Application de l'algorithme BBO dans la classification non supervisée

BESSEDIK Malika **BENATCHBA Karima**

AMIEUR Nardjes
hn_amieur@esi.dz
AOUADJ Iheb Nassim
hi_aouadj@esi.dz
ASSENINE Mohamed Sami
hm_assenine@esi.dz
AZIZ Rezak
hr_aziz@esi.dz
BOUMENDIL Anais
ha_boumendil@esi.dz
MEDJADJI Chaimaa
gc_medjadji@esi.dz

Résumé
Le clustering est l'une des méthodes les plus renommées du Machine learning en plus d'être un problème NP-difficile faisant l'objet de nombreuses tentatives d'optimisation. L'application de métaheuristiques à ce problème peut se révéler très intéressante. Voilà pourquoi nous proposons à travers cet article d'appliquer la métaheuristique Biogeography-Based Optimisation (BBO) au problème du clustering. Nous avons adapté cette dernière à la classification non supervisée ce qui nous a permis d'obtenir des résultats très encourageants. Nous avons à titre d'exemple atteint une justesse de 93 % sur le dataset Iris avec un temps d'exécution très correct. L'article suivant décrit l'approche adoptée et les résultats obtenus.

Mots clés : Clustering, BBO, optimisation, K-means.

1 Introduction

La classification non supervisée (Clustering) est l'une des approches les plus importantes du Machine Learning. Elle vise à découvrir des groupes de données similaires et à les regrouper en clusters. Cette approche très utilisée peut se faire selon plusieurs méthodes regroupées en quatre familles : clustering hiérarchique, basé densité, basé grille et enfin par partitionnement. Le présent travail ne traitera que de la méthode du clustering par partitionnement. Cette dernière consiste à construire des partitions puis à les évaluer itérativement jusqu'à obtenir des clusters homogènes.

Le clustering peut être vu comme un problème NP-difficile imposant le recours à des méthodes d'optimisation. De nombreux travaux se sont d'ailleurs penchés sur l'application des métaheuristiques au clustering. Parmi ces métaheuristiques on trouve la méthode nommée Biogeography-Based Optimisation (BBO). Il s'agit d'une méthode inspirée de la biogéographie qui étudie la répartition spatiale des espèces vivantes et les causes de cette répartition. L'algorithme manipule une population composée d'îles où chaque île est une solution possible. La fitness de l'île est représentée par son HSI (Habitat Suitability Index), plus le HSI est faible plus la solution est meilleure. Un taux d'immigration (arrivées venant de l'extérieur) et un taux d'émigration (départs vers l'extérieur) sont associés à chaque île. Le principe général de l'algorithme consiste à démarrer d'une population générée aléatoirement. Après évaluation de la population initiale certains individus sont choisis pour subir une migration. Ceci permet de générer une nouvelle population sur laquelle on applique une mutation. Enfin, nous avons une phase de remplacement par les nouveaux individus. Le processus est répété un certain nombre de fois.

BBO est une métaheuristique évolutionnaire récente puisqu'elle n'a été introduite qu'en 2008. Elle reste néanmoins puissante et applicable à de nombreux problèmes. Étant une métaheuristique elle offre en premier lieu un bon compromis entre le temps d'exécution et la qualité de la solution. En second lieu, le fait de manipuler une population lui permet d'échapper aux optima locaux. De plus, de nombreux chercheurs ont constaté que BBO donne de

meilleures performances que les autres méthodes sur de nombreux problèmes. On peut citer les travaux de (Yang et al, 2015) qui traitaient de la classification d'images du cerveau en combinant les SVM et BBO. Cette combinaison a donné de meilleurs résultats que d'autres métaheuristiques telles que PSO (optimisation par essaim particulier).

BBO a donc prouvé son efficacité et son adaptation à de nombreux problèmes. Les raisons énumérées ci-dessus nous ont poussé à opter pour cet algorithme afin de l'appliquer au clustering.

2 Travaux relatifs

De nombreux travaux ont traité de l'application des métaheuristiques au Clustering. Nous allons en citer quelques-uns.

L'un des points cruciaux à traiter avant d'appliquer une métaheuristique est le choix du codage de la solution. Plusieurs codages existent pour le Clustering. Un bon codage doit minimiser l'espace mémoire utilisé, être simple à manipuler en plus d'éviter les solutions non réalisables qui dans le cas du partitionnement sont l'obtention d'une classe vide qui donnerait un partitionnement à $k-1$ classes au lieu d'un partitionnement à k classes. Parmi les codages proposés on peut citer celui de (Raghavan et Birchard, 1979) qui consiste en un vecteur associant à chaque élément le cluster auquel il appartient. Cette méthode est la plus simple mais elle présente l'inconvénient d'occuper trop d'espace mémoire. (D.R. Jones et M.A. Beltramo, 1991) suggère l'utilisation d'une matrice qui contient les éléments en lignes et les classes en colonnes. La case (i, j) prend la valeur 1 si l'objet i appartient à la classe j sinon elle prend la valeur 0. Cette représentation peut produire des solutions non réalisables. Un autre codage inspiré des k-means propose d'associer dans un vecteur chaque classe à son centre. Cette méthode présente l'avantage de donner des solutions toujours réalisables puisqu'il n'y a jamais de cluster vide.

