Construction incrémentale et visualisation de graphes de voisinage par des fourmis artificielles

Julien Lavergne*, Hanene Azzag**
Christiane Guinot*,***, Gilles Venturini*

*Laboratoire d'Informatique,
Ecole Polytechnique de l'Université de Tours,
64 avenue Jean Portalis, 37200 Tours, France
{julien.lavergne,gilles.venturini}@univ-tours.fr,
http://www.antsearch.univ-tours.fr/webrtic
**Laboratoire d'Informatique de l'Université Paris-Nord
99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
hanene.azzag@lipn.univ-paris13.fr,
http://www-lipn.univ-paris13.fr/A3/
***CE.R.I.E.S, 20 rue Victor Noir, 92521 Neuilly-Sur-Seine, France
christiane.guinot@ceries-lab.com,
http://www.ceries.com

Résumé. Cet article décrit un nouvel algorithme incrémental nommé AntGraph pour la construction de graphes de voisinage. Il s'inspire du comportement d'auto-assemblage observé chez des fourmis réelles où ces dernières se fixent progressivement à un support fixe puis successivement aux fourmis déjà fixées afin de créer une structure vivante. Nous utilisons ainsi une approche à base de fourmis artificielles où chaque fourmi représente une donnée. Nous indiquons comment ce comportement peut être utilisé pour construire de manière incrémentale un graphe à partir d'une mesure de similarité entre les données. Nous montrons finalement que notre algorithme obtient de meilleurs résultats en comparaison avec le graphe de Voisins Relatifs, notamment en terme de temps de calcul.

1 Introduction

Dans cet article, nous nous intéressons au problème suivant : étant donné un ensemble de n données $d_1, ..., d_n$ et une matrice de similarité $M(d_i, d_j)$ entre ces données, comment permettre à un expert d'explorer cet ensemble de données de manière visuelle et avec une approche guidée par le contenu. Nous considérons que l'expert souhaite avoir une vue globale des données mais également exploiter localement les données Shneiderman (1996), et en particulier passer de l'une à l'autre par une relation de voisinage tenant compte de la similarité. Notre problème se décompose en deux parties : établir un graphe de voisinage entre les données à partir de la similarité, et visualiser ce graphe afin de permettre à l'utilisateur de l'explorer.

Nous allons donc nous concentrer sur les méthodes de construction de graphes de voisinage (voir un état de l'art dans Hacid et Zighed (2005)). Ce type de structure est également appelée

graphe de proximité. L'utilisation de ces graphes se retrouve aussi bien en fouille de données (classiques, spatiales) que dans l'apprentissage ou la classification de données (Ester et al., 1997). Cependant, les algorithmes de construction de ces graphes sont d'une grande complexité (par exemple $O(n^3)$ pour l'algorithme des Voisins Relatifs), ce qui les rend inefficaces face à de grands volumes de données.

Nous nous sommes intéressés dans cet article à une méthode biomimétique pour la construction incrémentale de ce type de graphe. Il s'agit d'une généralisation de l'algorithme AntTree proposé par Azzag (2005) dans sa thèse. L'auteur a introduit un algorithme de classification non supervisée hiérarchique capable de traiter n'importe quel type de données. Il se base sur le principe d'auto-assemblage observé chez une population de fourmis réelles présenté dans Lioni et al. (2001). Nous proposons ici une généralisation d'AntTree en utilisant un graphe comme modèle de structure. Nous allons donc chercher à construire un graphe de voisinage qui soit représentatif de la similarité existante entre ces données avec une complexité inférieure à celle observée dans les méthodes classiques (Voisins Relatifs, graphes de Gabriel, triangulation de Delaunay).

La suite de notre article est organisée comme suit : dans la section 2, nous présentons les principes des graphes de voisinage et quelques modèles de référence dont celui des Voisins Relatifs auquel nous comparons notre algorithme. Dans la section 3, nous détaillons l'algorithme de construction ainsi que l'ensemble des règles locales de comportement des fourmis artificielles. Nous précisons également, dans cette section, la méthode de visualisation utilisée pour réaliser l'affichage de graphes de voisinage. La section 4, quant à elle, est consacrée aux résultats et à l'étude comparative sur des bases de données numériques. La dernière section rassemble les conclusions faites au cours de l'article et présente des perspectives.

