Actes Coll. Insectes Sociaux, 13: 43-52 (2000)

CLASSIFICATION NON SUPERVISÉE PAR UNE POPULATION DE FOURMIS ARTIFICIELLES

Nicolas MONMARCHÉ, Gilles VENTURINI, Mohamed SLIMANE

Laboratoire d'Informatique, Université de Tours,
École d'Ingénieurs en Informatique pour l'Industrie (E3i),
64 av. Jean Portalis, 37200 Tours
{monmarche,slimane,venturini}@univ-tours.fr
Téléphone: +33-2-47-36-14-14
Fax: +33-2-47-36-14-22

Résumé. Nous présentons dans ce travail un nouvel algorithme hybride pour la classification de données. Cet algorithme s'inspire de la capacité des fourmis à rassembler, par exemple, leurs œufs en tas. Nous cherchons à découvrir automatiquement des classes dans des données numériques sans connaissances a priori du nombre de classes, sans partition initiale et sans qu'il soit nécessaire d'effectuer un paramétrage complexe. Les fourmis se déplacent sur une grille à deux dimensions et peuvent ramasser ou déposer des objets afin de partitionner les données. L'algorithme des centres mobiles est utilisé pour améliorer les résultats obtenus par les fourmis. Nous avons aussi introduit quelques raffinements comme une hétérogénéité des paramètres des fourmis, ou une mémoire locale à chaque fourmi. L'algorithme obtenu, noté AntClass a été testé avec succès sur des base de données numériques standards et réelles.

Mots Clés. *Fourmis artificielles, classification non supervisée, centres mobiles.*

Abstract. Unsupervised clustering by a population of artificial ants.
We present in this work a new hybrid algorithm for data clustering. This algorithm is inspired by the ability of ants to cluster, for instance, their eggs into heaps. This algorithm discovers automatically clusters in numerical data without any prior knowledge of a possible number of classes, without any initial partition and without complex parameter settings. Ants move on a 2D board and may pick up or drop objects to build a partition of data. The Kmeans algorithm is used to improve the result obtained by the ants. We also use other refinements such as an heterogeneous population of ants and a local memory for each ant. The final algorithm, called AntClass, has been applied with success on standard and real databases.

Keywords. *Artificial ants, unsupervised clustering, Kmeans algorithm.*

INTRODUCTION

Depuis quelques années, de nombreux travaux ont mis en valeur l'efficacité d'une approche stochastique basée sur les colonies de fourmis pour résoudre des problèmes divers. Cela concerne par exemple des problèmes d'optimisation combinatoire tels que le problème du voyageur de commerce (Dorigo et Gambardella, 1997), le problème de l'assignement quadratique (Gambardella et coll., 1997), les problèmes de routage dans les réseaux (Di Caro et Dorigo, 1997). Les problèmes d'optimisation numériques ont aussi été abordés avec des fourmis artificielles comme dans (Bilchev et Parmee, 1996), ainsi qu'en robotique (Deneubourg et coll., 1991; Goss et Deneubourg, 1991).

La classification fait aussi partie de ces problèmes dans lesquels les fourmis suggèrent des heuristiques très intéressantes pour les informaticiens. En se basant sur les travaux existants (Deneubourg et coll., 1991; Lumer et Faieta, 1994; Kuntz et coll., 1997), nous contribuons dans cet article à l'étude des fourmis classifieuses du point de vue de la découverte de connaissances, avec comme objectif de résoudre des problèmes réels. Dans de tels problèmes, nous considérons qu'un expert du domaine a collecté un ensemble de données et qu'il aimerait se voir proposer une partition de ses données en des classes pertinentes, sans avoir à fournir une partition de départ et sans connaître le nombre de classes qui seront nécessaires. Nous désirons éviter un paramétrage complexe car dans la plupart des cas les experts du domaine ne sont pas des informaticiens et ne savent par conséquent pas comment régler les paramètres d'une méthode de classification.

MATÉRIEL ET MÉTHODES

Les travaux initiaux concernant la classification d'objets avec des fourmis artificielles ont été menés par Deneubourg et ses collègues (Deneubourg et coll., 1991). Dans ces travaux, une colonie de robots est capable de classer des objets en utilisant des heuristiques issues des fourmis et sans contrôle centralisé. Ce travail originel a été utilisé comme base pour des études utilisant des fourmis pour classer des objets.