Après avoir défini le codage adéquat, il est possible d'appliquer une certaine méthode. Parmi les métaheuristiques les plus souvent appliquées au clustering, nous pouvons citer les algorithmes génétiques (AG). Leur application consiste généralement en une hybridation avec l'algorithme K-means (MacQueen, 1967). On peut par exemple citer la méthode proposée par (Maulik et Al, 2000). Les centres des clusters sont initialisés aléatoirement avant d'appliquer les étapes du k-means. Ensuite, on procède à la sélection, croisement et mutation. Un second algorithme a été proposé par (Krishna et Al, 1999). La mutation consiste à affecter un objet aléatoire à un groupe voisin tandis que le croisement est remplacé par les opérations du k-means (calcul des centres et réaffectation). Dans les deux méthodes proposées la fonction d'évaluation est l'inverse de l'inertie intra-groupes.

Une seconde métaheuristique pouvant être utilisée est celle du recuit simulé. Elle fut appliquée par Allam et Attab en 2009. Cette application consiste de nouveau en une hybridation avec les k-means. Cette dernière donne les étapes suivantes : construire k classes aléatoirement et calculer leurs centres. Pour un certain nombre d'itérations appliquer une perturbation en choisissant un objet aléatoirement avant de l'affecter à une classe et de recalculer les centres. Il faut ensuite calculer la différence notée d des erreurs avant et après perturbation. On accepte la nouvelle partition avec une probabilité $e^{\frac{d}{T}}$, T étant la température. Cette approche offre souvent de bons résultats avec des paramètres adaptés allant même jusqu'à surpasser K-means.

L'application des colonies de fourmis au clustering a aussi fait l'objet de nombreux travaux. Le comportement des fourmis fascine encore les chercheurs et son analyse peut aboutir à des algorithmes très efficaces. (Benyamina, 2013) décrit un algorithme itératif nommé ACOClus qui combine l'ACO et les k-means. L'algorithme passe par les phases suivantes :

1. Initialisation aléatoire des partitions associées à chaque fourmi.
2. Calcul de la visibilité entre chaque couple d'objets (i, j).
3. Initialisé le taux de phéromone entre chaque couple d'objets (i, j) à une valeur choisie empiriquement.
4. Associer chaque fourmi à une partition aléatoire et appliquer k-means sur ces partitions.
5. Pendant les itérations de k-means choisir deux objets i, j et décider si i sera affecté à la même classe que j selon une probabilité calculée en fonction du taux de phéromone et de la visibilité.

Enfin, l'application de BBO au clustering a fait l'objet de nombreux travaux. (Vijay et al, 2011) proposent d'initialiser les centres de k-means en utilisant le BBO. En effet, l'initialisation aléatoire des centres peut dégrader les performances et l'utilisation de BBO peut apporter une solution très satisfaisante. (Pal et Saraswat, 2017) proposent l'approche inverse en effectuant une initialisation avec k-means avant d'appliquer BBO. Il existe d'autres approches toutes aussi intéressantes et offrant de très bonnes performances.

3 Application du BBO à d'autres problèmes

L'algorithme BBO a été appliqué à de nombreux problèmes au vu de sa puissance. Nous avons cité plus haut l'hybridation de BBO et des SVM pour la classification d'images du cerveau menée dans les travaux de (Yang et al, 2015). La métaheuristique peut aussi être appliquée pour les problèmes NP-difficile classiques tels que le problème du voyageur de commerce. (Boussaid , 2013) l'applique à l'optimisation continue et notamment à l'allocation optimale de puissance dans un réseau de capteurs sans fil.

4 Approche adoptée

Le présent travail traite de l'adaptation de la métaheuristique BBO au problème du clustering. L'application de cette dernière peut en effet optimiser le problème de classification non supervisé et même offrir une méthode avec de très bonnes performances vu la

puissance de BBO. Afin de réaliser cette adaptation, il a fallu procéder à un certain nombre de choix en commençant par la génération de la population initiale. Nous avons opté pour l'utilisation de k-means pour la génération de cette dernière ce qui améliore d'autant plus les performances. En effet, cette hybridation permet à l'algorithme BBO de démarrer par un ensemble de solutions réalisables avant de les améliorer au fil d'un nombre d'itérations fixé. Dans chacune de ces itérations, l'algorithme procédera à l'évaluation de la population courante à l'aide d'une fonction objectif bien choisie avant de sélectionner les habitats qui subiront une migration puis une mutation. Ces opérateurs donneront alors de nouveaux individus qui remplaceront des éléments de la population en cours. BBO est guidé par un ensemble de paramètres que nous ajusterons pour tirer le meilleur de la métaheuristique.