2 Graphes de voisinage et visualisation

2.1 Principes des graphes de voisinage

Considérons un graphe $\mathcal{G}(\Omega, \mathcal{V})$ où Ω est l'ensemble des noeuds du graphe et $\mathcal{V}$ l'ensemble des arêtes contenu dans le graphe. $\mathcal{G}$ est appelé graphe de voisinage si la propriété ci-contre est respectée : il existe une relation binaire entre deux points $(a, b) \in \Omega^2$ si et seulement le couple de points $(a, b) \in \mathcal{V}$. En d'autres termes, Pour un point p donné de Ω, son voisinage $\Upsilon(p)$ est l'ensemble des points (sous-graphe) contenant p et tous les autres sommets directement connectés. Dans notre cas, chaque noeud (ou sommet) du graphe est une donnée. Les liens qui vont connecter les noeuds entre eux doivent représenter une information sur le voisinage des données. Cette information peut être une notion de distance entre les données.

Il existe plusieurs méthodes pour établir ce type de graphe. La triangulation de Delaunay Preparata et Shamos (1985) va relier entre elles les données qui vérifient la propriété suivante : le cercle passant par les trois sommets de chaque triangle ne contient aucune autre donnée. Le graphe de Gabriel se construit selon Gabriel et Sokal (1969) : deux données d_i et d_j sont voisines si l'hypersphère de rayon $|d_i - d_j|$ et passant par d_i et d_j est vide. Enfin, le graphe des Voisins Relatifs s'obtient si la propriété énoncée dans Toussaint (1980) est respectée : deux données d_i et d_j sont voisines si l'intersection des sphères centrées respectivement en d_i et d_j et de rayon $|d_i - d_j|$ est vide.

D'une manière générale, ces méthodes ne sont pas incrémentales, et nécessitent que les données soient numériques. Elles sont coûteuses en temps de calcul. En effet, pour chaque point d'un espace donné, la construction d'un graphe de voisinage nécessite de déterminer la proximité du point avec tous les autres points. De plus, la complexité de ces algorithmes s'avère être polynomiales en n. Attali (1995) nous renseigne sur la triangulation de Delaunay et le graphe de Gabriel qui ont une complexité en $O(n\log(n)+n^{\frac{N}{2}})$ où N représente le nombre d'attributs des données. A l'heure actuelle, les Voisins Relatifs s'expriment en $O(n^3)$, cité dans Hacid et Zighed (2005), et sont très largement utilisés pour résoudre de nombreux problèmes comme nous l'indique Toussaint (1991). Récemment, une extension a été proposée afin de construire de manière incrémentale un graphe de voisinage dans Hacid et Zighed (2005). Cette approche a pour but de compléter un graphe existant lors d'une phase de mise à jour.

2.2 Visualisation de graphes par forces et ressorts

Il existe de nombreux algorithmes de visualisation de graphes Di Battista et al. (1998). Les travaux de recherche sur les algorithmes de ressorts commencent avec Tutte (1963) et se poursuivent avec Eades (1984). Ce dernier utilise l'analogie suivante pour expliquer la visualisation dynamique de graphes : il compare les arêtes dans un graphe à des ressorts. Le système, ainsi considéré, engendre des forces entre les sommets. Ce qui provoque naturellement des déplacements de sommets. Les sommets s'attirent et se repoussent. La notion d'attraction entre sommets se réalise grâce aux arêtes qui cherchent à atteindre une distance cible associée. Eades (1984) ajoute la notion de forces de répulsion aux sommets. La condition d'arrêt initialement proposée pour un tel système est un nombre maximum d'itérations (évolution du graphe dans le temps).

Plusieurs recherches ont ensuite été consacrées au domaine. Nous pouvons citer entre autres Kamada et Kawai (1989), Frick et al. (1994) et Fruchterman et Reingold (1991). Ces différentes propositions ont amené à l'établissement de plusieurs modèles de visualisation dynamique de graphes. Nous nous baserons sur Fruchterman et Reingold (1991) qui offre une méthode générique de visualisation.

Etant donné un graphe de voisinage, on définit dans notre outil la longueur à atteindre entre chaque couple de noeuds voisins par la notion de similarité existante entre les deux données correspondantes. Ensuite, les noeuds sont placés initialement de manière aléatoire sur un plan 2D, et les forces et ressorts agissent jusqu'à stabilisation du graphe qui offre alors une visualisation homogène et agréable à l'utilisateur.

3 AntGraph

Notre modèle est une extension de l'algorithme AntTree présenté dans Azzag et al. (2003). AntTree effectue une classification non supervisée hiérarchique pour regrouper des données de n'importe quel type (numérique, symbolique, textuel, ...) sous forme d'un arbre. Nous généralisons ainsi ces principes dans le but de construire un graphe de fourmis avec comme connaissance de départ la mesure de similarité entre les données. Nous détaillons ci-dessous les règles de construction du graphe par des fourmis artificielles.

