analyser l'impact de la taille de l'échantillon sur les performances.

En outre, une validation finale a été effectuée sur des données réelles concernant le cancer du sein, visant à tester la robustesse de l'approche hybride dans des scénarios d'application réels. L'objectif de ces expérimentations est de démontrer que l'approche hybride, combinant le k-means et les métaheuristiques, peut rivaliser la méthode Elbow en termes d'exactitude, de précision, de RMSE, de MAE et de temps d'exécution.

3.2 Environnement utilisé :

Dans le cadre de mon projet de fin d'étude, qui porte sur les problématiques de clustering et d'optimisation, j'ai choisi d'utiliser Google Colab Pro comme environnement de développement. Google Colab Pro offre l'avantage d'un accès à des ressources de calcul plus importantes, ce qui est particulièrement utile pour les tâches gourmandes en ressources comme le traitement de grands ensembles de données ou l'exécution d'algorithmes complexes.

Pour la programmation, j'ai opté pour le langage Python, reconnu pour sa simplicité et son efficacité dans le domaine de la science des données et de l'apprentissage automatique. Python offre une vaste bibliothèque de modules et de frameworks qui facilitent la manipulation des données et la mise en œuvre d'algorithmes d'apprentissage automatique.

Parmi les frameworks utilisés, scikit-learn s'est avéré être un outil incontournable pour le clustering. Sa simplicité d'utilisation et la diversité des algorithmes de clustering disponibles m'ont permis d'expérimenter et de comparer différentes approches. De plus, pour les problèmes d'optimisation, j'ai utilisé scipy.optimize, un module de SciPy spécialisé dans l'optimisation numérique. Ce module offre une gamme de méthodes d'optimisation qui m'ont aidé à affiner les paramètres de mes modèles et à améliorer leurs performances.

3.3 Présentation des données :

3.3.1 Génération des données synthétiques :

Dans le cadre de mon projet de fin d'études sur le clustering et les problèmes d'optimisation, j'ai utilisé à la fois des données synthétiques et des données réelles pour tester et évaluer mes algorithmes.

Pour générer les données synthétiques, j'ai utilisé la fonction `make_blobs` de la bibliothèque scikit-learn. Cette fonction permet de créer des groupes de points dans un espace multidimensionnel, ce qui est idéal pour simuler des scénarios de clustering. Les paramètres que j'ai choisi pour `make_blobs` sont les suivants :

— **Écart type** : J'ai utilisé deux configurations différentes pour l'écart type des clusters. Dans le premier cas, j'ai fixé l'écart type à 0,2 pour créer des clusters bien définis et peu dispersés. Dans le second cas, j'ai augmenté l'écart type à 0,8 pour créer des clusters plus dispersés et donc plus difficiles à séparer.

— **Intervalle de sélection** : Les coordonnées des centres des clusters ont été choisies aléatoirement dans un intervalle de [-10, 10] dans toutes les dimensions.
— **Taille de la population** : J'ai utilisé deux configurations pour la taille de la population. Dans le premier cas, j'ai fixé la taille à 2000 points. Dans le second cas, j'ai varié la taille de la population entre 2000 et 11 000 points pour tester l'efficacité des algorithmes sur des ensembles de données de tailles différentes.
— **Nombre de clusters** : Le nombre de clusters a également été varié. Dans le premier cas, le nombre de clusters était compris entre 2 et 100, afin d'observer l'efficacité des algorithmes sur un large éventail de configurations. Dans le deuxième cas, j'ai fixé le nombre de clusters à 10 pour une comparaison plus ciblée.

3.3.2 **Données réelles sur le cancer du sein :**

En plus des données synthétiques, j'ai également utilisé le jeu de données `load_breast_cancer` de scikit-learn. Ce jeu de données contient des mesures issues d'images numérisées de tumeurs mammaires, avec des caractéristiques telles que le rayon, la texture, la périmétrie et l'aire des cellules tumorales. Les données sont étiquetées en fonction de la malignité de la tumeur, ce qui en fait un ensemble de données précieux pour tester des algorithmes de classification et de clustering dans un contexte médical réel.

L'utilisation de ces deux types de données m'a permis d'évaluer la robustesse et la polyvalence de mes algorithmes dans divers scénarios, allant de situations contrôlées avec des données synthétiques à des cas plus complexes et réalistes avec des données réelles.

3.4 **L'algorithme hybride combinant k-means et les métaheuristiques :**

Ce pseudo-code décrit une fonction qui choisit un indice de validation, calcule le coût associé à l'utilisation de k-means pour un nombre donné de clusters, et retourne soit le maximum soit le minimum de ce coût en fonction de l'indice sélectionné.

Algorithm 3 Algorithme de la fonction objective

Fonction objective (nb-clusters) :

Choisir l'indice de validation parmi Davies_Bouldin, Calinski_Harabasz, Silhouette

if l'indice de validation est Calinski_Harabasz ou Silhouette **then**

Appliquer le k-means (nb-clusters)

Cout = calculer le coût de k-means (nb-clusters) avec Calinski_Harabasz ou Silhouette

return maximum de Cout

else

return minimum de Cout

end if

3.4.1 Simulation d'algorithmes hybrides sur des données synthétiques avec une variation du nombre de clusters de 2 à 100 :

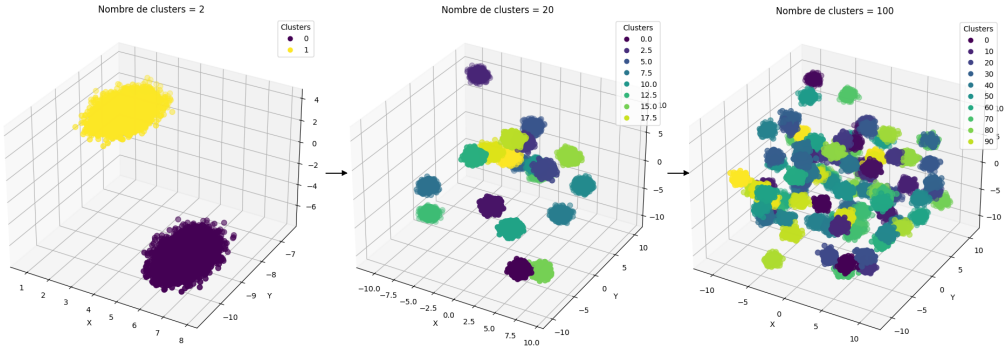

Figure 3.1 Variation du nombre de clusters de 2 à 100 clusters.

Algorithm 4 Optimisation de clustering utilisant des métaheuristiques

Input : Taille de la population = 2000, Espace de solutions = [2, 100], Liste-Métaheuristique = {ED, Direct, Shgo, BH (SLSQP), BH (BFGS), BH (Nelder), BH (COBYLA)}

Output : Nombre de cluster optimal, Exactitude, Précision, RMSE, MAE

```
while itération < 50 do
    for nb-centre = 2 to 100 do
        Générer des clusters (nb-centre, Taille de la population)
        Résultat_Elbow = Elbow & K-means (Fonction objective (nb-centre), Espace de solutions)
        Résultat_Algorithme_Hybride = Métaheuristique & K-means (Fonction objective (nb-centre), Espace de solutions)
    end for
end while
Exactitude = Calculer Exactitude (Résultat_Algorithme_Hybride)
Précision = Calculer Précision (Résultat_Algorithme_Hybride)
RMSE = Calculer RMSE (Résultat_Algorithme_Hybride)
MAE = Calculer MAE (Résultat_Algorithme_Hybride)
Temps d'exécution = temps d'exécution (Résultat_Algorithme_Hybride, Elbow)
return Exactitude, Précision, RMSE, MAE, Temps d'exécution
```

L'algorithme 04, commence par générer des données synthétiques organisées en clusters, en fixant la taille de l'échantillon à 2000 points. À chaque itération, il varie le nombre de clusters, de 2 à 100, afin de tester diverses configurations. Cette génération initiale sert de base pour simuler la progression d'algorithmes hybrides qui combinent k-means avec différentes méthodes métaheuristiques. L'objectif est de déterminer le nombre optimal de clusters pour chaque formation de clusters. Pour évaluer l'efficacité de ces approches hybrides, plusieurs indices de validation des clusters sont utilisés, notamment Davies-Bouldin, Calinski-Harabasz et Silhouette, permettant ainsi une comparaison rigoureuse des performances. En conclusion, l'algorithme fournit un retour détaillé sur les performances des méthodes testées, mesurant l'exactitude, la précision, le RMSE (Root Mean Square Error), le MAE (Mean Absolute Error) ainsi que le temps d'exécution pour chaque configuration testée. Cette approche exhaustive permet de cerner avec précision la méthode la plus efficace dans divers scénarios de clustering.

3.4.2 Simulation d'algorithmes hybrides sur des données synthétiques avec une variation de la taille d'échantillon de 2000 à 11000 points :

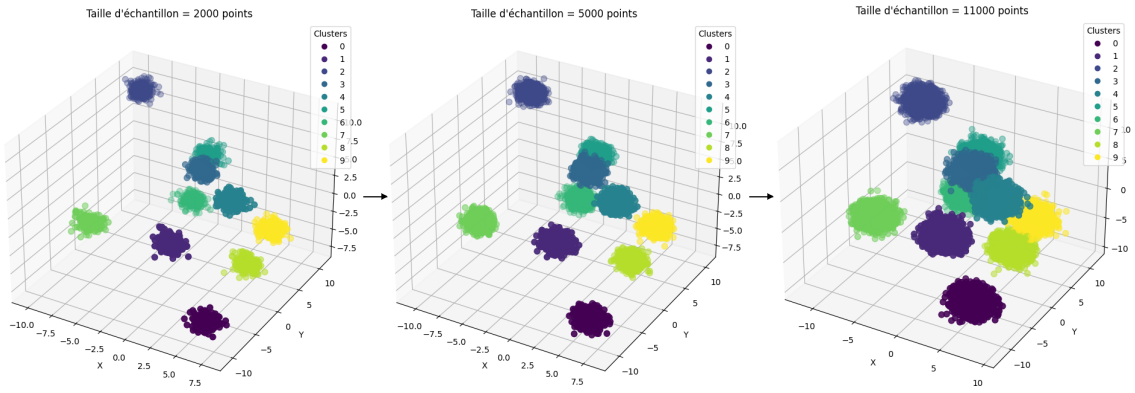

Figure 3.2 Variation de la taille d'échantillon de 2000 à 11000 points

Algorithm 5 Simulation des algorithmes hybrides k-means et métaheuristique

Input : Nombre de clusters = 10, Espace de solutions = [2, 100], Liste-Métaheuristique = {ED, Direct, Shgo, BH (SLSQP), BH (BFGS), BH (Nelder), BH (COBYLA)}

Output : Nombre de cluster optimal, Exactitude, Précision, RMSE, MAE

while itération < 50 **do**

for Taille d'échantillon = 2000 to 11000 **do**

Générer des clusters (nb-centre, Taille de la population)

Résultat_Elbow = Elbow & K-means (Fonction objective (nb-centre), Espace de solutions)

Résultat_Algorithme_Hybride = Métaheuristique & K-means (Fonction objective (nb-centre), Espace de solutions)

end for

end while

Exactitude = Calculer Exactitude (Résultat_Algorithme_Hybride)

Précision = Calculer Précision (Résultat_Algorithme_Hybride)

RMSE = Calculer RMSE (Résultat_Algorithme_Hybride)

MAE = Calculer MAE (Résultat_Algorithme_Hybride)

Temps d'exécution = temps d'exécution (Résultat_Algorithme_Hybride, Elbow)

return Exactitude, Précision, RMSE, MAE, Temps d'exécution

L'algorithme 05, commence par la génération de données synthétiques, structurées en clusters avec un nombre fixé à 10 clusters initiaux. À chaque itération, il ajuste la taille de l'échantillon, variant de 2000

à 11000 points, afin d'analyser l'impact de la taille de la population sur la détermination du nombre optimal de clusters. Cette procédure permet de simuler et d'évaluer la performance d'algorithmes hybrides combinant le k-means avec diverses méthodes métaheuristiques dans des contextes variés de volume de données. Pour assurer une évaluation rigoureuse, plusieurs indices de validation de clusters sont utilisés, notamment Davies-Bouldin, Calinski-Harabasz et Silhouette. Ces indices permettent de mesurer la qualité des clusters formés sous différentes configurations de taille d'échantillon. En conclusion de chaque cycle de simulation, l'algorithme compile et retourne les performances obtenues selon différentes métriques telles que l'exactitude, la précision, le RMSE (Root Mean Square Error), le MAE (Mean Absolute Error) et le temps d'exécution. Ces mesures fournissent une compréhension approfondie de l'efficacité des approches hybrides en fonction des variations dans la taille des données traitées.