Pour expliciter le travail effectué, nous commencerons par rappeler la formulation du problème du clustering avant de passer à la description détaillée de l'approche proposée. Nous décrirons donc comment nous avons adapté BBO au clustering en indiquant quel est l'encodage utilisé, quelle est la fonction objectif choisie avant de détailler chaque étape de l'algorithme. Afin de vérifier le bon fonctionnement de la méthode proposée, nous effectuerons quelques tests sur des jeux de données. Nous indiquerons comment les paramètres ont été ajustés pour chaque dataset avant d'évaluer les performances de notre méthode à travers l'analyse des résultats obtenus. Nous terminerons par une comparaison de la méthode avec K-means pour voir le réel apport de l'approche proposée.

5 Formulation du problème

Le problème du clustering vise à regrouper un ensemble de données en entrées en des groupes appelés clusters. Chaque donnée sera affectée à une certaine classe. Les dites classes ne sont pas connues au départ puisqu'il s'agit d'une classification non supervisée. Le clustering consiste donc à regrouper les données dans des clusters de manière à ce que ces derniers soient le plus homogènes possibles. Les données d'un même cluster seront très similaires tandis que les données de deux clusters différents ne doivent pas être semblables. Le clustering repose donc sur une mesure de similarité. Plusieurs approches existent pour le problème du clustering mais comme indiqué plus haut, nous ne traiterons que du clustering avec partitionnement dans le présent travail.

Les méthodes par partitionnement sont généralement des méthodes itératives. Elles démarrent de partitions initiales qui seront modifiées tout au long des itérations dans le but de maximiser la similarité entre les objets d'une même classe et de maximiser la dissimilarité entre des objets de deux classes différentes

Nous pouvons formaliser ce concept en considérant un jeu de m données noté X. Chaque donnée est décrite à l'aide de n attributs. Ces derniers permettent de décrire les données et donc de détecter les objets similaires. On peut donc avoir une matrice m x n où les lignes sont les données et les colonnes leurs attributs. Le but est de construire k clusters $C_1, ..., C_k$ où chaque cluster C_i contient des données homogènes. Ceci se traduit par la maximisation de la variance interclasse et la minimisation de la variance intra-classe. Chaque cluster construit représente une partition et les propriétés suivantes sont donc vérifiées :

$$C_i \neq \phi \ \ pour \ \ 1 \leq i \leq k$$
$$C_i \cap C_j \ \ pour \ \ i \neq j \ \ et \ \ 1 \leq i \leq k \ \ et \ \ 1 \leq j \leq k$$
$$C_1 \cup C_i \cup ... \cup C_k = X$$

6 Description de l'approche proposée

Comme indiqué plus haut BBO est une métaheuristique manipulant une population de solutions candidates dont la recherche est guidée par les caractéristiques de ladite population. Nous pouvons rappeler que le principe général de l'algorithme consiste à démarrer d'une solution initiale que l'on évaluera avant de procéder à un ensemble d'itérations durant lesquelles certains individus seront sélectionnés afin de subir les opérations de migration et de mutation. Cette dernière se fait selon un taux de mutation défini. Puis vient une phase de remplacement qui consiste à substituer certains parents par les nouveaux individus générés. Ainsi une nouvelle population de taille fixe est générée à chaque itération permettant d'améliorer au fur et à mesure la solution. BBO s'appuie donc sur les composants suivants :

Habitat:

L'algorithme BBO manipule une population d'individus appelés îles (ou habitats). Chaque île représente une solution possible au problème à résoudre. La *fitness* de chaque île est déterminée par son HSI (Habitat Suitability Index), une mesure de la qualité d'une solution candidate, et chaque île est représentée par des SIV (Suitability Index Variables). Une bonne solution au problème d'optimisation est une île avec un faible HSI. Les SIV représentent les caractéristiques de la population, dans le cas du clustering on peut définir SIV comme étant les coordonnées des centres des clusters.

Opérateurs d'émigration et de migration:

L'opérateur d'émigration est appliqué au niveau de chaque habitat Hi de la population, il consiste à remplacer la valeur d'un habitat par une valeur SIV choisie aléatoirement de l'espace de solutions. Chaque habitat possède son propre taux d'immigration et son taux d'émigration.

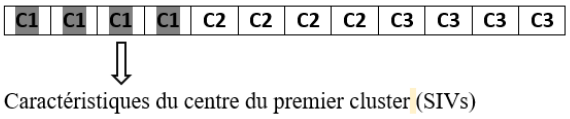
Afin d'atteindre notre objectif d'optimisation du clustering, nous avons adapté l'algorithme BBO au problème de classification non supervisée. Nous allons dans ce qui suit décrire l'approche suivie pour mettre en place cette adaptation. Afin de bien illustrer la méthode proposée nous allons exposer les différentes étapes de l'algorithme appliquées au jeu de données Fleurs IRIS. Ce dernier comprend un total de 150 observations réparties de manière égale entre les trois espèces de

fleurs d'iris (setosa, virginica et versicolor). Quatre caractéristiques sont mesurées pour chaque observation (la longueur et la largeur du sépale et du pétale).