La première application importante dans le monde réel a été développée pour les technologies VLSI (Kuntz et coll., 1997). Le problème consiste ici à trouver une partition d'un graphe. Pour ceci, le problème de partitionnement est traduit en problème de classification résolu par des fourmis artificielles. Les fourmis sont capables de ramasser/déposer des objets sur une grille à deux dimensions en suivant une mesure de densité locale des objets similaires. Les résultats peuvent être évalués par exemple avec une mesure d'entropie spatiale.

La deuxième application de fourmis classifieuses est plus proche des travaux que nous présentons. Il s'agit de laisser les fourmis classer les données d'un ensemble (Lumer et Faieta, 1994). Les objets sont initialements éparpillés aléatoirement sur une grille à deux dimensions. Les fourmis utilisent une mesure de densité locale d'objets similaires pour prendre leurs décisions. L'intérêt de ce travail est d'introduire les principes de classification par des fourmis dans des problèmes d'analyse des données.

Notations et heuristiques de AntClass

Les objets et leur métrique. Nous supposons qu'un ensemble $E = \{O_1, ..., O_n\}$ de n données ou objets ont été collectés par un expert du domaine. Chaque objet est un vecteur de k valeurs numériques $v_1, ..., v_k$ qui correspondent aux k attributs numériques $A_1, ..., A_k$.

Un paramètre important des algorithmes de classification est de savoir comment mesurer la similarité entre deux objets. Nous utiliserons par défaut, la distance euclidienne entre deux objets, que nous noterons D par la suite. Cependant, AntClass peut utiliser d'autres types de distance telles que les mesures de Minkowski ou de Hamming. Nous noterons D_{max} la distance maximale entre deux objets de E : $D_{max} = \max_{O_i, O_j \in E} D(O_i, O_j)$.

Initialement, tous les objets sont éparpillés aléatoirement sur une grille à deux dimensions décrite dans le paragraphe suivant. Les fourmis seront capables de ramasser ou de déposer des objets afin de créer des tas d'objets qui correspondent aux classes dans notre processus de classification.

La grille. Nous faisons évoluer les fourmis sur une grille à deux dimensions, de $m \times m$ cases. Cette grille est toroïdale afin que les fourmis passent d'un bord à l'autre. La taille de la grille est déterminée automatiquement : il faut que le nombre de cases soit au moins égal au nombre d'objets ($m^2 \geq n$). Si la grille est trop grande, les fourmis vont perdre beaucoup de temps à chercher les objets. Un bon compromis est de fixer la taille de la grille en fonction du nombre d'objets avec la relation suivante : $m^2 = n \times 4$.

Les tas d'objets. Un tas H est un ensemble d'au moins deux objets et il est localisé sur une case donnée comme il est montré dans la figure 1 à droite. Sur

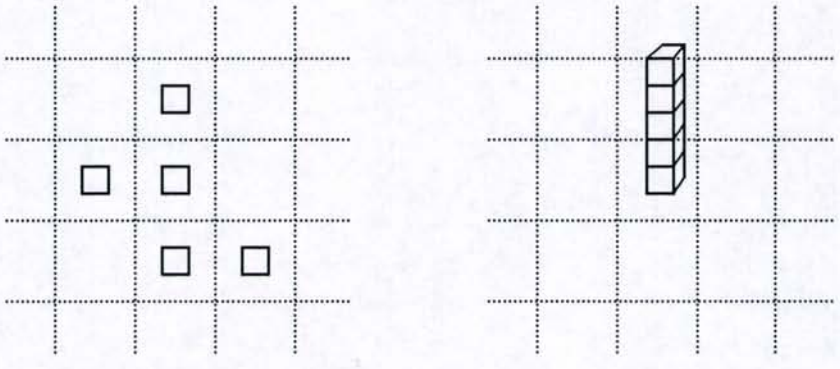

FIG. 1 – *Sur la gauche, nous montrons une classe selon la représentation de Lumer et Faieta (1994). A droite, la représentation que nous avons adoptée pour AntClass. On the left side we are presenting a cluster as in Lumer et Faieta (1994). On the right side, we show the cluster representation we have adopted for AntClass.*

la gauche de la figure 1, nous avons représenté une classe trouvée selon Lumer et Faieta (1994). Une classe a alors une certaine superficie et deux classes peuvent être en contact et poser un problème pour les séparer. L'amélioration majeure de la représentation que nous avons adoptée, vient de la facilité à identifier une classe. Un autre avantage de cette technique est qu'elle nous permet de définir des heuristiques plus précises pour la dépose ou la saisie d'objets sur un tas. Avec cet objectif, nous

définissons les notations suivantes, pour un tas H composé de n_H objets :