3.4.3 **Simulation d'algorithmes hybrides sur des données réel du cancer :**

Algorithm 6 Optimisation de Clustering sur les données du cancer du sein

Input : données load-breast-cancer, Liste-Métaheuristique = {ED, Direct, Shgo, BH (SLSQP), BH (BFGS), BH (Nelder), BH (COBYLA)}

Output : Nombre de cluster optimal, Exactitude, Précision, RMSE, MAE, Temps d'exécution

while itération < 50 **do**

Résultat_Elbow = Elbow & K-means (Fonction objective , Espace de solutions)

Résultat_Algorithme_Hybride = Métaheuristique & K-means (Fonction objective, Espace de solutions)

end while

Exactitude = Calculer Exactitude (Résultat_Algorithme_Hybride)

Précision = Calculer Précision (Résultat_Algorithme_Hybride)

RMSE = Calculer RMSE (Résultat_Algorithme_Hybride)

MAE = Calculer MAE (Résultat_Algorithme_Hybride)

Temps d'exécution = Calculer Temps d'exécution (Résultat_Algorithme_Hybride, Elbow)

return Exactitude, Précision, RMSE, MAE, Temps d'exécution

L'algorithme 06, réalise une simulation pour évaluer l'efficacité de méthodes hybrides combinant l'algorithme k-means et diverses approches métaheuristiques, spécifiquement conçu pour identifier le nombre optimal de clusters dans un jeu de données réel, en l'occurrence les données load-breast-cancer de la bibliothèque scikit-learn. Cet ensemble de données, qui contient des informations sur des cas de cancer du sein, est utilisé pour tester l'adaptabilité et l'efficacité des algorithmes face à des données réelles plutôt que synthétiques. Les méthodes évaluent la qualité des clusters formés en utilisant différents indices de

validation, tels que Davies-Bouldin, Calinski-Harabasz, et Silhouette, qui fournissent une mesure comparative de la compactness et de la séparation des clusters. À la fin de chaque simulation, l'algorithme calcule et retourne plusieurs métriques de performance, incluant l'exactitude, la précision, l'erreur quadratique moyenne (RMSE), l'erreur absolue moyenne (MAE) et le temps d'exécution. Ces résultats permettent d'analyser de manière approfondie la performance des méthodes hybrides dans le contexte de données réelles, offrant ainsi des insights précieux sur leur efficacité et leur applicabilité.

CHAPITRE 4
RÉSULTATS ET DISCUSSIONS

Le chapitre 4 constitue le cœur analytique de cette thèse, où sont présentés et examinés en détail les résultats obtenus à partir des différentes simulations menées avec notre algorithme hybride. Ce chapitre commence par une présentation des résultats issus des données synthétiques, où le nombre de clusters a varié de 2 à 100. Nous analyserons l'évolution de l'exactitude, de la racine de l'erreur quadratique moyenne (RMSE), ainsi que du temps d'exécution pour ces configurations. Nous explorons également les résultats obtenus sur des échantillons de tailles variant de 2000 à 11000 points, en mettant un accent particulier sur l'évolution de la précision et du RMSE, ainsi que sur les variations du temps d'exécution associées à chaque taille d'échantillon.

La seconde partie de ce chapitre est dédiée à l'analyse des résultats obtenus sur les données réelles de cancer, avec un focus spécifique sur les performances des indices de Calinski-Harabasz et de Silhouette. Une discussion approfondie des résultats obtenus, y compris une interprétation des mesures obtenues avec l'indice Davies-Bouldin sur les données synthétiques, permet de mettre en perspective l'efficacité de l'approche hybride. En outre, une comparaison avec des travaux antérieurs similaires est envisagée pour contextualiser et évaluer les progrès réalisés par cette recherche.

En conclusion, ce chapitre vise à fournir une analyse exhaustive et critique des performances de l'algorithme, en discutant des implications des résultats obtenus tant sur les données synthétiques que réelles. Cette analyse détaillée aidera à comprendre les avantages, les limites et les potentialités de l'approche hybride proposée pour le clustering optimal.

4.1 **Présentation des résultats :**

4.1.0.1 **Remarque :**

Dans notre projet, nous avons exploré diverses méthodes de validation de clustering, dont l'indice de validité silhouette. Cependant, cet indice n'est pas compatible avec tous les algorithmes d'optimisation. En particulier, l'algorithme Nelder-Mead et l'algorithme COBYLA (Constrained Optimization BY Linear Approximations) présentent des caractéristiques qui ne s'alignent pas bien avec les critères évalués par l'indice de

silhouette. L'algorithme Nelder-Mead, par exemple, est un algorithme de recherche directe qui ne repose pas sur des gradients, ce qui peut entraîner des difficultés dans l'évaluation de la qualité des clusters selon les mesures de silhouette. De même, COBYLA, en tant qu'algorithme d'optimisation contrainte, peut aboutir à des solutions qui ne sont pas idéales du point de vue de la silhouette en raison de ses contraintes spécifiques. Par conséquent, pour éviter des comparaisons inappropriées et potentiellement trompeuses, nous avons choisi d'exclure l'algorithme Nelder-Mead et l'algorithme COBYLA de notre comparaison des algorithmes de clustering.

4.1.1 **Présentation des résultats obtenus sur des données synthétiques avec variation du nombre de clusters de 2 à 100 clusters :**

4.1.1.1 **Évolution de l'exactitude (figure 4.1) :**

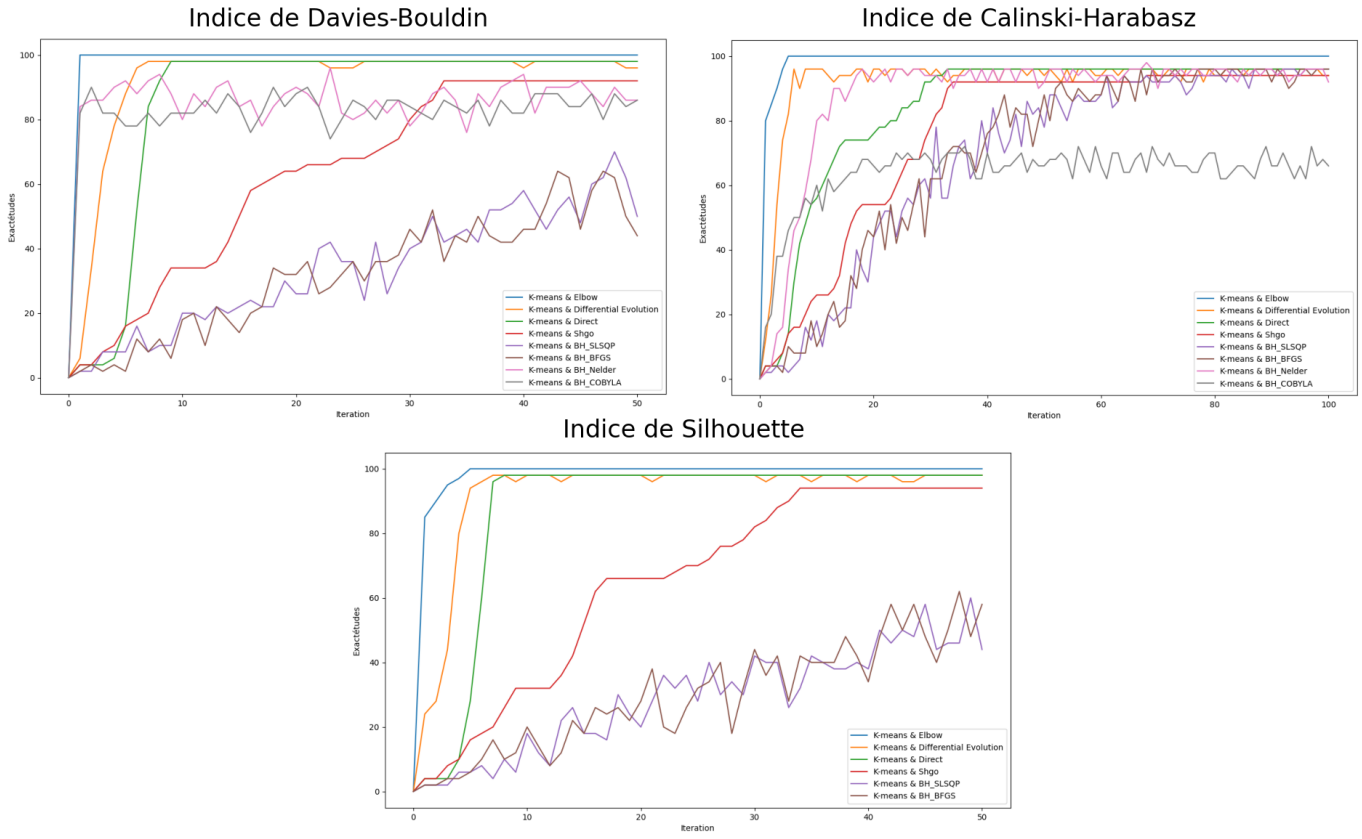

Figure 4.1 Progression en exactitude des algorithmes en utilisant les indices de validation Davies-Bouldin, Calinski-Harabasz et Silhouette, avec variation du nombre de clusters de 2 a 100 clusters.

Voici une analyse détaillée de l'évolution des différentes méthodes d'hybridation de K-means avec les métaheuristiques, en utilisant les indices de validation de clustering Davies-Bouldin, Calinski-Harabasz et Silhouette, présenter sur la figure 4.1.

Premier Graphique (Davies-Bouldin) :

- **K-means & Elbow (Bleu)** : Sert de référence avec une convergence rapide et une performance stable. Cela suggère une forte adéquation des clusters avec les données.
- **K-means & Direct (Vert)** : Présente une montée rapide indiquant une convergence initiale très efficace, et maintient une exactitude élevée, ce qui suggère que cette méthode est bien adaptée pour cet ensemble de données.
- **K-means & Differential Evolution (Orange)** : Montre également de bonnes performances, similaires à la méthode Direct (Vert), bien qu'avec un peu plus de variabilité.
- **K-means & Nelder (Rose) et K-means & BH-COBYLA (Gris)** : Bien que les performances de ces méthodes ne soient pas aussi élevées que celles obtenues avec des méthodes telles que l'Évolution Différentielle ou Direct, les résultats restent intéressants en termes d'exactitude, ce qui indique que les perspectives sont prometteuses.
- **Les autres méthodes (Rouge, Violet, Marron)** : Ces courbes montrent une plus grande variabilité et parfois une convergence plus lente, indiquant une performance inférieure sur cet indice par rapport aux méthodes de référence.

Deuxième Graphique (Calinski-Harabasz) :

- **K-means & Elbow (Bleu)** : La méthode de référence demeure performante sur ce critère également.
- **K-means & Differential Evolution (Orange) et K-means & Direct (Vert)** : Continuent d'afficher de fortes performances, bien qu'elles n'atteignent pas tout à fait les mêmes niveaux d'exactitude que la méthode de référence, mais on observe une baisse des performances de la méthode Évolution Différentielle lorsqu'elle est évaluée avec l'indice Calinski-Harabasz, ce qui indique qu'elle est moins performante avec cet indice.

  Les variations sont plus prononcées dans ce graphique, mais il est à noter que la tendance générale reste similaire au premier graphique.
- **K-means & BH-COBYLA (Gris)** : Les performances de cette méthode ont diminué lors de l'utilisation

de l'indice de validation Calinski-Harabasz par rapport à l'indice Davies-Bouldin, ce qui suggère que cette méthode est moins efficace avec cet indice.

— **K-means & BH-SLSQP (Violet), K-means & BH-BFGS (Marron) et K-means & Nelder (Rose)** : La performance de ces méthodes a considérablement augmenté, bien qu'elles ne soient pas encore à la hauteur de la méthode de référence. Cependant, leur progression est continue et prometteuse, ce qui suggère qu'elles sont mieux adaptées à l'indice Calinski-Harabasz.

Troisième Graphique (Silhouette) :

— **K-means & Elbow (Bleu)** : Conserve sa position de leader avec une exactitude élevée tout au long des itérations.

— **K-means & Differential Evolution (Orange) et K-means & Direct (Vert)** : Ces méthodes affichent des résultats compétitifs avec la méthode de référence, bien que la performance de Differential Evolution semble légèrement meilleure.

Les méthodes alternatives montrent une fois de plus une plus grande variabilité et des pics moins élevés, suggérant une adéquation moins optimale.

— **Observations générales :**

K-means & Differential Evolution et K-means & Direct semblent être les concurrents les plus forts par rapport à K-means & Elbow. Ils offrent une bonne exactitude et une convergence rapide selon vos indicateurs.

Les autres méthodes montrent une plus grande variabilité, qui pourrait être due à une multitude de facteurs, tels que la sensibilité aux conditions initiales ou une adaptation moins idéale aux spécificités des données synthétiques utilisées.

— **Conclusion :**

L'analyse visuelle indique que l'hybridation de K-means avec Differential Evolution et Direct pourraient offrir une alternative viable à K-means & Elbow, surtout si l'on cherche à réduire le temps de calcul tout en conservant une exactitude élevée. Cependant, une évaluation quantitative est nécessaire pour confirmer ces observations préliminaires.