6.1 Encodage de la solution

Chaque habitat représente une solution au problème du clustering. Chaque solution est encodée à l'aide d'une table de centres de clusters. Chaque centre est représenté dans l'espace des variables c'est-à-dire que si nous avons N attributs en entrée nous aurons N coordonnées pour chaque centre.

On peut donc représenter chaque habitat par une séquence de nombres réels représentant les K centres de clusters. Pour un espace à N dimensions (dans notre cas iris $N = 4$ dimensions), la longueur d'un habitat est de $N * K$ SIVs $= 4 * 3 = 12$, où les N premières positions (SIV) représentent les N dimensions du centre du premier cluster, les N positions suivantes représentent celles du centre du deuxième cluster et ainsi de suite. Le schéma suivant décrit l'encodage utilisé :



6.2 Fonction objectif utilisée

Dans notre étude, le nombre de clusters est fixé au début. Nous cherchons à minimiser la distance entre les points appartenant à un cluster et le centre associé à celui-ci. Nous allons donc affecter chaque point au cluster le plus proche avant de calculer la somme des distances entre les points d'une classe et son centre en utilisant une matrice des distances. Ceci définit donc notre fonction objectif. Nous pouvons illustrer ce principe par l'exemple suivant :

Voici donc la matrice des distances :

$$\begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}$$

Distance entre individu i et le centre j : d_{ij}

Pour chaque individu du dataset nous gardons le centre le plus proche :

$$\begin{pmatrix} d_{11} \\ d_{23} \\ d_{32} \end{pmatrix}$$

Le but et de minimiser la somme des distances entre chaque individu et le centre qui lui est proche, la fonction objectif ou l'HSI est calculé comme suit :
$HSI = Distanceentreindividu1etlecentre1 +$
$Distanceentreindividu2etlecentre3 +$
$Distanceentreindividu3etlecentre2$

6.3 Population initiale

Afin de générer la population initiale de BBO nous avons opté pour l'utilisation de l'algorithme K-means. Chaque habitat est donc initialisé par les coordonnées des centres des k clusters retournés par K-means suivant l'encodage préalablement décrit. Pour chaque habitat, nous faisons varier les paramètres de k-means pour différencier les solutions. Ce choix de population initiale permet de démarrer d'une population de solutions réalisables en plus d'offrir une hybridation séquentielle des deux méthodes.

6.4 Evaluation

L'évaluation consiste à appliquer la fonction objectif (ou HSI) présentée plus haut à chaque habitat. Par la suite les habitats sont triés suivant l'ordre croissant de leur HSI.

6.5 Opérateur de migration et d'émigration

Après l'évaluation il faut appliquer la migration permettant de générer une nouvelle population à chaque itération de BBO. Le nombre et la destination des migrants sont décidés en fonction des taux d'émigration et de immigration. Pour chaque habitat on calcule le taux d'immigration λ - arrivées venant de l'extérieur - et son taux d'émigration μ- départs vers l'extérieur

$$\mu = \frac{s}{smax}\left(\frac{le\,nombre\,s\,especes\,presentes\,dans\,une\,ile}{capacite\,maximale\,de\,l'ile}\right)$$
$$\lambda = 1 - \mu$$

Ces deux paramètres dépendent du nombre d'espèces présentes sur l'île. En effet, le taux d'immigration lui est inversement proportionnel tandis que le taux d'émigration augmente avec ce nombre d'espèces. Dans notre cas pour chaque habitat nous avons attribué au début de l'algorithme une valeur de μ qui n'est pas mise à jour :

$$\mu = \frac{(la\,taille\,de\,la\,population + 1 - (indice\,de\,la\,population))}{((la\,taille\,de\,la\,population + 1)}$$
$$\lambda = 1 - \mu$$

6.6 Mutation

Selon une probabilité de mutation (un paramètre en entrée de l'algorithme) on peut ajouter des caractéristiques aléatoires au niveau de chaque population, l'intervalle des caractéristiques aléatoires est un paramètre. Pour chaque population et pour chaque variable on calcule la probabilité de mutation et on modifie la valeur de la variable par une valeur aléatoire générée dans l'intervalle des valeurs $[borne_inf, borne_sup]$.

6.7 Élitisme

La stratégie élitiste consiste à conserver un nombre noté "keep" des meilleurs individus à chaque génération. Ainsi l'élitisme empêche l'individu le plus performant de disparaître au cours du remplacement.

Après chaque évaluation de la performance des individus, les "keep" meilleurs individus de la génération précédente sont réintroduits dans la population en

Paramètres	Notation
Taille de la population	nPop
Paramètre d'élitisme	keep
Taux min d'immigration	I
Taux max d'émigration	E
Nombre d'itérations	Iterations
Intervalle des valeur	Borne-inf Borne-sup
Dimension du problème	nDim
Probabilité de mutation	pmutate

Table 1. Tableau résumé des noms des variables de l'algorithme BBO

remplaçant les habitats de valeurs HSI la plus grande qui sont donc les moins bonnes solutions. L'amélioration des habitats continue à chaque itération jusqu'à atteindre le critère d'arrêt.