- $D_{max}(H)$ est la distance maximale entre deux objets de H : $D_{max}(H) = \max_{O_i,O_j \in H} D(O_i, O_j)$
- $O_{center}(H)$ est le centre de gravité des objets de H : $O_{center}(H) = \frac{1}{n_H} \sum_{O_i \in H} O_i$ On peut remarquer qu'en général, O_{center} ne correspond pas à un objet particulier.
- $O_{dissim}(H)$ est l'objet le plus dissimilaire de H, c'est à dire maximisant $D(., O_{center}(H))$,
- $D_{mean}(H)$ est la distance moyenne entre les objets de H et du centre de gravité $O_{center}(H)$: $D_{mean}(H) = \frac{1}{n_H} \sum_{O_i \in H} D(O_i, O_{center}(H))$
- $D_{maxcenter}(H)$ est la distance entre l'objet le plus dissimilaire du tas H et le centre de gravité : $D_{maxcenter}(H) = D(O_{dissim}(H), O_{center}(H))$

L'algorithme AntClass

La colonie se compose de p fourmis $a_i, ..., a_p$. Chaque fourmi est positionnée sur une case de la grille. Initialement cette position est générée de manière aléatoire et uniforme. Il n'existe évidemment aucun contrôle centralisé de la colonie. Chaque fourmi a_i se déplace et peut alors déposer ou ramasser un objet.

Le déplacement n'est pas totalement aléatoire. Initialement, a_i choisit une direction aléatoire parmi les 8 possibles. a_i a désormais une probabilité $P_{direction}$ de continuer dans cette direction à son prochain déplacement, sinon elle génère aléatoirement une nouvelle direction. Chaque fourmi possède aussi un paramètre *vitesse* qui indique le nombre de pas qu'elle fera dans la direction sélectionnée. Le critère d'arrêt de cet algorithme est simplement le nombre d'itérations.

Ramasser un objet. Quand une fourmi ne transporte pas d'objet, elle cherche un objet à ramasser en considérant les 8 cases qui lui sont voisines. Dès qu'un objet ou qu'un tas d'objets a été trouvé, trois cas sont à considérer :

1. un objet seul : la fourmi a une probabilité fixée de prendre l'objet.
2. Un tas de deux objets : nous avons la propriété suivante : $D_{maxcenter}(H) = D_{mean}(H)$. Il n'y a donc pas de réelle heuristique basée sur la distance D à appliquer ici. C'est pourquoi nous avons simplement donné à la fourmi une probabilité $P_{destroy}$ de ramasser un des objets et ainsi détruire le tas.
3. Un tas de plus de deux objets : la fourmi ramasse l'objet le plus éloigné du centre du tas à condition que cette dissimilarité soit supérieure à un seuil fixé T_{remove}. C'est une heuristique simple mais qui permet d'obtenir des tas plus homogènes.

Déposer un objet. Quand la fourmi transporte un objet, elle explore les 8 cases autour de sa position courante. Trois cas sont encore à considérer :

1. la case est vide : la fourmi a simplement une probabilité constante P_{drop} de déposer son objet.
2. La case ne contient qu'un seul objet : la fourmi dépose son objet et crée ainsi un tas de deux objets avec comme condition que son objet soit suffisament similaire à celui déjà en place sur la case (paramètre T_{create}).
3. La case contient un tas : la fourmi va ajouter l'objet qu'elle transporte au tas sous la condition qu'il est plus proche du centre de H que l'objet le plus

dissimilaire de H.

Afin d'éviter qu'une fourmi ne transporte un objet pendant trop longtemps, dans le cas d'objets très dissimilaires aux autres par exemple, la fourmi dépose cet objet automatiquement, après Max_{carry} itérations, sur la première case vide rencontrée.