4.1.1.2 **Évolution de la racine de l'erreur quadratique moyenne RMSE (Figure 4.2) :**

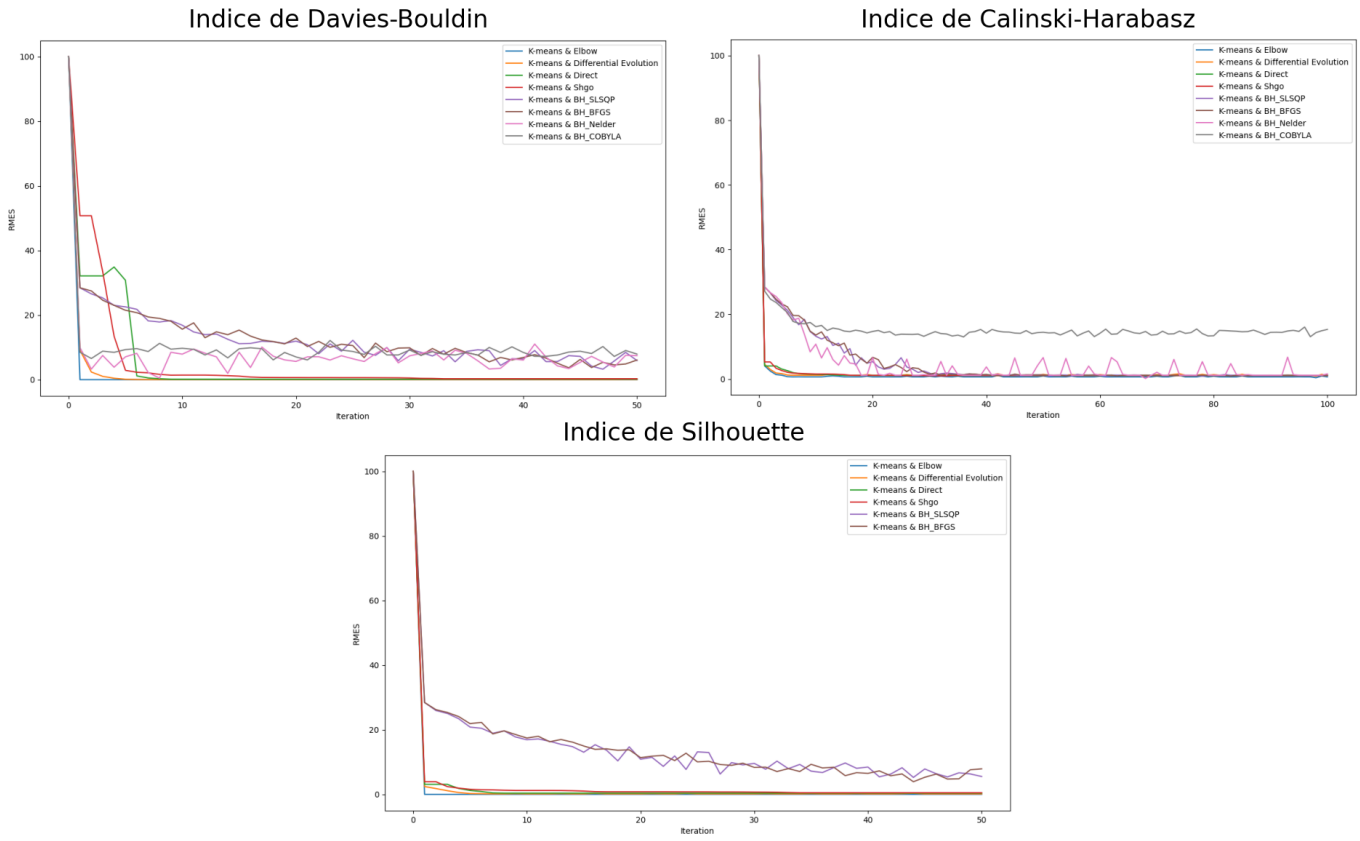

Figure 4.2 Progression de la racine de l'erreur quadratique moyenne RMSE des algorithmes en utilisant les indices de validation Davies-Bouldin, Calinski-Harabasz et Silhouette, avec variation du nombre de clusters de 2 a 100 clusters.

Nous analysons l'efficacité des différentes méthodes hybrides de K-means avec les métaheuristiques en termes de racine de l'erreur quadratique moyenne (RMSE). Les graphiques sont évalués en fonction des indices de validation de clustering Davies-Bouldin, Calinski-Harabasz, et Silhouette.

Premier Graphique (Davies-Bouldin) :

- **K-means & Elbow (Bleu) :** Montre une décroissance rapide du RMSE, indiquant une convergence efficace dès les premières itérations.
- **K-means & Differential Evolution (Orange) :** Présente une réduction rapide du RMSE, suggérant que l'hybridation avec cette métaheuristique est très compétitive.

— **K-means & Direct (Vert) :** La méthode montre également une réduction rapide et constante du RMSE.

Les autres méthodes montrent une plus grande variabilité dans la réduction du RMSE.

Deuxième Graphique (Calinski-Harabasz) :

— **K-means & Elbow (Bleu) :** Conserve un RMSE bas à travers les itérations.
— **K-means & Differential Evolution (Orange)** et **K-means & Direct (Vert) :** Suivent de près l'algorithme de référence en performance.

Toutes les méthodes convergent vers un RMSE faible, ce qui peut indiquer l'adéquation de la métrique Calinski-Harabasz avec ces approches hybrides.

Troisième Graphique (Silhouette) :

— **K-means & Elbow (Bleu) :** Affiche une excellente performance avec un RMSE qui stabilise rapidement.
— **K-means & Differential Evolution (Orange)** et **K-means & Direct (Vert) :** Ces méthodes sont très compétitives avec l'algorithme de référence.

Conclusion :

Les méthodes hybrides **K-means & Differential Evolution** et **K-means & Direct** semblent particulièrement prometteuses, montrant une convergence rapide et un RMSE bas. Elles pourraient représenter des alternatives viables à **K-means & Elbow**, notamment pour des données synthétiques. Une évaluation quantitative supplémentaire est nécessaire pour confirmer ces observations préliminaires.

4.1.1.3 **Évolution du temps d'exécution (Figure 4.3) :**

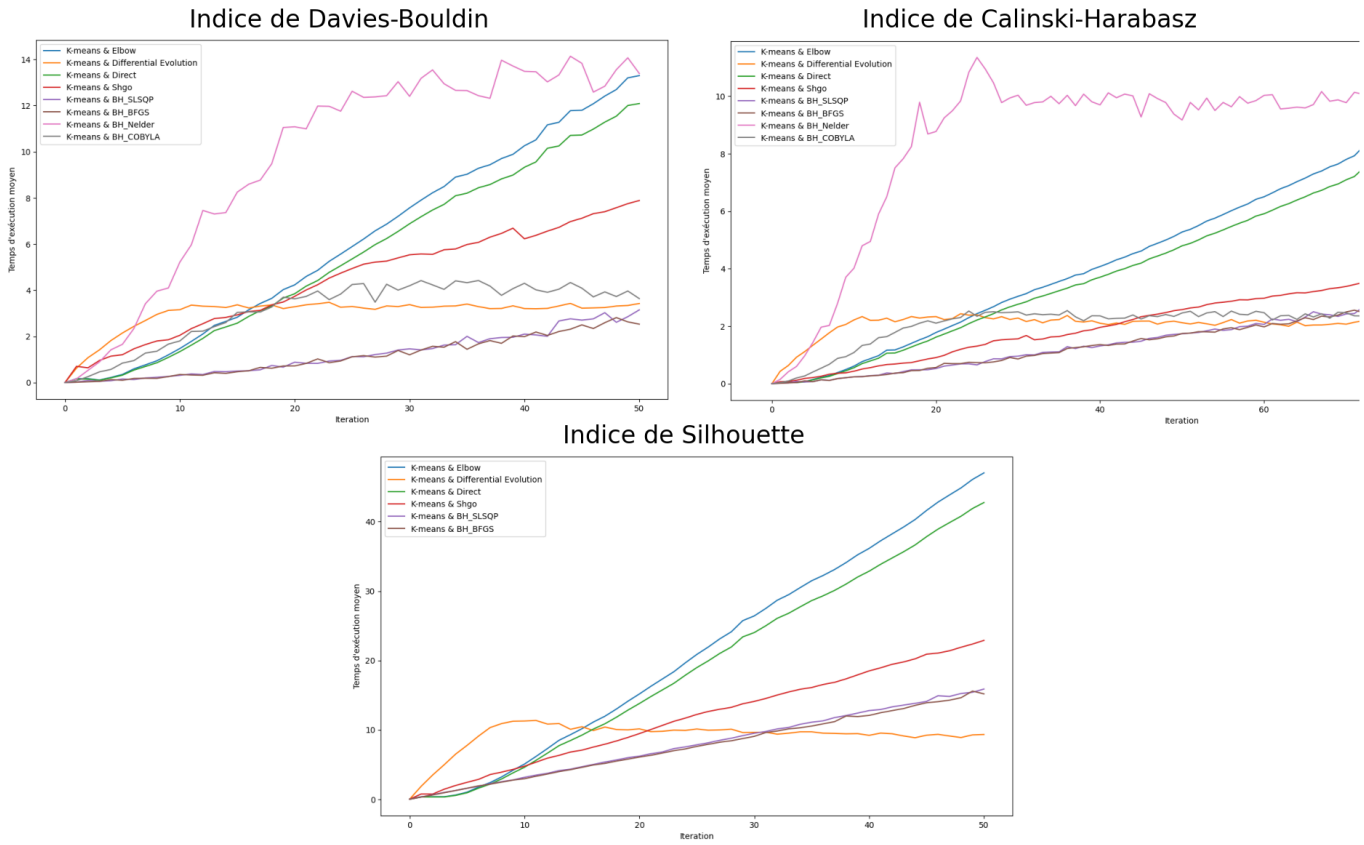

Figure 4.3 Progression du temps d'exécution des algorithmes en utilisant les indices de validation Davies-Bouldin, Calinski-Harabasz et Silhouette, avec variation du nombre de clusters de 2 a 100 clusters.

Nous abordons l'analyse de l'évolution du temps d'exécution pour différentes hybridations de K-means avec des métaheuristiques, en relation avec divers indices de validation.

Premier Graphique (Davies-Bouldin) :

- **K-means & Elbow (Bleu) :** Montre une progression lente du temps d'exécution, ce qui est attendu pour la méthode de référence.
- **K-means & Differential Evolution (Orange) et K-means & BH-COBYLA :** Le temps d'exécution est initialement bas, mais augmente rapidement au fil des itérations avant de commencer à diminuer progressivement.
- **K-means & Direct (Vert) :** Présente une augmentation modérée et constante du temps d'exécution,

suggérant une performance temporelle efficace.Toutefois, il reste supérieur par rapport aux autres méthodes.

- **K-means & BH_Nelder (Rose) :** Démontre un temps d'exécution plus élevé, signifiant potentiellement une performance inférieure pour cet indice.
- **K-means & BH_SLSQP (Violet) et K-means & BH_BFGS (Marron) :** Ces méthodes affichent un temps d'exécution inférieur par rapport aux autres méthodes, ce qui les rend très attrayantes pour réduire le temps d'exécution. Il reste néanmoins à confirmer leur efficacité selon d'autres métriques telles que l'exactitude, la précision et le RMSE.

Deuxième Graphique (Calinski-Harabasz) :

On constate que, en général, le temps d'exécution des méthodes a légèrement diminué par rapport à l'utilisation de l'indice Davies-Bouldin, ce qui suggère que l'indice Calinski-Harabasz est plus rapide. Les tendances sont similaires au premier graphique, avec K-means & BH_Nelder affichant toujours un temps d'exécution supérieur aux autres méthodes.

Troisième Graphique (Silhouette) :

On observe que le temps d'exécution de toutes les méthodes a augmenté de manière exponentielle, ce qui suggère que cet indice est particulièrement exigeant en termes de temps d'exécution par rapport aux indices Davies-Bouldin et Calinski-Harabasz. Il reste à évaluer son efficacité sur d'autres métriques telles que l'exactitude, la précision et le RMSE, mais la tendance générale reste identique.

Conclusion En général, l'hybridation de K-means avec des métaheuristiques réduites le temps d'exécution par rapport à K-means & Elbow, à l'exception de K-means & BH_Nelder avec certains indices. Des analyses statistiques supplémentaires pourraient confirmer ces tendances observées.