6.8 Paramètres de l'algorithme BBO

Étant donné que BBO est une métaheuristique, l'algorithme dispose d'un ensemble de paramètres qu'il faut bien ajuster afin d'améliorer la qualité de la solution obtenue. Nous pouvons résumer ces paramètres par le tableau 6.8.

Après avoir défini les différentes étapes de BBO et ses paramètres nous pouvons résumer l'approche à travers le pseudo algorithme suivant :

Data: Dataset X et nombre de clusters K
Result: Ensemble des centres de clusters optimaux
initialization
foreach H_i **do** $Habitat\,(H_i) = k - means\,(X,K)$;
Garder les deux meilleurs habitats
foreach *habitat* H_i *de la population* P **do** **foreach** *SIV de l'habitat* **do** **if** $nb\_alea < \lambda$ **then**
    Sélectionner les habitats à émigrer selon μ;
    $H_i\,(SIV_k) = H_j\,(SIV_k)$ *pour* $k = 1,2,...,m$ *SIV*;
else
    $H_i\,(SIV_k) = H_j\,(SIV_k)$;
end
;
;
foreach *habitat* H_i *de la population* P **do** **if** $nb\_alea < pmutate$ **then**
    Sélectionner les habitats à émigrer selon μ;
    $H_i\,(SIV_k) = nb\_aleadansledomaine$;
end
;
Modifier la population P

Algorithm 1: Algorithme BBO

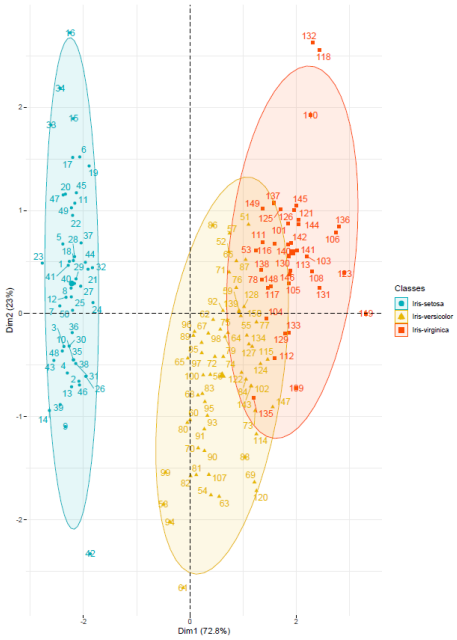


Fig. 1. Visualisation des composantes principales de dataset Iris en 2D

7 Tests et Résultats

7.1 Présentation des instances utilisées

Afin d'évaluer les bonnes performances de la méthode proposée, il est nécessaire d'effectuer des tests sur des jeux de données bien choisis. Nous avons dans notre cas opté pour l'utilisation des deux datasets Iris et hearth. Nous allons présenter brièvement chacun de ces derniers.

- Fleurs d'Iris : le dataset Iris regroupe un ensemble de 150 observations de trois espèces de la fleur : Iris setosa, Iris virginica et Iris versicolor. Pour chacune des trois espèces, 50 observations ont été collectées ce qui fait que nous avons une répartition équitable entre les classes. Sur chaque observation les quatre caractéristiques suivantes sont mesurées : la largeur du sépale et sa longueur ainsi que la largeur du pétale et sa longueur en centimètres. Afin de visualiser le dataset nous avons effectué une analyse en composantes principales ACP. Cette dernière indique que l'utilisation de deux axes suffit pour avoir 95.8% de l'information. La figure 1 nous permet de visualiser le dataset en deux dimensions suivant la répartition des observations en classes.

- Hearth : regroupe un ensemble d'attributs selon lesquels il est possible de prédire si un individu souffre d'une maladie cardiovasculaire. Le dataset utilisé regroupe 14 attributs tels que : l'âge, le sexe, la douleur de poitrine, la tension artérielle au repos, serum cholestrol, glycémie à jeun ... Nous avons de nouveau effectué une ACP pour visualiser notre jeux de données. Pour que la visualisation soit claire, nous avons dû nous limiter à deux dimensions comme le montre la figure 2

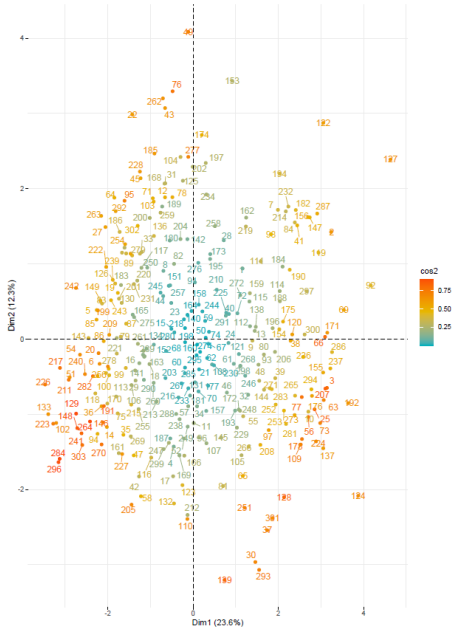


Fig. 2. Visualisation des composantes principales de dataset Hearth en 2D

7.2 Calibrage des paramètres

Étant donné que BBO est une métaheuristique, il faut ajuster un certain nombre de paramètres pour en tirer les performances optimales. Une manière de calibrer ces derniers consiste à effectuer une étude empirique à travers une série de tests dans laquelle plusieurs configurations de paramètres sont testées.