Nous considérons ici un ensemble de données triées de manière aléatoire. Chaque fourmi va représenter une donnée à regrouper. Nous choisissons ensuite aléatoirement une fourmi f_0

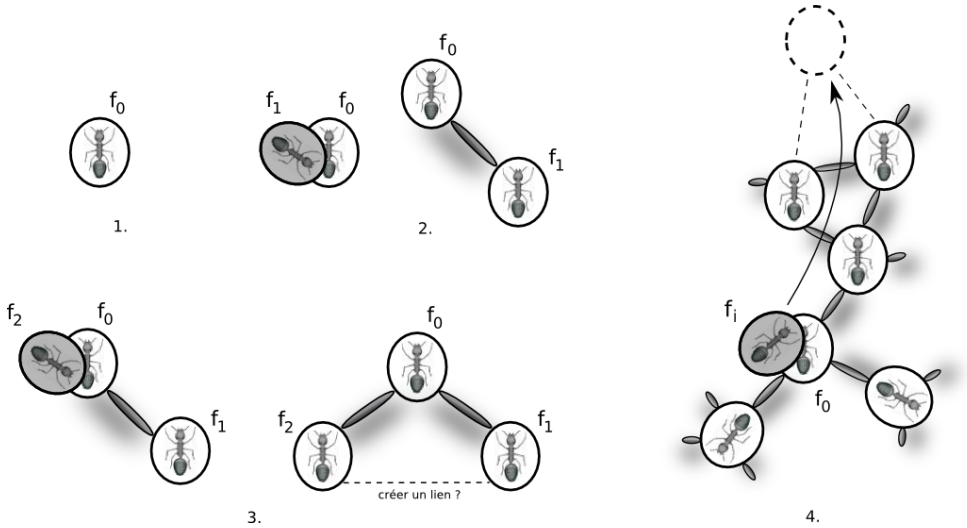


FIG. 1 – *Etapes de construction du graphe. En 1., f_0 est le premier sommet et constitue le support fixe. En 2., la fourmi suivante f_1 se déplace sur f_0 et se fixe à cette seule fourmi qui lui est la plus similaire. En 3., la fourmi f_2 se déplace sur f_0, se compare à elle et se fixe. f_1 est une fourmi fille de f_0. f_2 doit vérifier si f_1 lui est similaire ou pas. En 4., nous avons le principe généralisé de construction de AntGraph.*

que nous considérons comme le point d'entrée dans l'étape de construction du graphe, et donc comme le premier noeud de ce graphe. Puis nous simulons les actions de chaque fourmi f_i, qui entre dans le graphe par le noeud f_0, se déplace de noeud en noeud, et ce jusqu'à ce qu'elle se connecte dans le graphe. On peut alors passer à la fourmi suivante. Lorsque f_i est en déplacement, on note f_{pos} la fourmi sur laquelle elle se trouve. Ensuite, le voisinage perçu par f_i correspond à f_{pos} ainsi qu'aux fourmis connectées à f_{pos}. Intuitivement, lorsque f_i arrive dans le graphe, elle va suivre le chemin de similarité maximum indiqué dans le voisinage qu'elle perçoit, puis elle va se connecter en établissant un ou plusieurs liens (voir 1). L'algorithme est le suivant :

Le choix de la fourmi la plus similaire pour f_i implique plusieurs cas de décision par rapport à sa position :

- f_{pos} est la fourmi la plus similaire à f_i et ne possède pas de fourmis voisines. f_i se connecte alors directement à f_{pos}.
- f_{pos} n'est pas la fourmi la plus similaire à f_i et possède des fourmis voisines. Dans ce cas, f_i se déplace sur la fourmi voisine la plus similaire.
- f_{pos} est la fourmi la plus similaire et possède des fourmis voisines. Ce dernier cas se présente lorsque f_{pos}, la fourmi la plus similaire, possède des fourmis voisines qui pourraient être également similaires à f_i. Dans ce cas de figure, f_i se connecte sur f_{pos} et sur toutes les fourmis voisines de f_{pos} suffisamment similaires à elle par rapport à un seuil de tolérance S_t. L'opération est répétée récursivement pour toutes les fourmis connectées aux voisines de f_{pos}.

Algorithme 1 Vue d'ensemble de l'algorithme de construction incrémentale

ENTRÉES: f_0 est le 1^{er} noeud du graphe.
SORTIES: le graphe de voisinage G.
pour une fourmi f_i non connectée **faire**
  f_i se place en f_0 (i.e. $f_{pos} \leftarrow f_0$)
  si f_{pos} n'est pas la fourmi la plus similaire à f_i **alors**
    f_i se déplace sur la fourmi voisine de f_{pos} qui lui est la plus similaire
  sinon si f_{pos} est la fourmi la plus similaire à f_i **alors**
    f_i se connecte à f_{pos}
    si f_{pos} possède des fourmis voisines **alors**
      calcul du seuil de tolérance S_t
      f_i interroge les fourmis voisines de f_{pos}
      f_i se connecte aux voisines de f_{pos} les plus similaires ($\geq S_t$) et récursivement aux voisines de ces fourmis avec la même condition (similarité $\geq S_t$)
    finsi
  finsi
fin pour

Nous précisons que la valeur du seuil de tolérance S_t est calculé de la manière suivante : $S_t = \alpha * sim(f_i, f_{pos})$ avec $\{\alpha \in \mathbb{R}, 0 \leq \alpha \leq 1\}$.