Mémoire locale des fourmis. Comme les fourmis réelles ont la possibilité de mémoriser plusieurs sites dans leur environnement (voir par exemple Fresneau (1985)), nous avons ajouté une mémoire à chaque fourmi dans le but d'accélérer la classification, d'une manière similaire à Lumer et Faieta (1994). Dès qu'une fourmi rencontre un tas H, elle stocke dans sa mémoire la position de H sur la grille ainsi que O_{center} et $D_{maxcenter}(H)$. Le comportement de dépose et le mouvement de la fourmi sont modifiés de la façon suivante : quand la fourmi transporte un objet, elle cherche dans sa mémoire un tas H sur lequel elle pourrait déposer l'objet. Si elle en trouve un, alors la fourmi se dirige vers H. Si la fourmi n'a pas déposé son objet en chemin, elle le déposera sur H si il est toujours valable, c'est-à-dire si il n'a pas été détruit ou si il n'a pas été trop modifié par d'autres fourmis. Les fourmis ne disposent que de quatre emplacements mémoire. Quand une fourmi rencontre un nouveau tas et que sa mémoire est pleine, le plus vieux tas est remplacé par le nouveau. Les fourmis peuvent ainsi oublier un tas. Comme résultat de cette mémorisation, la fourmi acroîtra la proportion des objets qui seront affectés à un tas.

Paramètres hétérogènes des fourmis. Comme nous l'avons mentionné précédement,

Paramètre	**Description**	**Valeur (ou intervalle)**
Speed	amplitude des mouvements	$[1, 10]$
$P_{direction}$	prob. de se déplacer dans la même direction	$[0.5, 1[$
Max_{carry}	durée max. de transport des objets	$[20, 200]$
P_{load}	prob. de ramasser un objet seul	$[0.4, 0.8]$
$P_{destroy}$	prob. de détruire un tas de 2 objets	$[0, 0.6]$
T_{remove}	dissimilarité min. nécéssaire pour enlever un objet d'un tas	$[0.1, 0.2]$
T_{create}	dissimilarité max. autorisée pour créer un tas de 2 objets	$[0.05, 0.2]$

TAB. 1 – *Les paramètres des fourmis dans AntClass. Pour chaque fourmi, ils sont générés aléatoirement à l'intérieur des bornes indiquées. The ant's parameters in AntClass. For each ant, they are randomly generated inside the given bounds.*

nous devons absolument éviter un paramétrage complexe afin de simplifier l'utilisation de AntClass par des experts du domaine. La manière de résoudre ce problème est une fois encore de nous inspirer des fourmis réelles et d'utiliser une population hétérogène de fourmis ayant des comportements différents. Initialement les paramètres des fourmis sont générés aléatoirement à l'intérieur des bornes données dans le tableau 1. Ces valeurs seront utilisées dans la suite de cet article pour toutes les données traitées. Ces bornes ont été obtenues après de nombreux tests sur les ensembles de données artificielles.

Hybridation

L'algorithme précédent basé sur les fourmis seules avait pour avantage majeur de donner une partition pertinente des données sans aucune information initiale quant à la partition finale. Cependant, deux problèmes importants demeurent. Le premier est dû au fait que certains objets ne sont affectés à aucun tas quand l'algorithme s'arrête, nous les appelerons «objets libres» dans cet article. Ils peuvent correspondre à des objets qui sont encore transportés par les fourmis ou à des objets qui sont seuls sur la grille. Le second problème est que si un objet a été affecté à un mauvais tas, cela peut prendre un certain temps pour que cet objet soit transporté sur le bon tas. Il faut en effet qu'une fourmi s'intéresse à ce tas.

La solution que nous proposons alors est de combiner deux algorithmes complémentaires, l'un basé sur les fourmis et l'autre sur les centres mobiles. L'algorithme des centres mobiles (ACM) est un algorithme itératif qui nécessite une partition initiale des données (Jain et Dubes, 1988). Ici, la partition initiale est donnée par les fourmis. L'ACM procède de la façon suivante : les centres de chaque classe sont calculés, puis tous les objets sont affectés à la classe dont le centre est le plus proche. Ce cycle est réitéré un certain nombre d'itérations de suite ou jusqu'à ce qu'aucune modification de la partition ne soit constatée durant un cycle.