Nous avons donc effectué plusieurs tests en faisant varier plusieurs paramètres. Afin de mesurer la qualité de chaque configuration nous avons eu recours à deux métriques :

1. Accuracy (justesse) : Elle mesure le pourcentage de prédictions correctes. Cette métrique nous permettra de comparer les classes que prédit notre modèle aux classes réelles.
2. Indice de Rand : mesure la similarité entre deux partitions d'un ensemble. Comme il est particulièrement adapté au clustering, il nous permettra d'évaluer la qualité des partitions obtenues.

Nous avons aussi calculé le nombre de fausses prédictions pour chaque configuration de paramètres afin d'illustrer de manière plus explicite les résultats de chaque combinaison.

Le calibrage des paramètres a été effectué pour chaque dataset. Nous allons commencer par résumer les résultats obtenus pour le dataset Iris à travers le tableau 2

A partir du tableau précédent, nous pouvons conclure que pour le dataset Iris la meilleure configuration n'est autre que la suivante : une population de taille 50, un nombre d'itérations de 200, un élitisme de 2 et l'application d'une probabilité de mutation de 0.01. En effet, on peut observer que cette combinaison donne seulement 11 observations mal classées, une accuracy de 0.926 et enfin un indice de Rand de 0.802. Les résultats sont aussi influencés par la population initiale que nous avons ici généré avec k-means. La première moitié de la population est initialisée avec k-means (random_state=3) qui donne de bons résultats sur ce dataset avec une faible altération, la deuxième moitié est initialisée avec k-means (random_state variant) avec une faible altération.

Pour le dataset hearth les différentes configurations des paramètres donnent les résultats dans le tableau 3

Le tableau précédent nous permet de conclure que la meilleure configuration est une population de taille 50, un élitisme de 2, un nombre d'itérations de 100 et pas de mutation. En plus de ces paramètres, nous avons conclu qu'il est préférable d'initialiser la première partie de la population avec k-means (random_state=8) sans altération et la seconde moitié par avec k-means (random_state variant) sans aucune altération.

7.3 Etude des performances de l'approche proposée

Selon les tests effectués plus haut nous pouvons énoncer que l'algorithme proposé donne de bons résultats en particulier sur le dataset Iris. On peut en effet voir qu'avec des paramètres bien ajustés nous pouvons atteindre une justesse de 0.93 ce qui signifie que 93 % des observations ont été correctement classées. Un indice de Rand de 0.82 a été atteint preuve que l'homogénéité des clusters obtenus est élevée. Pour le jeu de données hearth, la combinaison optimale de paramètres nous permet d'atteindre une justesse de 84% et un Rand score de 0.47. Bien que les performances soient moins bonnes comparées au premier dataset, elles demeurent correctes. Ceci nous permet de dire que l'adaptation de BBO au clustering se révèle très intéressante et donne des résultats satisfaisants.

Pour ce qui est du comportement de l'algorithme, il dépend de la combinaison de paramètres choisie. D'abord, la taille de la population n'a pas à être trop grande. Une population de taille 50 nous donne généralement de bons résultats et l'augmenter d'autant plus ferait qu'il y aurait trop de solutions à manipuler. Ensuite, pour le nombre d'itérations il faut avoir un nombre assez élevé pour bien explorer l'espace de recherche et assurer une certaine intensification sans pour autant avoir beaucoup d'itérations puisque cela dégraderait le temps d'exécution. Pour ce qui est de l'élitisme nous avons convenu que fixer le paramètre à 2 donnait de bons résultats. Cela permet de ne pas perdre les meilleures solutions sans pour autant restreindre l'apparition de nouvelles au long des itérations. Enfin, l'application de la mutation peut offrir une certaine diversification mais le choix de son utilisation dépend du dataset et seule une étude empirique peut nous orienter sur ce choix.

En plus du choix des paramètres, le second facteur influençant les performances est la population initiale. L'utilisation de k-means pour générer cette dernière permet à BBO de démarrer d'un ensemble de solutions réalisables en plus d'offrir une hybridation ce qui constitue un point positif. En effet, démarrer de solutions potentielles est généralement meilleur que le fait de commencer par une population initiale aléatoire. Cependant, la performance de

Taille population	Élitisme	Mutation	Nombre d'itérations	Rand score	Accuracy	Mal classé	HSI
50	2	oui	200	0.802	0.926	11	102.76
50	2	non	200	0.729	0.9133	13	99.24
100	2	oui	200	0.786	0.92	12	102.49
100	2	non	200	0.741	0.9	15	97.62
50	5	oui	200	0.716	0.8933	16	97.72
50	5	non	200	0.77	0.906	14	97.79
50	3	oui	100	0.75	0.9	15	98.51
100	2	non	100	0.785	0.9133	13	98.46