Cet algorithme est bien incrémental : les fourmis/données sont ajoutées une à une dans le graphe.

4 Etude comparative

Nous réalisons notre étude sur des bases numériques artificielles et réelles (voir la partie gauche de la table 1). Notre méthode peut traiter tout type de données, du moment que la similarité existe. Les bases de données artificielles {Art1, ..., Art6} nous sont fournies par Azzag (2005). Les bases réelles que nous utilisons proviennent du CE.R.I.E.S., Guinot et al. (2001), pour la base de même nom et du UCI Repository of Machine Learning, Blake et Merz (1998), pour les autres. Il est à noter que pour chaque test, la fourmi support f_0 est choisie aléatoirement. Il en est de même pour les fourmis ajoutées une à une dans le graphe.

4.1 Visualisations

Nous nous intéressons dans cette section à l'étude comparative des visualisations issues de Voisins Relatifs et de AntGraph. Comme cité auparavant, nous utilisons un algorithme à base de ressorts pour produire cette visualisation. Par soucis de respect des contraintes du nombre de pages, nous ne pouvons afficher de manière exhaustive la visualisation de toutes les bases de données précédemment citées dans la table 1. A cet effet, nous présentons, dans un premier temps, une comparaison des visualisations de quelques bases de données entre Voisins Relatifs et AntGraph (figure 2). Dans un second temps, nous nous intéressons à l'influence du seuil de tolérance S_t sur la construction d'un graphe avec AntGraph (figure 3).

Données	# classes réelles	M	N	T_{Exec} Voisins relatifs	T_{Exec} AntGraph
Art1	4	2	400	1,00 [0,04]	0,0071 [0,000]
Art2	2	2	1000	15,35 [0,18]	0,0303 [0,003]
Art3	4	2	1100	23,04 [0,53]	0,0413 [0,008]
Art4	2	2	200	0,16 [0,03]	0,0027 [0,000]
Art5	9	2	900	13,01 [0,14]	0,0168 [0,004]
Art6	4	8	400	1,26 [0,05]	0,0082 [0,000]
Iris	3	4	150	0,09 [0,03]	0,0021 [0,000]
Pima	2	8	768	4,87 [0,04]	0,0257 [0,004]
Soybean	4	35	47	0,00 [0,01]	0,0004 [0,000]
Thyroïd	3	5	215	0,14 [0,02]	0,0042 [0,000]
Vehicle	4	18	846	6,32 [0,10]	0,0295 [0,004]
Wine	3	13	178	0,11 [0,02]	0,0026 [0,000]
CE.R.I.E.S.	6	4	259	0,24 [0,02]	0,0044 [0,000]
Glass	6	9	214	0,16 [0,03]	0,0041 [0,000]

TAB. 1 – *Jeux de données utilisés pour notre comparaison entre Voisins Relatifs et AntGraph. M représente le nombre d'attributs et N le nombre de données. Les temps d'exécution sont en secondes. Les méthodes ont été programmées en JAVA et les tests réalisés sur un Pentium4 2Ghz et 768Mo de mémoire vive.*

Nous pouvons remarquer sur la figure 2, que les visualisations obtenues pour AntGraph ont une ressemblance très forte avec les visualisations obtenues pour Voisins Relatifs. Nous distinguons bien les regroupements de données. Nous retrouvons visuellement les classes réelles : 3 classes pour Iris, 4 pour Art6 et 2 pour Art2.

La figure 3 nous présente le rôle du seuil de tolérance S_t dans la qualité de construction du graphe. Les visualisations correspondantes nous permettent de constater que la valeur de la constante α a une influence sur la formation des graphes. Nous avons fait évoluer la constante de 0,1 à 0,99. Nous présentons trois visualisations pour les valeurs 0,8, 0,9 et 0,97. En dessous de 0,9, nous obtenons des graphes où les noeuds sont pratiquement tous connectés les uns aux autres : les fourmis ne se connectent pas aux fourmis les plus similaires mais à beaucoup d'autres qui ne le sont pas du tout. Pour une valeur α de 0,9 ou légèrement supérieure, nous visualisons de manière esthétique le regroupement des données en classes : les fourmis se connectent à leurs voisines les plus similaires. Enfin, plus la valeur de α est proche de 1, plus le graphe perd de l'information : les fourmis ne se connectent pratiquement plus qu'à une seule voire deux fourmis similaires (le graphe se transforme en arbre). Les visualisations les plus probantes que nous obtenons s'obtiennent pour des valeurs α supérieures à 0,9.