4.1.2 Résultats obtenus sur des données synthétiques générées en faisant varier la taille d'échantillon entre 2000 à 11000 points :

4.1.2.1 Évolution de la précision (Figure 4.4) :

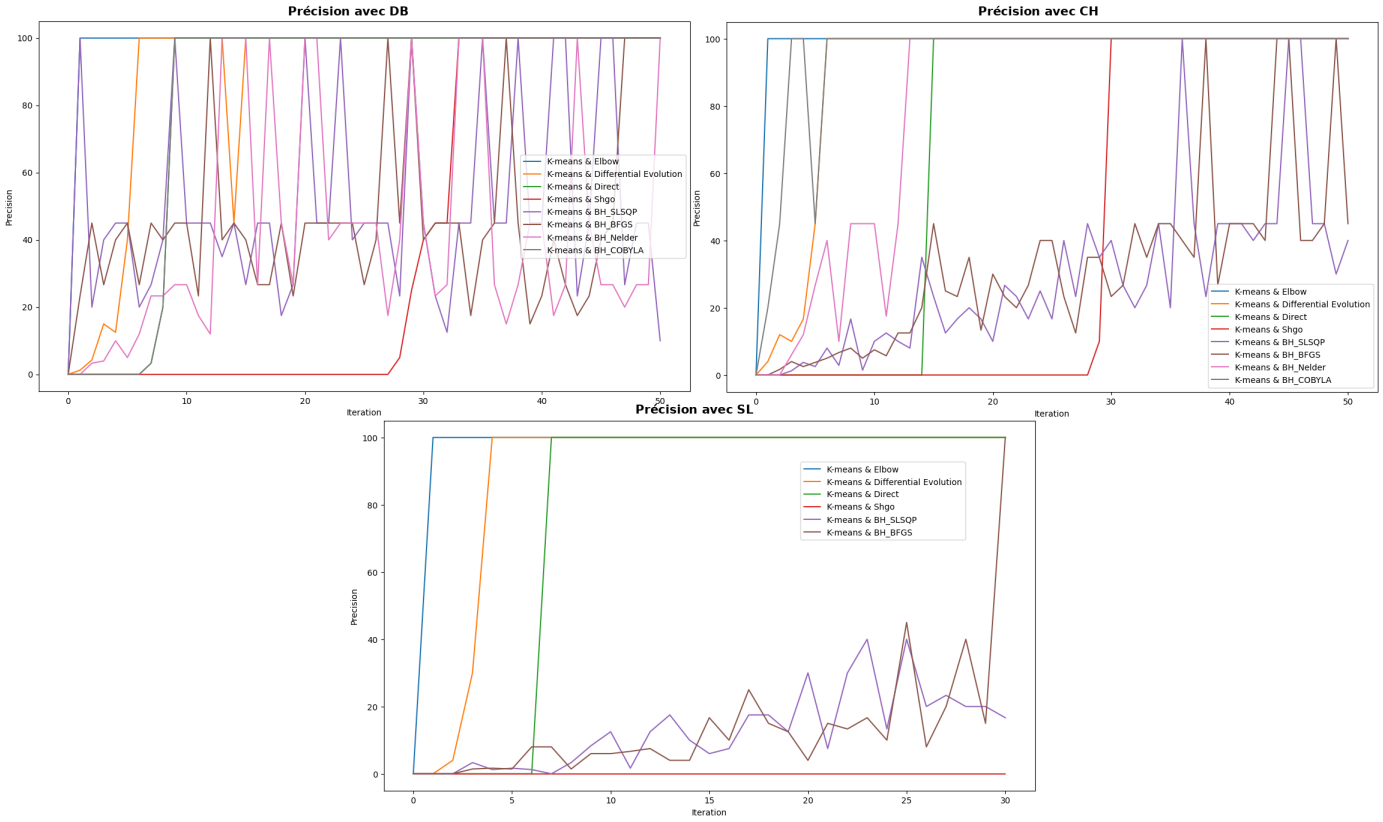

Figure 4.4 Progression de la précision des algorithmes en utilisant les indices de validation Davies-Bouldin, Calinski-Harabasz et Silhouette, avec variation de la taille d'échantillon entre 2000 à 11000 points.

Dans le cadre d'une étude comparative, de l'efficacité des hybridations de K-means avec des métaheuristiques, nous avons observé les comportements suivants :

Première Observation : Variabilité des méthodes hybrides

— Une variabilité importante a été notée dans les graphiques utilisant l'indice de validation de Davies-Bouldin, suggérant une sensibilité aux configurations des données ou à l'initialisation des algorithmes.

Deuxième Observation : Amélioration progressive

— Les méthodes telles que K-means & Shgo, K-means & BH-SLSQP, K-means & BH-Cobyla et K-means & BH-Nelder montrent une amélioration progressive, ce qui pourrait indiquer leur potentiel pour des

scénarios exigeant une convergence stable.

Troisième Observation : Performances variables selon l'indice de Validation

— Une performance variable en fonction de l'indice de validation a été remarquée, avec une tendance à l'amélioration en passant de l'indice Davies-Bouldin à Calinski-Harabasz, et encore plus avec Silhouette.

Quatrième Observation : Rivalité avec la méthode de référence

— K-means & Direct et K-means & Differential Evolution ont montré des résultats prometteurs, en compétition avec la méthode de référence K-means & Elbow.

Conclusion

— Il est conclu que bien que K-means & Elbow demeure un standard robuste, les hybridations avec des métaheuristiques peuvent offrir de nouvelles approches efficaces, méritant des études quantitatives plus approfondies pour valider les observations préliminaires.

4.1.2.2 Évolution de la racine de l'erreur quadratique moyenne RMSE (Figure 4.5) :

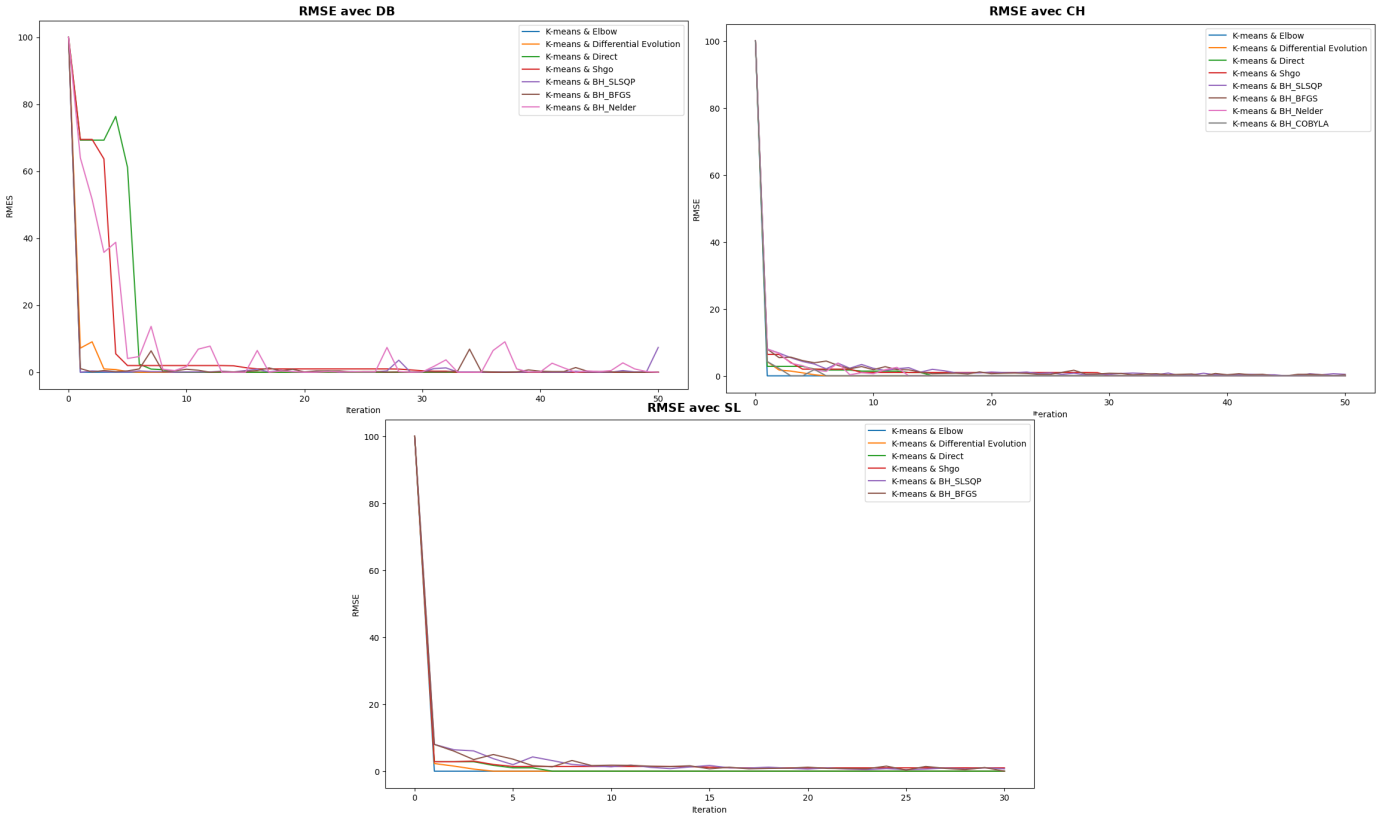

Figure 4.5 Progression de la racine de l'erreur quadratique moyenne RMSE des algorithmes en utilisant les indices de validation Davies-Bouldin, Calinski-Harabasz et Silhouette, avec variation la la taille d'échantillon entre 2000 à 11000 points.

Voici une analyse détaillée de l'évolution des différentes méthodes d'hybridation de K-means avec les métaheuristiques, en utilisant les indices de validation de clustering Davies-Bouldin, Calinski-Harabasz et Silhouette.

Premier Graphique (Davies-Bouldin)

— K-means & Elbow (Bleu) : Sert de référence avec une convergence rapide et une performance stable. Cela suggère une forte adéquation des clusters avec les données.

— K-means & Differential Evolution (Orange) : Une chute rapide indiquant une convergence initiale très efficace, et maintient une RMSE très faible, bien adaptée pour cet ensemble de données.

— K-means & Direct (Vert) : Bonnes performances avec un peu plus de variabilité comparée à Differen-

tial Evolution.

— Autres méthodes (Rouge, Violet, Marron, gris, rose) : Plus grande variabilité et parfois une convergence plus lente, performance inférieure sur cet indice.

Deuxième Graphique (Calinski-Harabasz) :

— La tendance générale reste similaire au premier graphique avec la méthode de référence performante et les méthodes Differential Evolution et Direct affichant de fortes performances.

— Pour les autres méthodes on observe une diminution significative de la RMSE lors de l'utilisation de l'indice CH comme fonction objective, ce qui suggère que cet indice est plus adapté à ces méthodes par rapport à l'indice DB.

Troisième Graphique (Silhouette) :

— Les méthodes Differential Evolution et Direct restent compétitives avec une performance légèrement meilleure pour la Differential Evolution.

— Les méthodes alternatives montrent une plus grande variabilité et des pics moins élevés.

Observations générales :

— K-means & Differential Evolution et K-means & Direct semblent être les concurrents les plus forts face à K-means & Elbow.

— Les autres méthodes montrent une plus grande variabilité qui pourrait être due à des facteurs comme la sensibilité aux conditions initiales ou une adaptation moins idéale aux spécificités des données synthétiques.

Conclusion :

— Une alternative viable à K-means & Elbow pourrait être l'hybridation de K-means avec Differential Evolution et Direct, en particulier pour réduire le temps de calcul tout en conservant une haute exactitude. Une évaluation quantitative est recommandée pour confirmer ces observations préliminaires.