Table 2. Tableau comparatif des résultats des méthodes BBO et k-means sur le dataset Iris

Taille population	Élitisme	Mutation	Nombre d'itérations	Rand score	Accuracy	Mal classé	HSI
50	2	oui	200	0.2729	0.7623	72	92788.4
50	2	non	200	0.47	0.84	47	92880.2
100	2	oui	200	0.2729	0.7623	72	92788.4
100	2	non	200	0.47	0.84	47	92880.2
50	5	oui	200	0.2729	0.7623	72	92880.2
50	5	non	200	0.47	0.84	47	92880.2
50	2	oui	50	0.43	0.83	51	92880.2
50	2	non	100	0.47	0.84	47	92880.2

Table 3. Tableau comparatif des résultats des méthodes BBO et k-means sur le dataset heart

BBO devient dépendante à celle de k-means. Nous avons en effet remarqué que BBO était fortement impacté par le résultat de k-means qui dépend à son tour de l'instance en entrée. Pour le dataset Iris k-means donne de très bons résultats ce qui affecte positivement les résultats de BBO. En revanche, pour hearth les performances de k-means sont moins bonnes ce qui cette fois-ci influence négativement BBO. Ceci est donc relatif au jeu de données qui est plus complexe dans le cas de hearth qui regroupe 14 attributs au lieu de 4 pour Iris.

Pour ce qui est du temps d'exécution, le résultat obtenu est assez correct puisque l'algorithme nécessite environ 69 secondes pour s'exécuter sur le dataset Iris.

En résumé, l'adaptation de BBO au clustering peut s'avérer très intéressante. Il faut seulement prendre le soin d'ajuster les paramètres selon le dataset en entrée. Il faut aussi bien fixer la population initiale encore une fois selon le jeu de données considéré.

7.4 Etude comparative entre l'approche proposée et k-mean

Pour l'étude comparative nous nous sommes principalement basées sur les deux datasets déjà présentés: Iris et hearth.

Nous avons donc exécuté k means et BBO sur les deux dataset et nous avons utilisés des metriques pour la comparaison:

- la justesse des résultats par rapport aux données réelles.
- l'indice de rand: il s'agit d'une mesure de similarité entre deux partitions d'un ensemble. son principe est de calculer le taux d'accord entre deux partitions et cela indépendamment des labels des partitions

7.4.1 Kmeans et BBO vs la classification originale sur Dataset iris

A travers les graphes dans fig.3 et les résultats numériques **??**. On remarque que les individus se situant au niveau de la frontière entre les deux classes, qui ont été mal classés par le k means sont bien classés avec bbo. Ceci se voit bien à travers l'accuracy et le rand score qui sont plus significatif dans le cas de BBO. En comparant les valeurs de la fonction objective HSI, le bbo a une valeur plus grande. Ceci justifie le fait qu'il a bien classé même les individus loins du centre contrairement au k means.

7.4.2 Kmeans et BBO vs la classification originale sur Dataset iris

A travers les graphes dans fig.4 et les résultats numériques **??**, on remarque que dans le cas de ce dataset les classes de ce dataset ne sont pas bien séparées ce qui rend le clustering plus difficile. Les individus se situant au niveau de la frontière entre les deux classes, qui ont été mal classés par le k means sont bien classés avec bbo. Ceci peut se confirmer par les résultats numériques a travers l'accuracy et le rand score.

En comparant les valeurs de la fonction objective HSI, le bbo a une valeur plus grande. Ceci justifie le fait qu'il a bien classé même les individus loins du centre contrairement au kmeans.

8 Répartition des tâches

Tâche	Membres responsables
Etat de l'art	BOUMENDIL Anais
Implémentation de l'algorithme	AMIEUR Nardjes
	MEDJADJI Chaimaa
Algorithme k-means et ACP	ASSENINE Mohamed Sami
Calibrage des paramètres	AZIZ Rezak
Tests	AMIEUR Nardjes
	MEDJADJI Chaimaa
Rédaction de l'article	AOUADJ Iheb Nassim
	BOUMENDIL Anais

Table 4. Tableau de la répartition des tâches entre les membres de l'équipe

9 Références Bibliographiques

1 U. Maulik [et S. Bandyopadhyay. Genetic algorithm-based clustering technique. Pattern Recognition.

2 K. Krishna et M. Narasimha Murty. Genetic k-means algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B.

3 Laetitia Jourdan. Métaheuristiques pour l'extraction de connaissances : Application à la génomique Université des Sciences et Technologie de Lille - Lille I, 2003. Français.

4 Allam Farida et Attab Saida. Hybridation de la méthode des k-means avec le recuit simulé, Université de Mouloud Mammeri, Tizi-Ouzou, 2009.

5 Arbia Djamila. Métaheuristiques appliquées à la classification non supervisée de données, Université Mohamed Boudiaf, M'SILA, 2019.