4.2 Temps d'exécution

Nous avons représenté dans la partie droite de la table 1 les temps d'exécution obtenus par les deux méthodes et uniquement pour le calcul du graphe. En comparaison avec l'algorithme des Voisins Relatifs, notre algorithme obtient les meilleurs temps sur la totalité des bases testées. Par exemple pour les bases Art2 et Art5, AntGraph est respectivement 500 à 800 fois plus

Art1

	0	1	$\sum$
0	77816	1492	79308
1	235	257	492
$\sum$	78051	1749	

Iris

	0	1	$\sum$
0	10580	392	10792
1	100	103	203
$\sum$	10680	495	

Art2

	0	1	$\sum$
0	492909	53318	498227
1	676	597	1273
$\sum$	493585	5915	

Pima

	0	1	$\sum$
0	286279	6846	293125
1	955	448	1403
$\sum$	287234	7294	

Art3

	0	1	$\sum$
0	597013	6040	603053
1	711	686	1397
$\sum$	597724	6726	

Soybean

	0	1	$\sum$
0	958	47	1005
1	41	35	76
$\sum$	999	82	

Art4

	0	1	$\sum$
0	19339	319	19658
1	115	127	242
$\sum$	19454	446	

Thyroïd

	0	1	$\sum$
0	21384	1318	22072
1	169	134	303
$\sum$	21553	1452	

TAB. 2 – *Extrait de matrices de confusion des arêtes entre Voisins Relatifs et AntGraph. "0" signifie absence de lien et "1" présence d'un lien. Une table indique donc le nombre liens en accord ((0,0) et (1,1)) ou en désaccord ((0,1) et (1,0)) entre les graphes construits par chacune des méthodes.*

rapide. Nous pouvons remarquer que plus le nombre de données est important, plus notre algorithme est rapide par rapport à l'algorithme des Voisins Relatifs. La complexité de ce dernier étant en $O(n^3)$, le temps d'exécution devient prohibitif pour les grands ensembles de données. A ce niveau, notre algorithme conserve des temps d'exécution très compétitifs, ce qui présage de nouvelles applications dans le cadre des grandes bases de données, et représente un avantage certain.

Cette baisse de la complexité est du au fait que les nouvelles fourmis qui arrivent ne se comparent pas à l'ensemble des données déjà présentes dans le graphe mais suivent les chemins de plus grande similarité et ne rencontrent ainsi qu'un nombre très limité de données. A titre indicatif, dans un arbre équilibré, en suivant une branche de la racine vers les feuilles on ne rencontre que $O(ln(n))$ noeuds.

Néanmoins, ces performances en terme de temps de calcul ne renseignent pas directement sur la qualité du graphe construit. Nous avons donc poursuivi nos expérimentations pour montrer que le graphe construit par AntGraph est représentatif de la similarité entre les données, au même titre que Voisins Relatifs.

Données	Voisins Relatifs	AntGraph	k=1	k=2	k=3	D
Art1	0,021	0,058	0,020	0,058	0,134	0,258
Art2	0,011	0,050	0,014	0,046	0,084	0,240
Art3	0,012	0,049	0,012	0,048	0,080	0,242
Art4	0,037	0,072	0,032	0,061	0,135	0,393
Art5	0,016	0,054	0,016	0,058	0,110	0,294
Art6	0,086	0,133	0,084	0,131	0,148	0,275
Pima	0,068	0,114	0,072	0,113	0,138	0,213
Glass	0,051	0,078	0,051	0,099	0,141	0,255
Soybean	0,250	0,260	0,221	0.341	0,337	0,443
Iris	0,048	0,086	0,044	0,108	0,152	0,322
Vehicle	0,062	0,107	0,064	0,105	0,131	0,255
Thyroïd	0,037	0,070	0,040	0,084	0,145	0,174
CE.R.I.E.S.	0,065	0,109	0,730	0,137	0,191	0,270
Wine	0,115	0,158	0,116	0,174	0,190	0,276

TAB. 3 – *Moyenne des distances des arcs valués à 1 pour Voisins Relatifs, AntGraph, des graphes construits sur les* $k = 1,\ 2\ ou\ 3$ *plus proches voisins, et sur un graphe reliant toutes les données.*

4.3 Qualité des graphes construits

Nous rappelons qu'un graphe peut se représenter sous la forme d'une matrice binaire où 1 correspond à l'existence d'un lien entre deux sommets. Nous pouvons donc mesurer, avec une matrice de confusion, l'accord en terme de liens entre deux graphes (voir table 2).