4.1.2.3 Évolution du temps d'exécution (Figure 4.6) :

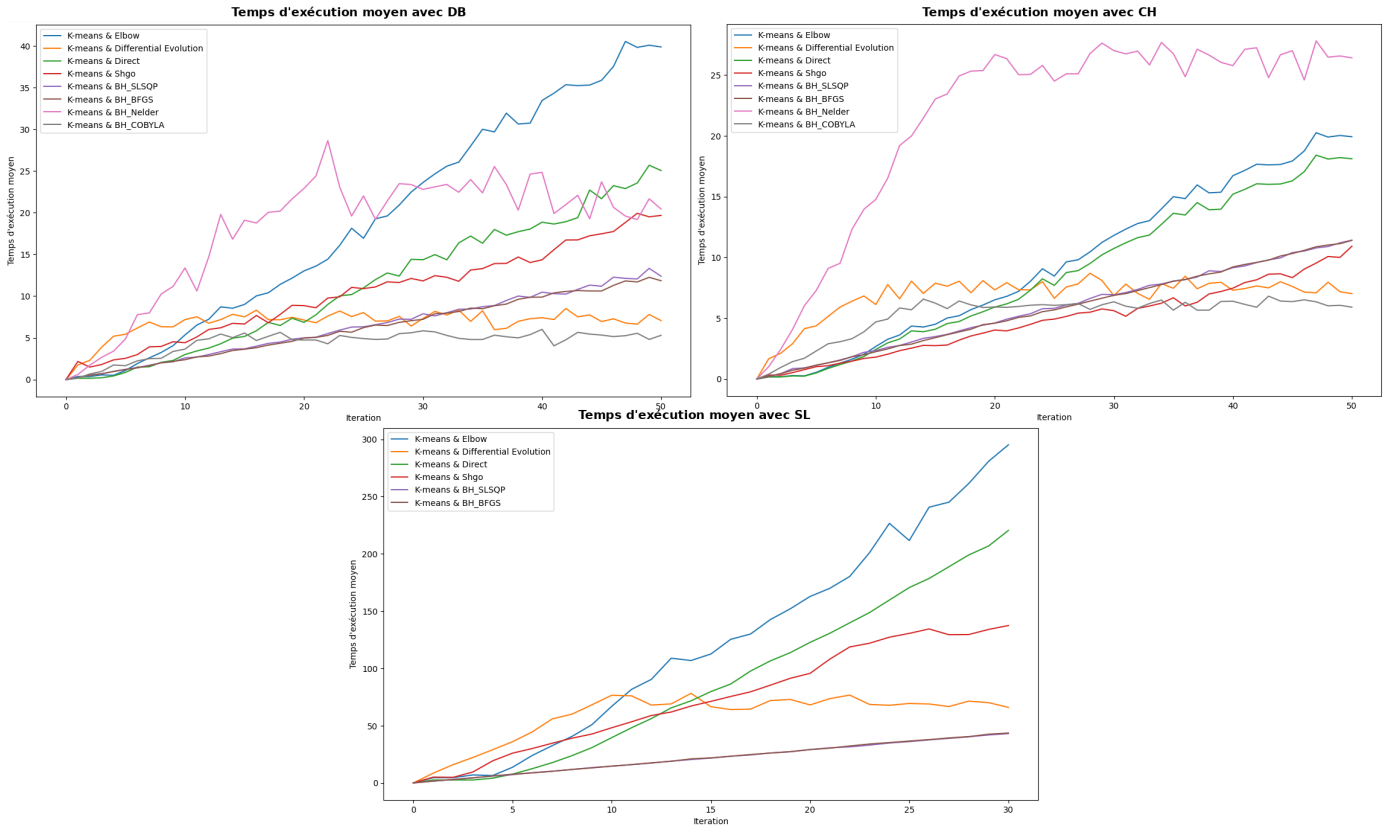

Figure 4.6 Progression du temps d'exécution des algorithmes en utilisant les indices de validation Davies-Bouldin, Calinski-Harabasz et Silhouette, avec variation la la taille d'échantillon entre 2000 à 11000 points.

L'analyse des graphes en termes de temps d'exécution moyen des différentes hybridations du k-means avec des métaheuristiques, en utilisant différents indices de validation, nous amène aux observations suivantes :

Davies-Bouldin (DB) comme fonction objective :

- L'hybridation k-means avec Elbow présente une augmentation significative du temps d'exécution, ce qui indique une performance temporelle suboptimale.
- Differential Evolution et Direct montrent une croissance modérée du temps d'exécution, suggérant une efficacité temporelle raisonnable.
- BH-Nelder et BH-COBYLA démontrent une meilleure optimisation du temps d'exécution, avec une augmentation moins abrupte au fil des itérations.

Calinski-Harabasz (CH) comme fonction objective :

— On constate que, en général, le temps d'exécution des méthodes a légèrement diminué par rapport à l'utilisation de l'indice Davies-Bouldin, ce qui suggère que l'indice Calinski-Harabasz est plus rapide.

— Une gestion du temps d'exécution améliorée est observée par rapport au graphe DB, avec des augmentations moins prononcées.

— Les performances de Differential Evolution et Direct restent robustes, indiquant une bonne stabilité de ces méthodes.

Silhouette (SL) comme fonction objective :

— Il est constaté que le temps d'exécution pour toutes les méthodes augmente de façon exponentielle, ce qui indique que cet indice nécessite considérablement plus de temps d'exécution comparé aux indices Davies-Bouldin et Calinski-Harabasz.

— Une optimisation significative du temps d'exécution est notée, avec des courbes plus plates pour la plupart des méthodes.

— La méthode de référence k-means avec Elbow reste la moins performante en termes d'efficacité temporelle.

Conclusion : Les méthodes d'hybridation de k-means avec des métaheuristiques, en particulier Direct et Differential Evolution, surpassent la méthode de référence k-means avec Elbow en termes de temps d'exécution moyen. L'utilisation de l'indice Calinski-Harabasz en particulier, semble favoriser une meilleure efficacité temporelle, ce qui suggère que cet indice, combiné à des métaheuristiques spécifiques, pourrait offrir une approche optimale pour la minimisation du temps d'exécution tout en préservant une bonne exactitude du clustering.

4.1.3 Présentation et analyse des résultats obtenus sur les données de cancer :

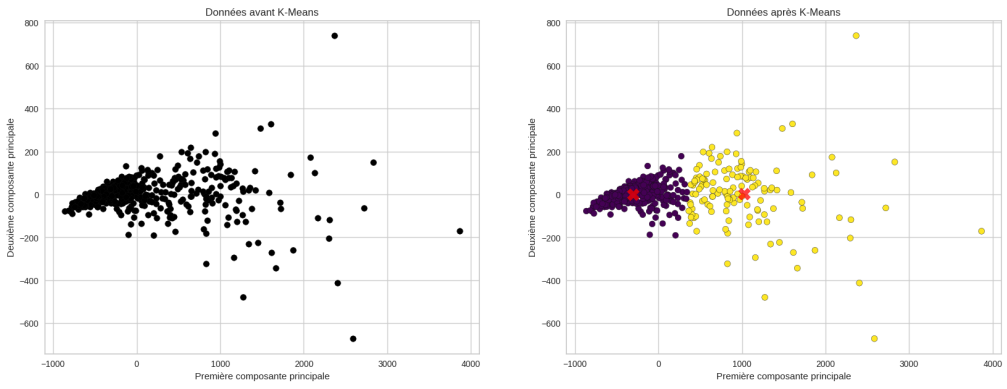

Figure 4.7 Présentation des données issues de l'ensemble load-breast-cancer avant et après l'application de l'algorithme k-means.

4.1.3.1 Analyse des résultats obtenus avec l'indice de Calinski-Harabasz :

Évolution de la racine de l'erreur quadratique moyenne RMSE (Figure 4.8) :

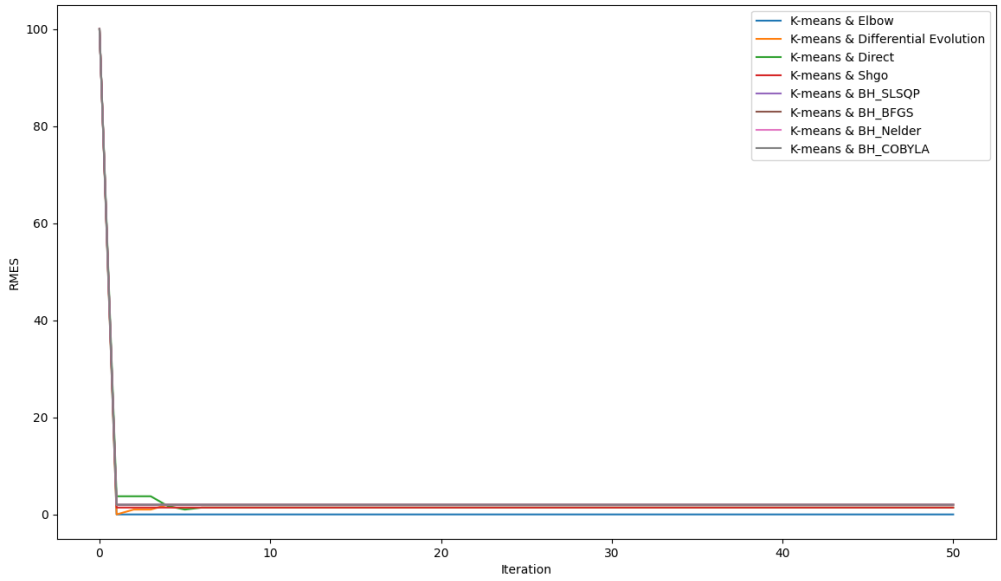

Figure 4.8 Évolution de la RMSE des algorithmes en fonction de l'indice de validité CH

Sur ce graphique, la valeur de la *RMSE* (racine de l'erreur quadratique moyenne) est utilisée pour évaluer et comparer la performance de diverses méthodes d'hybridation de l'algorithme K-means avec des métaheuristiques. La méthode de référence est K-means avec la méthode Elbow, représentée en bleu. Les données utilisées sont issues de l'ensemble de données `load_breast_cancer` de scikit-learn, et l'indice de validité Calinski-Harabasz a été utilisé pour la validation du clustering.

Analysons le graphique :

- **Courbe bleue (K-means & Elbow)** : Cette méthode sert de référence pour la comparaison. Elle montre un pic initial élevé de RMSE qui décroît rapidement au cours des premières itérations et qui se stabilise ensuite. Cela indique que la méthode Elbow permet de trouver rapidement un nombre raisonnable de clusters, même si le RMSE initial est élevé.
- **Les autres courbes** représentent les hybridations de K-means avec différentes métaheuristiques. Aucune de ces méthodes n'a un RMSE inférieur à celui de la méthode de référence (K-means & Elbow) après le pic initial. Cela suggère que toutes les hybridations ont au moins la même efficacité

que la méthode Elbow pour stabiliser la valeur de RMSE, mais pas une meilleure efficacité.

— Toutes les méthodes semblent converger vers des valeurs similaires de RMSE, ce qui suggère que, bien qu'elles aient pu commencer avec des valeurs différentes de RMSE, elles finissent par offrir une qualité de clustering comparable.

— **Les courbes orange, verte, rouge, violette, marron, rose et grise** représentent respectivement les hybridations de K-means avec Differential Evolution, Direct, Shgo, BH-SLSQP, BH-BFGS, BH-Nelder et BH-COBYLA. Elles convergent toutes assez rapidement vers des valeurs basses de RMSE, ce qui indique que ces méthodes d'hybridation optimisent efficacement les centres de clusters pour minimiser la distance quadratique totale entre les points de données et le centre de leur cluster attribué.

Pour conclure, le graphique montre que l'hybridation de K-means avec des métaheuristiques est efficace pour le clustering de l'ensemble de données `load_breast_cancer`. Malgré des valeurs initiales de RMSE différentes, toutes les méthodes convergent vers une performance similaire, ce qui suggère qu'elles sont des alternatives viables à l'approche Elbow traditionnelle. Cependant, il serait important de prendre en compte d'autres mesures de performance et la nature des données pour une évaluation complète.

Évolution du temps d'exécution :

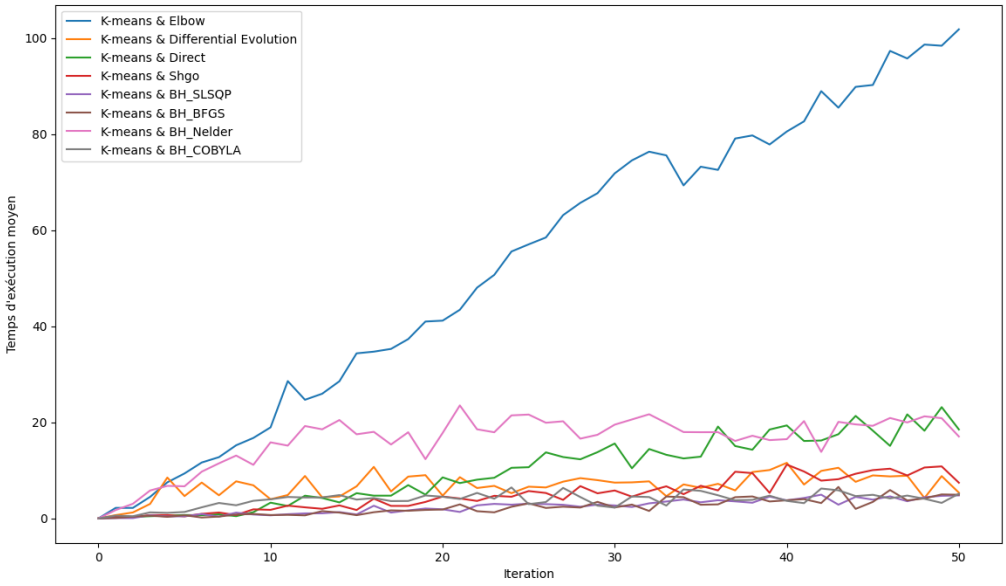

Figure 4.9 Évolution du temps d'exécution moyen des algorithmes en fonction de du nombre d'itérations avec l'indice de validité CH

Ce graphique illustre l'évolution du temps d'exécution moyen de différentes méthodes d'hybridation de

l'algorithme K-means avec des métaheuristiques, en utilisant l'ensemble de données `load_breast_cancer` de scikit-learn et l'indice de validité Calinski-Harabasz pour valider le clustering. L'algorithme de référence est le K-means avec l'approche Elbow, représentée en bleu.