Algorithme final

l'algorithme des centres mobiles n'est pas optimal et il est incapable de donner une estimation du nombre de classes. Nous avons expérimentalement remarqué que les deux précédentes étapes de AntClass (fourmis et centres mobiles) donnaient de bon résultats en ce qui concerne les erreurs de classification. Cependant, le nombre de classes était toujours surestimé. Nous avons observé que ces deux premières étapes de AntClass généraient beaucoup de petits tas très homogènes. L'idée consiste à considérer ces petits tas homogènes comme des objets eux-mêmes, on augmente en quelque sorte la capacité de transport des fourmis.

Dans le but de traiter les tas d'objets par les fourmis, nous avons simplement adapté les algorithmes décrits précédemment : les fourmis seront désormais capables de transporter un tas entier d'objets. L'algorithme de ramassage d'un tas est globalement identique à celui de ramassage d'un objet seul. Les fourmis ramassent un tas avec la même probabilité P_{load}. Cependant nous avons dû ajouter un mécanisme pour éviter que les fourmis ne transportent tous les tas en même temps car ils deviendront de moins en moins nombreux. S'il reste peu de tas sur la grille, il est important qu'ils ne soient pas tous transportés en même temps car les fourmis seraient incapables de les agréger. Quand un tas vient d'être déposé, il est marqué avec une sorte de phéromone qui évite à d'autres fourmis de ramasser ce tas pendant un certain temps (500 itérations). Les fourmis déposent un tas H_1 sur un autre tas H_2 si :

$$\frac{D(O_{center}(H_1), O_{center}(H_2))}{D_{max}} \leq T_{create}$$

Quand H_1 et H_2 sont rassemblés, ils forment un unique tas H_3. Cela signifie que deux tas qui ont été rassemblés ne peuvent plus être séparés. Ceci accélère la convergence.

Enfin, nous avons remarqué que l'étape précédente (les fourmis qui classent les tas d'objets) fournit une bonne approximation du nombre de classes mais peut introduire un certain nombre d'erreurs de classification. Afin de corriger ces erreurs, nous

utilisons une fois de plus l'algorithme des centres mobiles sur les objets. AntClass se déroule donc en quatre étapes :

1. classification des objets par l'algorithme basé sur les fourmis,
2. utilisation de l'algorithme des centres mobiles sur la partition obtenue par les fourmis.
3. classification des tas par les fourmis, et
4. utilisation les centres mobiles une dernière fois.

De plus, nous pouvons ajouter que toutes les valeurs dans les données sont normalisées afin déviter tout problème d'échelle entre les attributs.

RÉSULTATS

Bases de données et paramétrages

Nous avons appliqué AntClass à plusieurs bases de données, à des données artificielles semblables à celles utilisées par Lumer et Faieta (1994), à des bases de données réelles issues du «Machine Learning repository», et à des problèmes réels en coopération avec l'industrie. Nous ne décrirons pas cette dernière application pour des raisons de confidentialité, mais nous pouvons mentionner que les résultats obtenus par AntClass sont pertinents par rapport à ceux obtenus par SAS (http://www.sas.com), ce qui est déjà très positif.

Base de données	n	k	c
Artif. 1	80	2	4
Artif. 2	270	2	9
Artif. 3	200	2	4
Artif. 4	150	10	3
Iris	178	4	3
Wine	178	13	3
Glass	214	9	2-6
Soybean	47	21	4
Thyroid	215	5	3
Breast cancers	699	9	2

TAB. 2 – *Les 12 bases de données utilisées dans cet article. Pour chaque base, nous indiquons le nombre d'objets (n), le nombre d'attributs numériques (k) et le nombre de classes (c). The 12 databases used in this article. For each database we indicate the number of objects (n), the number of attributes (k) and the number of clusters (c).*

Les informations générales sur les bases de données sont représentées dans le tableau 2. Comme nous le verrons dans la suite, «Artif. 1»...«Artif. 4» ont été utilisées pour évaluer AntClass sur des bases de données dont les propriétées sont connues. Dans ces bases artificielles, les exemples ont été générés en suivant des lois gaussiennes.

Dans le but d'évaluer les partitions obtenues par AntClass, nous avons en fait utilisé des bases de données pour l'apprentissage supervisé dont nous connaissons le nombre de classes (c). Quand AntClass est utilisé pour partitionner les données cet

attribut c ne lui est pas fourni. C'est seulement a posteriori que cette information de classe lui est fournie afin d'évaluer les résultats.