Nous pouvons remarquer dans un premier temps que pour l'ensemble de nos matrices de confusion, le rapport du nombre total de non-liens (un 0 entre deux sommets) pour Voisins Relatifs d'une part et pour AntGraph d'autre part est relativement proche de 1. C'est le cas par exemple pour la base Art4 où le graphe de Voisins Relatifs totalise 19658 liens à 0 pour un équivalent dans le graphe de AntGraph à 19454. Les deux méthodes sont donc en accord sur ce point. On remarque de même qu'il y a un accord entre les liens qui sont créés dans les deux graphes : la moitié des liens de Voisins Relatifs se retrouvent dans AntGraph. Enfin, nous constatons que notre algorithme produit entre 2 à 4 fois plus de liens que Voisins Relatifs. C'est le cas par exemple pour la base Iris (Voisins Relatifs contient 203 liens pour 495 liens dans AntGraph) avec un rapport de 2 et la base Art3 (Voisins Relatifs contient 1273 liens pour 5915 liens dans AntGraph) avec un rapport de 4.

On constate donc qu'AntGraph construit plus de liens que Voisins Relatifs. Dans le cas de Voisins Relatifs, il est difficile de rencontrer la situation où un noeud est voisin d'un grand nombre d'autres noeuds. Concernant AntGraph, nous devons alors nous interroger sur la pertinence de ces liens par rapport à Voisins Relatifs. Pour cela, nous avons mesuré, dans un premier temps, pour un graphe donné, la similarité moyenne entre les voisins. Dans un second temps, nous avons effectué une mesure entre les données en considérant les k plus proches voisins ($k = 1,\ 2,\ ou\ 3$). Et dans un troisième et dernier temps, la mesure a été faite sur l'ensemble des données.

La table 3 donne les résultats obtenus. On peut constater alors qu'AntGraph crée des liens supplémentaires mais pas de manière aberrante par rapport à la similarité. En effet, la similarité moyenne des liens de AntGraph est du même ordre de grandeur que celle des 2 plus proches voisins, et reste de plus très inférieure à la similarité moyenne du graphe complet.

5 Conclusion

Nous avons proposé dans cet article une méthode pour la construction de graphes de voisinage. Elle s'inspire d'un modèle de construction de structures chez les fourmis artificielles. Le graphe est construit de manière progressive et incrémentale à partir d'un noeud initial. Chaque fourmi se déplace en suivant le chemin de plus grande similarité afin de trouver un noeud sur laquelle elle se connecte. Nous avons testé notre approche sur un ensemble de bases et nous avons montré que les temps d'exécution sont compétitifs par rapport à l'algorithme des Voisins Relatifs. Par ailleurs, nos tests sur la qualité des graphes ont montré également que notre approche propose des graphes représentatifs de la similarité entre les données.

Dans les perspectives en cours d'étude, nous pouvons principalement indiquer les pistes suivantes. D'une part le traitement de grands volumes de données paraît possible grâce à des temps très courts. Ainsi nous souhaitons faire des tests sur des flux de données afin de visualiser leur évolution au cours du temps. Ensuite, un défaut de notre approche (et des Voisins Relatifs) vient du fait que la construction du graphe est découplée de l'algorithme d'affichage. Autrement dit, pour un graphe très important, il est facile de le calculer grâce à la complexité faible d'AntGraph, mais il devient difficile de l'afficher car le temps de convergence des algorithmes à base de forces et de ressorts devient important. Donc nous allons combiner la construction incrémentale du graphe avec une version incrémentale de l'algorithme de visualisation. Enfin, nous avons commencé à tester des opérations interactives permettant de parcourir le graphe, de le modifier en décrochant des noeuds qui vont ensuite être réinjectés dans le graphe. De cette manière nous allons rendre plus interactif cet outil d'exploration de données.

Remerciements Les auteurs tiennent à remercier Hakim Hacid et Djamel A. Zighed pour leur aide dans l'implémentation des Voisins Relatifs.

Références

Attali, D. (1995). *Squelettes et graphes de Voronoï 2D et 3D*. Ph. D. thesis, Laboratoire des Images et des Signaux (LIS), INPG (Grenoble).

Azzag, H. (2005). *Classification hiérarchique par des fourmis artificielles : applications à la fouille de données et de textes pour le Web*. Ph. D. thesis, Université de Tours.

Azzag, H., N. Monmarché, M. Slimane, G. Venturini, et C. Guinot (2003). Anttree : a new model for clustering with artificial ants. In *IEEE Congress on Evolutionary Computation*, Canberra, Australia, pp. 2642-2647.