Voici les observations clés de l'analyse du graphique :

- **Courbe bleue (K-means & Elbow)** : Cette courbe montre une augmentation progressive du temps d'exécution moyen avec le nombre d'itérations. C'est la méthode de référence et semble avoir le temps d'exécution le plus élevé parmi les méthodes testées, ce qui pourrait être attribué à la méthode Elbow nécessitant de nombreux calculs pour évaluer le nombre optimal de clusters à chaque itération.
- **Courbes des autres méthodes** : Les courbes représentent le temps d'exécution moyen des différentes méthodes d'hybridation du K-means avec des métaheuristiques. Elles commencent toutes avec des temps inférieurs par rapport à la méthode de référence et restent généralement en dessous de celle-ci tout au long des 50 itérations. Cela suggère que ces méthodes sont plus rapides que K-means avec Elbow.
- **Courbe orange (K-means & Differential Evolution)** : Cette méthode montre un temps d'exécution moyen relativement bas et stable, ce qui suggère que l'algorithme de Differential Evolution peut aider à accélérer le processus de clustering sans nécessiter de temps supplémentaire pour des calculs itératifs importants.
- **Courbe verte (K-means & Direct)** et **courbe rose (K-means & BH-Nelder)** : Elles montrent des tendances similaires avec des fluctuations dans le temps d'exécution, mais restent sous la courbe de référence, ce qui indique une efficacité en termes de temps tout en utilisant ces méthodes d'optimisation.
- **Courbes violette (K-means & BH-SLSQP), marron (K-means & BH-BFGS), rouge (K-means & Shgo)** et **grise (K-means & BH-COBYLA)** : Ces méthodes affichent une tendance relativement stable avec des augmentations occasionnelles, suggérant que bien qu'elles peuvent parfois nécessiter plus de temps pour certaines itérations, en moyenne, elles restent plus efficaces en termes de temps par rapport à la méthode de référence.

En conclusion, bien que la méthode de référence Elbow offre une base pour l'évaluation, les méthodes d'hybridation semblent fournir une optimisation plus rapide du clustering, ce qui peut être avantageux pour

des applications nécessitant une grande vitesse de traitement des données. Cependant, il est essentiel de considérer également la qualité du clustering obtenue, pas seulement le temps d'exécution.

4.1.3.2 Analyse des résultats obtenus avec l'indice de Silhouette :

Évolution de la RMSE :

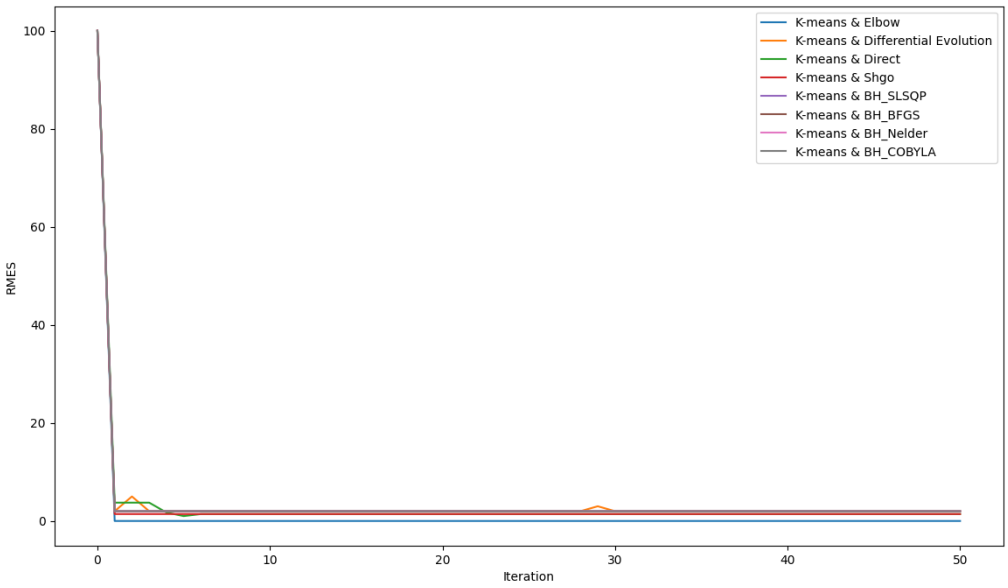

Figure 4.10 Évolution de la RMSE des algorithmes en fonction du nombre d'itérations avec l'indice de validité Silhouette

Le graphique partagé illustre l'évolution de la racine de l'erreur quadratique moyenne (RMSE) pour différentes hybridations de l'algorithme K-means avec des métaheuristiques, appliquées à l'ensemble de données `load_breast_cancer` de scikit-learn. L'indice de validité utilisé pour le clustering est l'indice de silhouette, qui est un moyen d'évaluer la qualité des clusters obtenus sans connaître les étiquettes véritables.

Le graphe montre le comportement suivant :

- **La courbe bleue (K-means & Elbow)** représente la méthode de référence. On observe une chute très rapide de la RMSE au début, qui se stabilise ensuite. Cela indique que la méthode de référence atteint rapidement une solution stable en termes de minimisation de la RMSE.
- **Les autres courbes** représentent les différentes hybridations du K-means avec des métaheuristiques. Aucune de ces courbes ne dépasse la courbe de référence après la chute initiale, ce qui suggère

qu'aucune hybridation ne conduit à une détérioration de la performance par rapport à la méthode de référence en termes de RMSE.

— Les méthodes **K-means & Differential Evolution (orange)**, **K-means & Direct (vert)**, **K-means & Shgo (rouge)**, **K-means & BH-SLSQP (violet)**, **K-means & BH-BFGS (marron)**, **K-means & BH-Nelder (rose)**, et **K-means & BH-COBYLA (gris)** semblent avoir une performance similaire mais pas meilleure que la méthode Elbow en ce qui concerne la RMSE, indiquant une convergence rapide vers des solutions de qualité.

— Toutes les méthodes, à l'exception de K-means & Elbow, présentent une certaine variabilité après la stabilisation, mais restent dans un intervalle de RMSE très bas, ce qui pourrait indiquer une équivalence dans la qualité de clustering ou des différences minimes qui peuvent dépendre d'autres facteurs tels que la configuration des paramètres de l'algorithme.

En conclusion, ce graphique suggère que l'hybridation de K-means avec des métaheuristiques pourrait être une stratégie viable pour maintenir ou améliorer la qualité de clustering mesurée par la RMSE, sans compromettre les performances par rapport à la méthode de référence K-means & Elbow. Cependant, il est essentiel de prendre en compte que la RMSE n'est qu'un aspect de la performance de clustering et que l'indice de silhouette doit également être considéré pour évaluer la cohésion et la séparation des clusters.

Évolution du temps d'exécution :

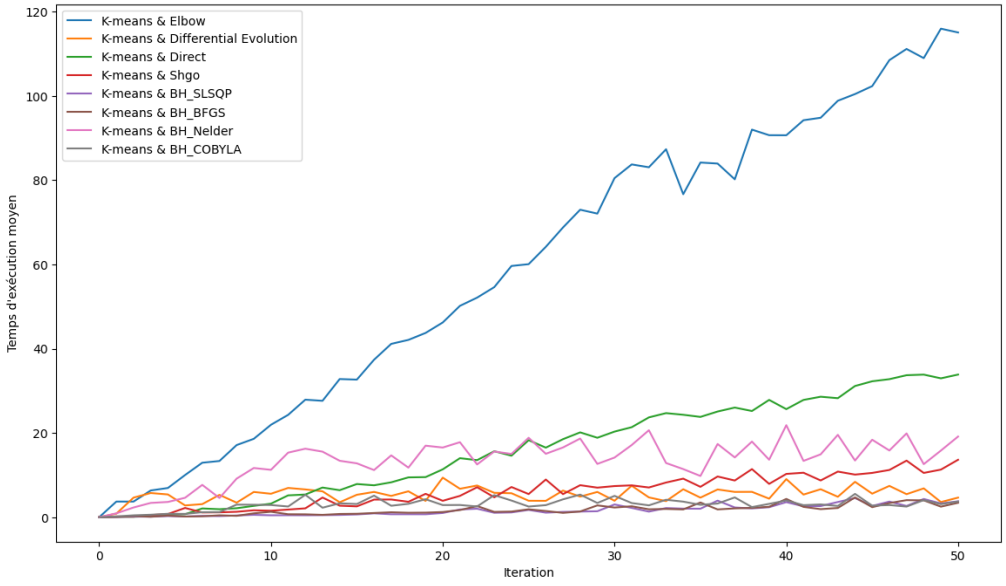

Figure 4.11 Évolution du temps d'exécution moyen des algorithmes en fonction de du nombre d'itérations avec l'indice de validité Silhouette

Le graphique représente le temps d'exécution moyen de différentes hybridations de l'algorithme K-means avec des métaheuristiques en fonction du nombre d'itérations, utilisant l'ensemble de données `load_breast_cancer` de scikit-learn et validant le clustering à l'aide de l'indice de validité Silhouette.

Analyse du graphique :

- **Courbe bleue (K-means & Elbow)** : L'algorithme de référence montre une augmentation notable et continue du temps d'exécution moyen au fur et à mesure des itérations, indiquant que la méthode Elbow peut être coûteuse en termes de calculs.
- **Les autres courbes** représentent le temps d'exécution moyen des hybridations de K-means avec diverses métaheuristiques. Toutes augmentent plus lentement que la méthode de référence, suggérant une meilleure efficacité temporelle.
- **Courbe verte (K-means & Direct)** : Cette courbe reste relativement plates et basses, suggérant que cette méthode sont est rapide et peut-être plus directes dans sa convergence vers une solution.
- **Courbes rouge (K-means & Shgo)**, **Courbe orange (K-means & Differential Evolution)**, **violet (K-means & BH-SLSQP)**, **marron (K-means & BH-BFGS)**, **rose (K-means & BH-Nelder)** et **grise (K-means & BH-COBYLA)** : Elles montrent des augmentations progressives en dessous de la courbe de référence, indiquant leur efficacité temporelle avec des performances relativement stables au cours des itérations.

En conclusion, les hybridations de K-means avec des métaheuristiques peuvent offrir des avantages en termes de temps d'exécution moyen par rapport à la méthode de référence K-means & Elbow. Cependant, il est crucial de considérer également la qualité des clusters formés, évaluée par l'indice de validité Silhouette, pour une analyse comparative complète. L'équilibre entre le temps d'exécution et la qualité des clusters est important.

4.2 Discussion et interprétation des résultats des résultats

4.2.1 Discussion et interprétation des résultats obtenus avec Davies-Bouldin sur des données synthétiques :

Méthode	Exactitude (%)	Précision (%)	RMSE	MAE	Temps d'exécution
K-means et Elbow	100	100	0	0	9.26
K-means et ED	93.2	93.2	0.28	0.11	3.44
K-means et Direct	87.56	87.53	3.39	1.73	4.63
K-means et BH-Nelder	86.72	86.72	6.5	1.87	13.46
K-means et BH-COBYLA	83.56	83.56	8.45	3.26	3.81
K-means et Shgo	64.08	64.06	3.63	2.28	4.07
K-means et BH-SLSQP	34.08	33.99	11.44	7.23	1.04
K-means et BH-BFGS	32.56	32.56	11.67	7.42	1.08

Table 4.1 Résumé des résultats obtenus avec l'indice de validation Davies-Bouldin, illustrant l'évolution moyenne de différents algorithmes en termes de précision, d'exactitude, d'erreur quadratique moyenne, d'erreur absolue moyenne, et de temps d'exécution moyen exprimé en minutes.

Les résultats obtenus mettent en évidence l'efficacité des approches hybrides entre K-means et différentes métaheuristiques pour résoudre le problème NP-difficile du clustering optimal. L'utilisation de l'indice de validité Davies-Bouldin comme fonction objective a permis d'évaluer la qualité des clusters obtenus.

Les résultats montrent que les hybridations K-means & Differential Evolution et K-means & Direct rivalisent efficacement avec la méthode de référence K-means & Elbow en termes d'exactitude, de racine de l'erreur quadratique moyenne (RMSE) et de temps d'exécution moyen. Ces approches parviennent à maintenir une bonne séparation des clusters et à converger vers des solutions optimales avec une gestion efficace du temps d'exécution.

En revanche, les méthodes K-means & BH-COBYLA, K-means & BH-BFGS, K-means & Shgo, K-means & BH-SLSQP et K-means & BH-Nelder ne rivalisent pas autant avec la méthode Elbow sur certains critères. Ceci peut être attribué à leur sensibilité à la configuration des clusters ou à leur stratégie d'optimisation, qui peut ne pas être aussi efficace pour maintenir la qualité du clustering avec des structures de données complexes.

Ces observations soulignent l'importance de choisir une méthode d'hybridation adaptée à la taille et à la complexité structurelle des données pour obtenir un clustering de qualité. La variabilité des performances des différentes approches indique également la nécessité d'une analyse approfondie pour identifier les

facteurs qui influencent leur efficacité dans divers scénarios de clustering.