6 Benyamina Ahmed. Application des algorithmes de colonies de fourmis pour l'optimisation et la classification des images, Université des sciences et de la technologie d'Oran Mohamed Boudiaf, 2013.

7 Yang, G.; Yang, J. Automated classification of brain images using wavelet-energy and biogeography-based optimization. Multimedia Tools and Applications, 2015.

8 Vijay Kumar, Jitender Kumar Chhabra, Dinesh Kumar. Initializing Cluster Center for K-Means Using Biogeography Based Optimization, 2

9 Raju Pal, Mukesh Saraswat. Data clustering using enhanced biogeography-based optimization, 2017.

10] Ilhem Boussaid. Perfectionnement de métaheuristiques pour l'optimisation continue, Université Paris-Est, 2013.

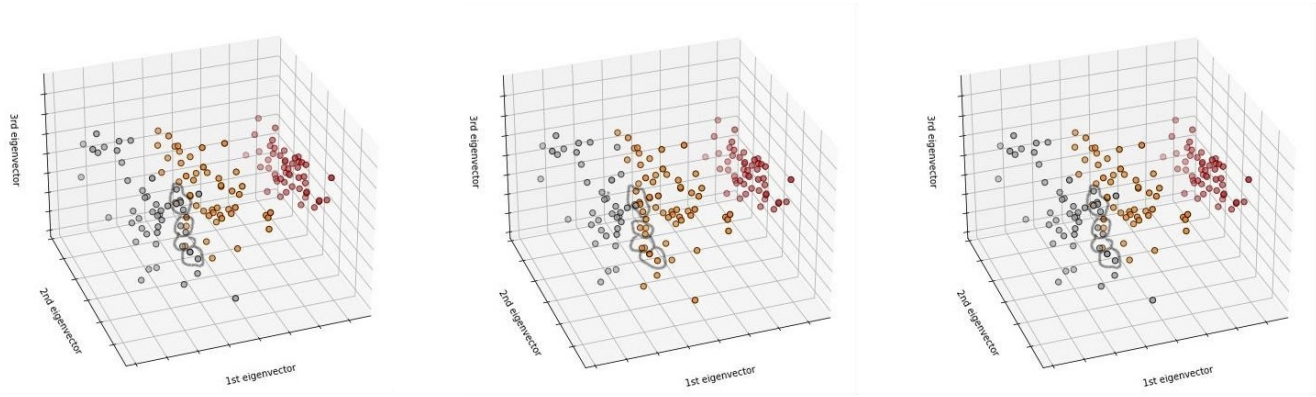


Fig. 3. Visualisation de la classification originale, par k-means et par BBO de dataset iris en 3D

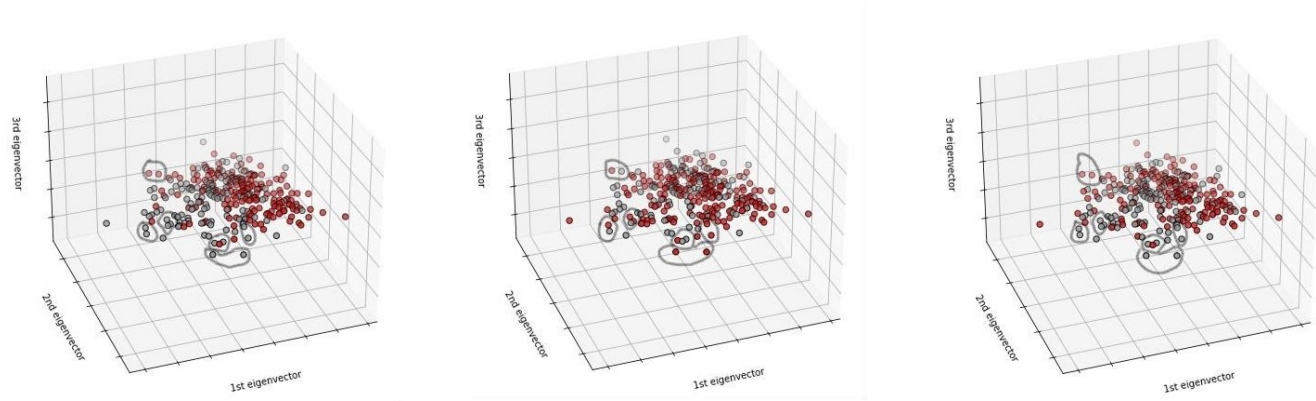


Fig. 4. Visualisation de la classification originale, par k-means et par BBO de dataset Hearth en 3D

	Mal classifié	Accuracy	Hsi	Rand_score
Kmeans	16	0.89333	97.204	0.7302
BBO	10	0.9333	99.823	0.7714

Table 5. Évaluation des métriques sur les résultats de BBO et K-means sur IRIS Dataset

	Mal classifié	Accuracy	Hsi	Rand_score
Kmeans	128	0.577	11931.44	0.0205
BBO	72	0.7623	92878.48	0.2729

Table 6. Évaluation des métriques sur les résultats de BBO et K-means sur Hearth Dataset