Blake, C. et C. Merz (1998). UCI repository of machine learning databases.

Di Battista, G., P. Eades, R. Tamassia, et I. G. Tollis (1998). *Graph Drawing : Algorithms for the Visualization of Graphs*. Upper Saddle River, NJ, USA : Prentice Hall PTR.

Eades, P. (1984). A heuristic for graph drawing. In *Congressus Numeranium*, Volume 42, pp. 140–160.

Ester, M., H.-P. Kriegel, et J. Sander (1997). Spatial data mining : A database approach, proceedings of the 5th symposium on spatial databases.

Frick, A., A. Ludwig, et H. Mehldau (1994). A fast adaptive layout algorithm for undirected graphs. In *DIMACS International Work. Graph Drawing, GD'94*.

Fruchterman, T. et E. Reingold (1991). Graph drawing by force-directed placement. In *Software - Practice and Experience*, Volume 21, pp. 1129–1164.

Gabriel, K. R. et R. R. Sokal (1969). A new statistical approach to geographic variation analysis. In *Systematic zoology*, Chapter 18, pp. 259–278.

Guinot, C., D. Malvy, J. Latreille, M. Tenenhaus, F. Morizot, S. Lopez, I. Le Fur, L. Dubertret, et E. Tschachler (2001). Recherche d'une classification de la peau du visage humaine saine à l'aide de la méthode de classification hiérarchique ascendante. In *VIIèmes Rencontres de la Société Francophone de Classification (SFC)*.

Hacid, H. et D. A. Zighed (2005). An effective method for locally neighborhood graphs updating. In *DEXA 2005*, pp. 930–939.

Kamada, T. et S. Kawai (1989). An algorithm for general undirected graphs. In *Information Processing Letters*, Volume 31, pp. 7–15.

Lioni, A., C. Sauwens, G. Theraulaz, et J.-L. Deneubourg (2001). The dynamics of chain formation in oecophylla longinoda. *Journal of Insect Behavior 14*, 679–696.

Preparata, F. et M. I. Shamos (1985). *Computational Geometry-Introduction*. Springer-Verlag.

Shneiderman, B. (1996). The eyes have it : A task by data type taxonomy for information visualizations. In *VL '96 : Proceedings of the 1996 IEEE Symposium on Visual Languages*, Washington, DC, USA, pp. 336. IEEE Computer Society.

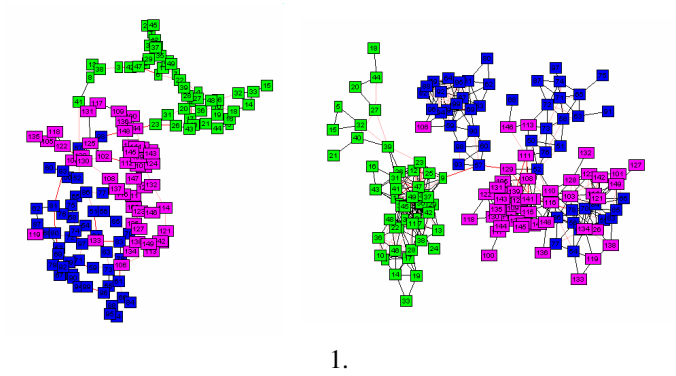
Toussaint, G. T. (1980). The relative neighborhood graphs in a finite planar set. In *Pattern recognition*, Chapter 12, pp. 261–268.

Toussaint, G. T. (1991). Some insolved problems on proximity graphs. In D. W. Dearholt et F. Harrary (Eds.), *Memoranda in computer and cognitive science MCCS-91-224*, Computing research laboratory, New Mexico state University Las Cruces.

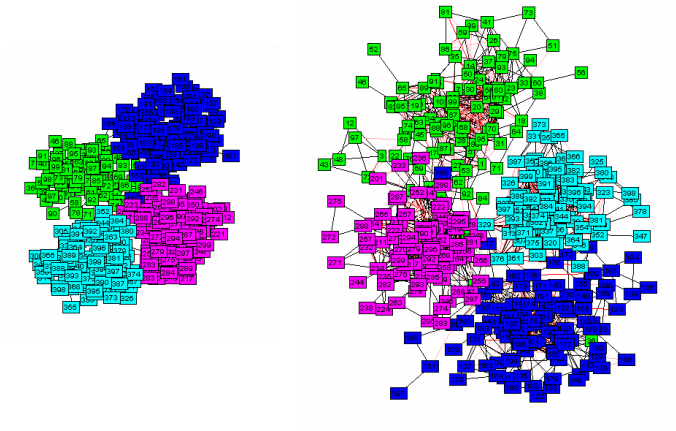
Tutte, W. T. (1963). How to draw a graph. In *Proceedings London Mathematical Society*, Volume 13, pp. 743–768. Tutte, W. T.