4.2.2 Discussion et interprétation des résultats obtenus avec Calinski-Harabasz sur des données synthétiques :

Méthode	Exactitude (%)	Précision (%)	RMSE	MAE	Temps d'exécution
K-means et Elbow	98.00	98.00	0.75	0.27	5.41
K-means et ED	92.82	92.77	1.25	0.27	2.07
K-means et Direct	85.30	85.30	1.23	0.80	4.92
K-means et BH-Nelder	88.32	88.32	4.05	1.80	8.79
K-means et BH-COBYLA	64.04	64.04	15.00	8.72	2.14
K-means et Shgo	76.84	76.84	1.25	1.07	2.54
K-means et BH-SLSQP	69.30	68.95	4.08	2.15	1.73
K-means et BH-BFGS	69.96	69.96	4.19	2.54	1.70

Table 4.2 Résumé des résultats obtenus avec l'indice de validation d Calinski-Harabasz, illustrant l'évolution moyenne de différents algorithmes en termes de précision, d'exactitude, d'erreur quadratique moyenne, d'erreur absolue moyenne, et de temps d'exécution moyen exprimé en minutes.

Dans le cadre de cette simulation, l'exploration des résultats obtenus révèle des insights intéressants sur les performances des différentes approches hybrides entre K-means et diverses métaheuristiques, dans le contexte du problème NP-difficile de clustering optimal. L'utilisation de l'indice de validité Calinski-Harabasz comme fonction objective a permis d'évaluer la qualité des clusters obtenus et de comparer les différentes hybridations.

Les résultats indiquent que les hybridations K-means & Differential Evolution et K-means & Direct rivalisent efficacement avec la méthode de référence K-means & Elbow sur différentes métriques, y compris l'exactitude, la racine de l'erreur quadratique moyenne (RMSE) et le temps d'exécution moyen. Cela suggère que ces approches hybrides parviennent à maintenir une bonne séparation des clusters et à converger vers des solutions optimales, tout en gérant efficacement le temps d'exécution.

D'autre part, les méthodes K-means & BH-COBYLA, K-means & BH-BFGS, K-means & Shgo, K-means & BH-SLSQP et K-means & BH-Nelder ne rivalisent pas autant avec la méthode Elbow sur certaines métriques comme l'exactitude. Cependant, il est remarquable de constater une amélioration significative de leurs performances par rapport aux résultats précédemment obtenus avec l'indice de validité Davies-Bouldin. Ceci peut être attribué à la sensibilité de ces méthodes aux paramètres de clustering ou à leur stratégie d'opti-

misation, qui peut être mieux adaptée à l'indice Calinski-Harabasz.

Ces observations mettent en évidence l'importance de choisir une méthode d'hybridation adaptée à la taille et à la complexité structurelle des données pour obtenir un clustering de qualité. La variabilité des performances des différentes approches indique également la nécessité d'une analyse approfondie pour identifier les facteurs qui influencent leur efficacité dans divers scénarios de clustering.

En conclusion, les résultats obtenus dans cette simulation démontrent le potentiel des approches hybrides entre K-means et des métaheuristiques pour résoudre efficacement le problème du clustering optimal, offrant une alternative compétitive à la méthode Elbow tout en réduisant le temps d'exécution. Les améliorations observées avec l'indice Calinski-Harabasz soulignent l'importance de la sélection d'une fonction objective adaptée pour évaluer la qualité des clusters obtenus.

4.2.3 Discussion et interprétation des résultats obtenus avec l'indice de validité Silhouette sur des données synthétiques :

Méthode	Exactitude (%)	Précision (%)	RMSE	MAE	Temps d'exécution
K-means et Elbow	100.00	100.00	0.00	0.00	21.58
K-means et ED	93.24	93.23	0.26	0.10	9.34
K-means et Direct	85.30	85.30	1.23	0.80	4.92
K-means et Shgo	76.84	76.84	1.25	1.07	2.54
K-means et BH-SLSQP	69.30	68.95	4.08	2.51	1.73
K-means et BH-BFGS	69.96	69.96	4.19	2.54	1.70

Table 4.3 Résumé des résultats obtenus avec l'indice de validation Silhouette , illustrant l'évolution moyenne de différents algorithmes en termes de précision, d'exactitude, d'erreur quadratique moyenne, d'erreur absolue moyenne, et de temps d'exécution moyen exprimé en minutes.

Les résultats montrent que les hybridations K-means & Direct et K-means & Differential Evolution rivalisent efficacement avec la méthode de référence K-means & Elbow sur différentes métriques. Cela suggère que ces approches hybrides parviennent à maintenir une bonne séparation des clusters et à converger vers des solutions optimales.

D'autre part, les méthodes K-means & BH-BFGS, K-means & Shgo et K-means & BH-SLSQP ne rivalisent pas autant avec la méthode Elbow sur certaines métriques comme l'exactitude. Toutefois, il est à noter que le temps d'exécution moyen utilisant l'indice Silhouette est considérablement plus long que celui observé

avec l'indice Davies-Bouldin et l'indice Calinski-Harabasz. Ceci peut être attribué à la nature de l'indice Silhouette, qui nécessite des calculs supplémentaires pour évaluer la qualité des clusters, entraînant ainsi une augmentation du temps d'exécution.

Ces observations soulignent l'importance de choisir une méthode d'hybridation adaptée à la taille et à la complexité structurelle des données pour obtenir un clustering de qualité. La variabilité des performances des différentes approches indique également la nécessité d'une analyse approfondie pour identifier les facteurs qui influencent leur efficacité dans divers scénarios de clustering.

En conclusion, les résultats obtenus dans cette simulation mettent en évidence le potentiel des approches hybrides entre K-means et des métaheuristiques pour résoudre efficacement le problème du clustering optimal, offrant une alternative compétitive à la méthode Elbow tout en gérant le temps d'exécution. Les différences observées avec l'indice Silhouette soulignent l'importance de la sélection d'une fonction objective adaptée pour évaluer la qualité des clusters obtenus.

4.2.4 Discussion et interprétation des résultats obtenus avec l'indice de validité Silhouette et Calinski-Harabasz sur les données réelles de cancer :

Dans le cadre de ce projet, les résultats obtenus sur l'ensemble de données `load_breast_cancer` de scikit-learn mettent en évidence les performances des différentes approches hybrides entre K-means et diverses métaheuristiques pour résoudre le problème NP-difficile de clustering optimal. Les indices de validité Silhouette et Calinski-Harabasz ont été utilisés comme fonctions objectif pour évaluer la qualité des clusters obtenus.

L'analyse des résultats montre que l'hybridation K-means & Elbow se distingue par son RMSE nul pour toutes les itérations, ce qui indique une excellente précision dans la détermination du nombre optimal de clusters. Cependant, cette méthode présente le temps d'exécution moyen le plus élevé par rapport aux autres méthodes, ce qui souligne le compromis entre la précision et l'efficacité temporelle.

D'autre part, les hybridations K-means & BH-BFGS, K-means & Shgo, K-means & BH-SLSQP, K-means & Direct et K-means & Differential Evolution montrent une compétitivité notable par rapport à K-means & Elbow en termes de RMSE, avec une légère erreur de 2 clusters. De plus, elles offrent un temps d'exécution moyen considérablement réduit, ce qui démontre leur efficacité pour ce type de données.

Les résultats obtenus avec les deux indices de validité, Silhouette et Calinski-Harabasz, sont similaires en termes de RMSE. Cependant, l'utilisation de l'indice Silhouette entraîne un temps d'exécution moyen légèrement meilleur que celui obtenu avec l'indice Calinski-Harabasz. Ceci suggère que l'indice Silhouette peut offrir un avantage en termes d'efficacité temporelle pour l'évaluation de la qualité des clusters.

En conclusion, les résultats de ces tests mettent en évidence l'importance de choisir une méthode d'hybridation adaptée aux caractéristiques des données et aux contraintes de temps d'exécution. Les approches hybrides entre K-means et diverses métaheuristiques se révèlent être des alternatives prometteuses à la méthode Elbow traditionnelle, offrant une combinaison efficace de précision et d'efficacité temporelle pour le clustering optimal.

4.2.5 **Comparaison des résultats avec des projets antérieurs similaires :**

Dans le contexte de notre projet et l'article "Automatic Clustering Using an Improved Differential Evolution Algorithm" (Das *et al.*, 2007), on observe une convergence des objectifs malgré des approches diverses. Notre projet explore l'hybridation de l'algorithme K-means avec les métaheuristiques **(tel que K-means & Differential Evolution et K-means & Direct)** pour résoudre le problème NP-complet de détermination du nombre optimal de clusters, un défi notable dans le clustering de données non étiquetées. L'article, de son côté, illustre une amélioration significative de l'algorithme de l'évolution différentielle, appliquée au clustering automatique sans connaissance préalable du nombre de partitions nécessaires, montrant ainsi une autre voie prometteuse pour surmonter des limitations similaires.

L'article "Boosting k-means clustering with symbiotic organisms search for automatic clustering problems" (Ikotun et Ezugwu, 2022), examine une approche similaire en hybridant K-means avec l'algorithme de recherche d'organismes symbiotiques (SOS), un algorithme métaheuristique visant à améliorer la sélection des centroids initiaux de K-means pour surmonter ses limitations, notamment sa sensibilité aux conditions initiales et sa tendance à converger vers des optima locaux. L'approche proposée dans l'article a montré une amélioration significative de la performance par rapport au K-means classique et à d'autres approches hybrides, en utilisant également des jeux de données de l'UCI pour les tests.

L'article analysé, "K-Means-Based Nature-Inspired Metaheuristic Algorithms for Automatic Data Clustering Problems : Recent Advances and Future Directions" (Ikotun *et al.*, 2021), se concentre sur la synthèse des approches récentes pour améliorer l'algorithme de K-means à travers l'utilisation de métaheuristiques ins-

pirées de la nature, ciblant également la détermination automatique du nombre de clusters. Cet article évalue diverses améliorations apportées à K-means, qui visent à améliorer la performance et la capacité de cet algorithme à gérer des problèmes de clustering automatique sans avoir à prédéfinir le nombre de clusters.

L'article, "Dynamic clustering using particle swarm optimization with application in image segmentation" (Omran *et al.*, 2006), présente une approche similaire en utilisant l'optimisation par essaim particulaire (PSO) pour dynamiquement déterminer et optimiser le nombre de clusters en minimisant l'intervention de l'utilisateur. Cette méthode commence par une partition initiale du jeu de données en un nombre relativement élevé de clusters pour minimiser les effets des conditions initiales. Le nombre optimal de clusters est ensuite raffiné à l'aide de K-means, ce qui souligne l'efficacité de combiner K-means avec d'autres méthodes d'optimisation pour surmonter ses limitations, notamment sa sensibilité aux conditions initiales et sa tendance à converger vers des solutions sous-optimales

Notre travail, ainsi que les études que nous avons consultées, soulignent l'efficacité et la pertinence des approches hybrides entre k-means et les métaheuristiques dans le clustering des données. Ils montrent que ces techniques peuvent fournir des solutions robustes et flexibles pour le clustering, surtout quand le nombre de clusters n'est pas connu à l'avance. Ces convergences indiquent une direction prometteuse pour les recherches futures et pour des applications pratiques dans divers domaines nécessitant une analyse de données sophistiquée.

4.3 **Conclusion du chapitre analyse et discussions :**

En conclusion, ce chapitre a exploré en profondeur l'efficacité des approches hybrides entre K-means et diverses métaheuristiques dans le contexte du clustering optimal, un problème reconnu pour sa complexité NP-difficile. L'utilisation des indices de validité Davies-Bouldin, Silhouette et Calinski-Harabasz comme fonctions objectif a permis d'évaluer la qualité des clusters obtenus et de comparer les performances des différentes hybridations.

Les résultats obtenus avec l'indice Davies-Bouldin ont mis en évidence la compétitivité des hybridations **K-means & Differential Evolution et K-means & Direct** par rapport à la méthode de référence K-means & Elbow, en termes d'exactitude, de RMSE et de temps d'exécution. Cependant, certaines méthodes comme K-means & BH-COBYLA et K-means & BH-BFGS ont montré des performances inférieures sur certains critères,

probablement en raison de leur sensibilité aux paramètres de clustering ou à leur stratégie d'optimisation.

L'analyse des résultats avec l'indice Calinski-Harabasz a révélé des observations similaires, avec une amélioration notable des performances des méthodes K-means & BH-COBYLA, K-means & BH-BFGS, K-means & Shgo, K-means & BH-SLSQP et K-means & BH-Nelder par rapport aux résultats obtenus avec l'indice Davies-Bouldin. Cette amélioration suggère que ces méthodes peuvent être mieux adaptées à l'indice Calinski-Harabasz pour évaluer la qualité du clustering.

L'évaluation des résultats avec l'indice Silhouette a également confirmé l'efficacité des hybridations **K-means & Direct et K-means & Differential Evolution**. Néanmoins, il a été observé que le temps d'exécution moyen avec l'indice Silhouette est légèrement supérieur à celui des autres indices, ce qui indique que cet indice nécessite des calculs supplémentaires pour évaluer la qualité des clusters.