Nous avons défini deux mesures de performance pour évaluer la différence entre la classification obtenue et celle recherchée. La première est un taux d'erreur de classification (E_c). Elle est calculée de la manière suivante : pour une classe donnée H obtenue par AntClass, nous considérons la classe réelle majoritaire de H selon l'attribut c. Tous les objets de H qui n'appartiennent pas à cette classe sont considérés comme mal classés. Le taux d'erreur est alors le rapport entre le nombre total d'objets mal classés pour tous les tas et le nombre d'objets de E. La seconde mesure de performance est le nombre de classes trouvées (c').

Tous les tests ont été effectués sur un PC standard (Pentium 166). Un test est effectué en quelques secondes (de 10 à 20 secondes). Tous les résultats présentés sont une moyenne sur 50 essais. Les paramètres de AntClass sont identiques pour toutes les bases de données (voir tableau 1). Les fourmis sont simulées pendant 2000 itérations pour classer les objets, et 50000 itérations pour classer les tas. Le nombre d'itérations de l'algorithme des centres mobiles a été limité à 10. Du point de vue de l'implémentation informatique, on peut noter que le travail des fourmis peut aisément être répartis sur plusieurs ordinateurs pour accélérer cette étape.

Résultats sur les bases de données artificielles

Base de données	**1: Fourmis** sur les objets	**2: ACM** sur les objets	**3: Fourmis** sur les tas	**4: ACM** sur les tas
Artif. 1: E_c	11.58 %	0.21 %	0.42 %	0.00 %
c'	8.15	7.76	4.24	4
Artif. 2: E_c	17.24 %	0.52 %	2.22 %	0.00 %
c'	22.30	17.07	10.46	9.02
Artif. 3: E_c	20.35 %	6.32 %	6.93 %	4.66 %
c'	15.06	14.98	5.42	4.42
Artif. 4: E_c	22.23 %	3.32 %	2.68 %	1.33 %
c'	5.22	5.18	2.94	2.96

TAB. 3 – *Résultats itermédiaires et finaux obtenus pour chacune des quatres étapes de AntClass pour les quatres bases artificielles. «ACM» signifie Algorithme des centres mobiles, E_c correspond au taux d'erreur de classification et c' correspond au nombre de classes trouvé. Intermediary and final results obtained for each of the four step of AntClass for the artificial databases. ACM is the Kmeans algorithm, E_c is the classification error and c' is the number of found clusters.*

Les résultats sont présentés dans le tableau 3. Nous avons indiqué les résultats intermédiaires obtenus à chacuns des 4 étapes de AntClass afin d'illustrer la progression qui a lieu au cours de la classification.

Le premier algorithme utilisant seulement les fourmis sur les objets trouve une classification initiale des données mais qui contient trop de classes ainsi que des erreurs de classification. La convergence de cette approche purement stochastique serait longue à obtenir. Il faut également gérer le cas des objets libres qui n'ont été affectés à aucun tas.

A la fin de la deuxième étape d'AntClass, les erreurs de classification ont été

réduites mais le nombre de classes est encore beaucoup trop élevé. Cela est dû au fait que l'ACM est très sensible à la partition initiale qui lui est fournie. Si celle-ci contient trop de classes, la partition finale sera loin de l'optimum. Lors de cette étape, tous les objets sont affectés à un tas.

Une fois la troisième étape effectuée, les fourmis font converger la classification vers le bon nombre de classes en agrégeant directement des tas d'objets. On peut noter également qu'il reste des erreurs de classification. Finalement, lors de la dernière étape, l'utilisation de l'ACM diminue ces erreurs de classification. Mais cette fois, étant donné que le nombre de classes a été bien mieux évalué et qu'une partition quasi-optimale a été calculée, l'ACM trouve une partition souvent optimale.