Summary

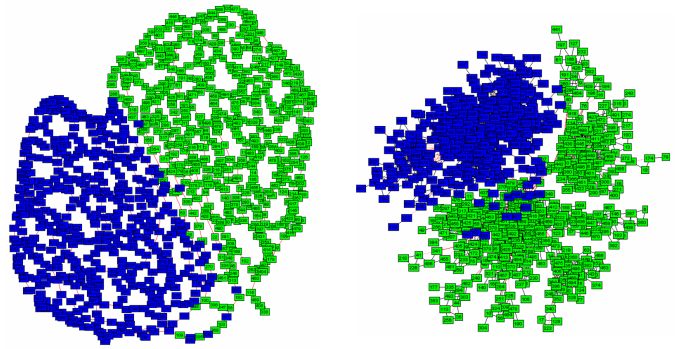
In this paper we present a new incremental algorithm for building neighborhood graphs between data. It is inspired from the self-assembling behavior observed in real ants where ants progressively become attached to an existing support and then successively to other attached ants. We use an artificial ants approach. Each ant represents one data. The way ants move and build a graph depends on the similarity between the data. We have compared our results to those obtained by the relative neightborhood algorithm on numerical databases (either artificial, real, or from the CE.R.I.E.S.), and we show that our method is competitive especially with respect to execution times.



1.

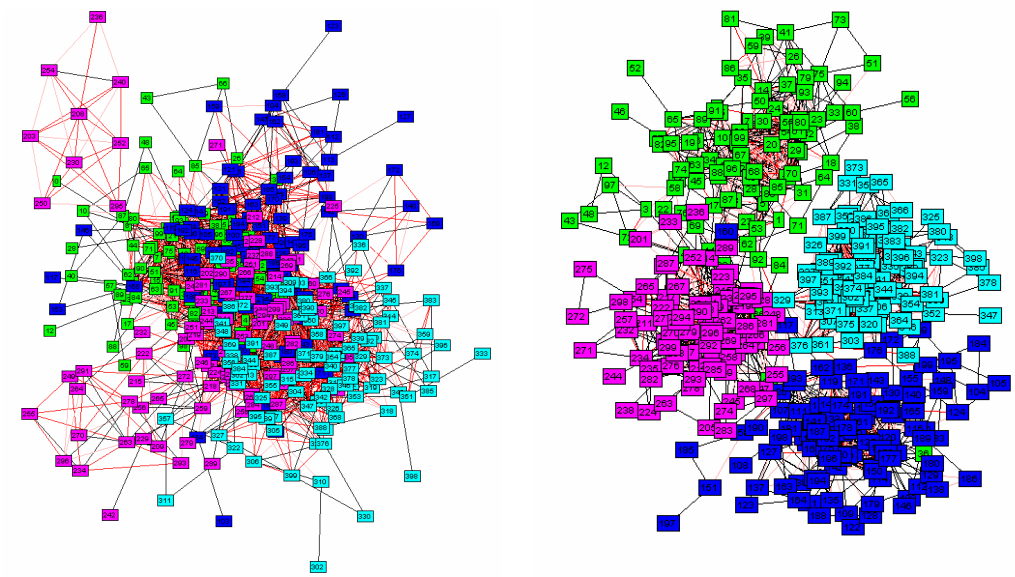


2.

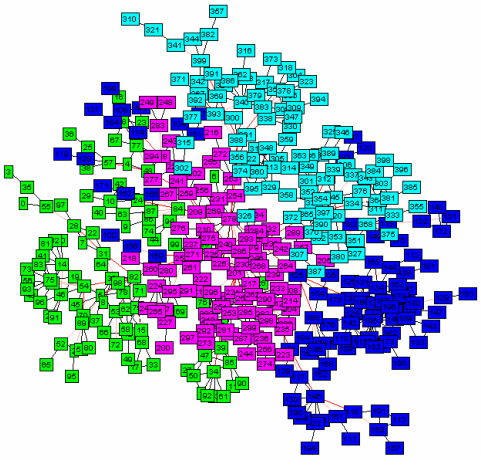


3.

FIG. 2 – *Comparaison Voisins Relatifs (à gauche) avec AntGraph (à droite) pour les bases 1. Iris, 2. Art6 et 3. Art2.*



$\alpha = 0,8$ $\alpha = 0,9$



$\alpha = 0,97$

FIG. 3 – *Extrait de visualisations de la base Art6 pour laquelle nous avons fait évoluer la constante α intervenant dans le calcul du seuil de tolérance S_t.*