L'analyse des résultats sur l'ensemble de données load_breast_cancer a démontré l'excellente précision de l'hybridation K-means & Elbow, malgré son temps d'exécution élevé. Les hybridations K-means & BH-BFGS, K-means & Shgo, K-means & BH-SLSQP, **K-means & Direct et K-means & Differential Evolution** ont montré une compétitivité notable en termes de RMSE et un temps d'exécution nettement réduit, ce qui souligne leur efficacité pour ce type de données.

En somme, cette analyse approfondie des résultats a révélé que les approches hybrides entre K-means et diverses métaheuristiques constituent des alternatives prometteuses à la méthode Elbow traditionnelle pour le clustering optimal. Ces approches offrent une combinaison efficace de précision et d'efficacité temporelle, tout en s'adaptant aux spécificités des différentes métriques d'évaluation et des ensembles de données.

CONCLUSION

La conclusion de mon projet de fin d'études, qui vise à résoudre le défi de détermination du nombre optimal de clusters dans le contexte de l'utilisation de l'algorithme K-means, souligne l'importance de l'approche hybride intégrant les métaheuristiques. Ce problème, étant NP-complet, échappe aux solutions exactes dans un temps polynomial, les méthodes exactes existantes, telles que la méthode Elbow, nécessitant un temps exponentiel.

Mon projet se distingue par une hybridation novatrice entre K-means et différentes métaheuristiques, notamment la Differential Evolution, SHGO, Direct, et Basin Hopping. L'objectif était de développer une méthode capable de concurrencer efficacement la méthode Elbow, en fournissant une estimation approximative du nombre optimal de clusters K en un temps d'exécution raisonnable.

L'évaluation de ces algorithmes hybrides a été réalisée sur des données synthétiques générées à l'aide du modèle de mélange gaussien de la bibliothèque sklearn. Trois indices de validation de clustering (Davies-Bouldin, Calinski-Harabasz, et Silhouette) ont été utilisés pour mesurer la performance. Les métriques telles que l'exactitude, la précision, l'erreur quadratique moyenne, et le temps d'exécution ont servi à suivre la progression.

Les résultats de la première expérience, variante le nombre de clusters de 2 à 100, démontrent que les algorithmes hybrides K-means & Differential Evolution et K-means & Direct sont très compétitifs par rapport à la méthode de référence K-means & Elbow, performant bien sur toutes les métriques et indices de validation. Il est également observé que le choix de l'indice de validation influence significativement les résultats des algorithmes, certains se montrant plus performants avec l'indice Calinski-Harabasz, tandis que d'autres s'améliorent avec Davies-Bouldin. L'indice de validation Silhouette, bien qu'efficace, s'est avéré coûteux en termes de temps d'exécution.

La deuxième expérience, qui consistait à varier la taille des échantillons de 2000 à 11000, a confirmé la stabilité des résultats observés précédemment, bien que Davies-Bouldin ait montré une sensibilité notable à la taille de l'échantillon, contrairement aux indices Calinski-Harabasz et Silhouette qui se sont avérés plus stables.

Dans la troisième expérience, l'application de ces algorithmes hybrides sur le jeu de données réel de cancer (load_breast_cancer de scikit-learn) a produit de bons résultats, avec une marge d'erreur acceptable de 2 clusters. Sur ces données, l'indice Silhouette s'est révélé légèrement plus rapide que Calinski-Harabasz, une observation non constatée dans les expériences précédentes.

En conclusion, l'hybridation de K-means avec des métaheuristiques se présente comme une alternative viable pour résoudre le problème du nombre optimal de clusters. Cette approche offre non seulement une compétitivité notable face à la méthode Elbow en termes d'exactitude, de précision, d'erreur quadratique moyenne, mais aussi en termes de temps d'exécution.

Les perspectives futures incluent l'application de ces algorithmes sur d'autres jeux de données réels pour explorer leur comportement dans différents scénarios et tester l'intégration d'autres métaheuristiques avec différents indices de validation afin d'améliorer davantage les résultats. Cette recherche ouvre la voie à des investigations plus approfondies sur l'efficacité des approches hybrides dans le domaine de l'analyse de données.

BIBLIOGRAPHIE

Abdulhafedh, A. (2021). Incorporating k-means, hierarchical clustering and pca in customer segmentation. *Journal of City and Development*, *3*(1), 12–30.

Androniceanu, A., Georgescu, I., Nica, E. et Popescu Gheorghe, H. (2018). A computational analysis of the romanian interregional policy. Dans *Innovation Management and Education Excellence through Vision 2020, Paper Presented at the 31st International Business Information Management Association Conference (IBIMA), Milan, Italy, April*, 25–26.

Arima, C., Hakamada, K., Okamoto, M. et Hanai, T. (2008). Modified fuzzy gap statistic for estimating preferable number of clusters in fuzzy k-means clustering. *Journal of bioscience and bioengineering*, *105*(3), 273–281.

Arora, S. et Barak, B. (2009). *Computational Complexity : A Modern Approach*. Cambridge University Press.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, *6*(1), 76–90.

Claesen, M. et Moor, B. D. (2015). Hyperparameter search in machine learning. *arXiv preprint arXiv :1502.02127*.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. et Stein, C. (2009). *Introduction to Algorithms* (3rd éd.). MIT Press.

Das, S., Abraham, A. et Konar, A. (2007). Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on systems, man, and cybernetics-Part A : Systems and Humans*, *38*(1), 218–237.

Davies, D. L. et Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2), 224–227.

Endres, S. (2010). *Global optimization of multiplicative functions*. (Thèse de doctorat). University of Cambridge.

Endres, S. et Sandgren, E. (2008). Restart strategies in global deterministic search methods. *Journal of Global Optimization*, *41*(2), 277–306.

Endres, S. et Sandholm, T. (2008). Behaviorally founded choice of focal points in coordination games. *Games and Economic Behavior*, *63*(1), 216–233.

Fletcher, R. (1970). A new approach to variable metric algorithms. *The computer journal*, *13*(3), 317–322.

Garey, M. R. et Johnson, D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co.

Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of*

computation, *24*(109), 23–26.

Hastie, T., Tibshirani, R. et Friedman, J. (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer.

Hyndman, R. J. et Athanasopoulos, G. (2018). *Forecasting : Principles and Practice*. OTexts.

Ikotun, A. M., Almutari, M. S. et Ezugwu, A. E. (2021). K-means-based nature-inspired metaheuristic algorithms for automatic data clustering problems : Recent advances and future directions. *Applied Sciences*, *11*(23), 11246.

Ikotun, A. M. et Ezugwu, A. E. (2022). Boosting k-means clustering with symbiotic organisms search for automatic clustering problems. *PLoS One*, *17*(8), e0272861.

Jones, D. R., Perttunen, C. D. et Stuckman, B. E. (1993). Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, *79*(1), 157–181.

Karo, I. M. K., MaulanaAdhinugraha, K. et Huda, A. F. (2017). A cluster validity for spatial clustering based on davies bouldin index and polygon dissimilarity function. Dans *2017 Second International Conference on Informatics and Computing (ICIC)*, 1–6. IEEE.

Kennedy, J. et Eberhart, R. (1995). Particle swarm optimization. Dans *Proceedings of ICNN'95-international conference on neural networks*, volume 4, 1942–1948. IEEE.

Ketchen, D. J. et Shook, C. L. (1996). The application of cluster analysis in strategic management research : an analysis and critique. *Strategic management journal*, *17*(6), 441–458.

Kraft, D. (1988). A software package for sequential quadratic programming. *DFVLR Forschungsbericht*.

Kwedlo, W. (2011). A clustering method combining differential evolution with the k-means algorithm. *Pattern Recognition Letters*, *32*(12), 1613–1621.

Lagarias, J., Reeds, J., Wright, M. et Wright, P. (1998). Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, *9*(1), 112–147.

Lampinen, J. A., Price, K. V. et Storn, R. M. (2005). *Differential evolution*. Springer.

Leary, R. H. et Doye, J. P. K. (2001). A transformation of variables that removes the driving mode from the vibrational normal modes of molecular systems. *Journal of Chemical Physics*, *115*(10), 4863–4869.

Lima, S. P. et Cruz, M. D. (2020). A genetic algorithm using calinski-harabasz index for automatic clustering problem. *Revista Brasileira de Computação Aplicada*, *12*(3), 97–106.

Lloyd, S. P. (1982). Least squares quantization in pcm. Dans *IEEE transactions on information theory*, volume 28, 129–137. IEEE.

Łukasik, S., Kowalski, P. A., Charytanowicz, M. et Kulczycki, P. (2017). Data clustering with grasshopper optimization algorithm. Dans *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 71–74. IEEE.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *1*(14), 281–297.

McCallum, A., Nigam, K. et Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. Dans *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 169–178.

Nelder, J. et Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, *7*(4), 308–313.

Niijima, S. et Okuno, Y. (2008). Laplacian linear discriminant analysis approach to unsupervised feature selection. *IEEE/ACM transactions on computational biology and bioinformatics*, *6*(4), 605–614.

Nocedal, J. et Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Omran, M. G., Salman, A. et Engelbrecht, A. P. (2006). Dynamic clustering using particle swarm optimization with application in image segmentation. *Pattern Analysis and Applications*, *8*, 332–344.

Papadimitriou, C. H. (1994). *Computational Complexity*. Addison-Wesley.

Pham, D. T., Dimov, S. S. et Nguyen, C. D. (2005). Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C : Journal of Mechanical Engineering Science*, *219*(1), 103–119.

Powell, M. (1998). Direct search algorithms for optimization calculations. *Acta Numerica*, *7*, 287–336.

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

Sedgewick, R. et Wayne, K. (2011). *Algorithms* (4th éd.). Addison-Wesley Professional.

Shanno, D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, *24*(111), 647–656.

Shi, C., Wei, B., Wei, S., Wang, W., Liu, H. et Liu, J. (2021). A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm. *EURASIP Journal on Wireless Communications and Networking*, *2021*(1), 1–16.

Sipser, M. (2005). *Introduction to the Theory of Computation* (2nd éd.). Thomson Course Technology.

Sokolova, M. et Lapalme, G. (2009). *A systematic analysis of performance measures for classification tasks*, volume 45.

Solorio-Fernández, S., Carrasco-Ochoa, J. A. et Martínez-Trinidad, J. F. (2016). A new hybrid filter–wrapper feature selection method for clustering based on ranking. *Neurocomputing*, *214*, 866–880.

Steinbach, M., Karypis, G. et Kumar, V. (2000). A comparison of document clustering techniques. Dans *KDD workshop on text mining*, volume 400, 525–526.

Stillinger, D., Stillinger, F. et Head-Gordon, M. (1995). Simplicial partitioning for the global optimization of molecular structures. *Physical Review E*, *52*(3), 2872.

Stillinger, D. H. et Weber, T. A. (1985). Packing structures and transitions in liquids and solids. *Science*, *225*(4666), 983–989.

Storn, R. et Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*, 341–359.

Sudheera, P., Sajja, V. R., Kumar, S. D. et Rao, N. G. (2016). Detection of dental plaque using enhanced k-means and silhouette methods. Dans *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 559–563. IEEE.

Sutton, R. S. et Barto, A. G. (2018). *Reinforcement Learning : An Introduction*. MIT press.

Thomas, J. C. R., Peñas, M. S. et Mora, M. (2013). New version of davies-bouldin index for clustering validation based on cylindrical distance. Dans *2013 32nd International Conference of the Chilean Computer Science Society (SCCC)*, 49–53. IEEE.

Tibshirani, R., Walther, G. et Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, *63*(2), 411–423.

Wales, D. J. et Doye, J. P. K. (1997). Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, *101*(28), 5111–5116.

Willmott, C. J. et Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, *30*(1), 79–82.

Yan, M. et Ye, K. (2007). Determining the number of clusters using the weighted gap statistic. *Biometrics*, *63*(4), 1031–1037.

Yang, J., Lee, J.-Y., Choi, M. et Joo, Y. (2019). A new approach to determine the optimal number of clusters based on the gap statistic. Dans *International Conference on Machine Learning for Networking*, 227–239. Springer.

Yang, L. et Shami, A. (2020). On hyperparameter optimization of machine learning algorithms : Theory and practice. *Neurocomputing*, *415*, 295–316.

Yuan, C. et Yang, H. (2019). Research on k-value selection method of k-means clustering algorithm. *J*, *2*(2), 226–235.

Žalik, K. R. et Žalik, B. (2011). Validity index for clusters of different sizes and densities. *Pattern Recognition Letters*, *32*(2), 221–234.

Zhang, G., Zhang, C. et Zhang, H. (2018). Improved k-means algorithm based on density canopy. *Knowledge-based systems*, *145*, 289–297.

Zhang, Y., Wang, W., Zhang, X. et Li, Y. (2008). A cluster validity index for fuzzy clustering. *Information*

Sciences, *178*(4), 1205–1218.

Zhao, W., Ma, H. et He, Q. (2009). Parallel k-means clustering based on mapreduce. Dans *Cloud Computing*, 674–679. Springer.