Résultats sur des bases de données réelles

Base de donnée	c	$\bar{c'}$	E_c
Artif. 1	4	4	0.00 %
Artif. 2	9	9.02	0.00 %
Artif. 3	4	4.42	4.66 %
Artif. 4	3	2.96	1.33 %
Iris	3	3.02	15.4 %
Wine	3	3.06	5.38 %
Glass	2-6	7.7	4.48 %
Soybean	4	4.82	0.13 %
Thyroid	3	3.28	6.38 %
Breast cancers	2	4.6	3.50 %

TAB. 4 – *Résultats obtenus par AntClass sur des bases artificielles et réelles. Results obtained by AntClass with artificial and real databases.*

Ces résultats sont présentés dans le tableau 4. Les bons résultats obtenus sur les bases artificielles sont confirmés sur les bases réelles, sauf peut être pour la base des Iris de Fisher (Fisher, 1936) où le nombre de classes est correct mais où il reste des erreurs de classification. Une des trois classes (Sétosa) est facilement séparable des autres, mais il est bien connu que les deux autres sont impossibles à séparer correctement même en connaissant la classification à l'avance (cas de l'apprentissage supervisé).

Pour les autres bases, les taux d'erreur de classification a posteriori sont très bons. De plus, AntClass approche très bien le nombre de classes réel. Pour la base Glass, on peut définir de 2 à 6 classes. On constate bien que AntClass découvre les 6 classes «naturelles», ce qui souligne bien un trait caractéristique des données.

DISCUSSION

AntClass peut être considéré comme une extention des travaux de Lumer et Faieta (1994). Parmi les améliorations que nous présentons dans cet article, on peut noter par exemple les points suivants : l'introduction d'heuristiques plus robustes inspirées des fourmis, la résolution du problème des objets non affectés, l'accélération

de la convergence grâce à l'algorithme des centres mobiles, l'utilisation d'une classification hiérarchique sur les tas d'objets, l'expérimentation de l'algorithme obtenu sur de nombreux problèmes réels et une comparaison avec les résultats obtenus par les centres mobiles seuls.

Les travaux futurs consistent à tester comment se comporte AntClass sur des bases de données de taille plus importante, comme dans une application réelle en cours d'étude. Nous allons également considérer d'autres formes d'interaction entre les fourmis, comme l'échange d'objets lorsque deux fourmis se croisent.

RÉFÉRENCES

BILCHEV, G. et PARMEE, I. (1996). « Constrained Optimisation with an Ant Colony Search Model ». Dans *Second Conference on Adaptive Computing in Engineering Design and Control*, Plymouth, UK.

DENEUBOURG, J.-L., GOSS, S., FRANKS, N., SENDOVA-FRANKS, A., DETRAIN, C., et CHRETIEN, L. (1991). « The dynamic of collective sorting robot-like ants and ant-like robots ». Dans MEYER, J.-A. et WILSON, S., éditeurs, *First International Conference on Simulation of Adaptive Behavior*. MIT-Press.

DI CARO, G. et DORIGO, M. (1997). « AntNet: A Mobile Agents Approach to Adaptive Routing ». Rapport Technique 97-12, IRIDIA, Université Libre de Bruxelles, Belgium.

DORIGO, M. et GAMBARDELLA, L. (1997). « Ant Colony Sytem: A Cooperative Learning Approach to the Travelling Salesman Problem ». *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

FISHER, R. (1936). « The Use of Multiple Measurements in Axonomic Problems ». *Annals of Eugenics*, 7:179–188.

FRESNEAU, D. (1985). « Individual foraging and path fidelity in a ponerine ant ». *Insectes Sociaux, Paris*, 32(2):109–116.

GAMBARDELLA, L., TAILLARD, E., et DORIGO, M. (1997). « Ant Colonies for the QAP ». Rapport Technique 97-4, IDSIA, Lugano, Switzerland.

GOSS, S. et DENEUBOURG, J.-L. (1991). « Harvesting by a group of robots ». Dans VARELA, F. et BOURGINE, P., éditeurs, *First European Conference on Artificial Life, Paris, France*, pages 195–204. MIT Press.

JAIN, A. et DUBES, R. (1988). *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series.

KUNTZ, P., LAYZELL, P., et SNYERS, D. (1997). « A Colony of Ant-like Agents for Partitioning in VLSI Technology ». Dans HUSBANDS, P. et HARVEY, I., éditeurs, *Fourth European Conference on Artificial Life*, pages 417–424. MIT Press, Boston.

LUMER, E. et FAIETA, B. (1994). « Diversity and Adaptation in Populations of Clustering Ants ». Dans *Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3*, pages 501–508.