Clustering Web Search Results Using Fuzzy Ants

Steven Schockaert,* Martine De Cock,† Chris Cornelis,‡ Etienne E. Kerre§
Department of Applied Mathematics and Computer Science,
Ghent University, Fuzziness and Uncertainty Modelling Research Unit,
Krijgslaan 281 (S9), B-9000 Gent, Belgium

Algorithms for clustering Web search results have to be efficient and robust. Furthermore they must be able to cluster a data set without using any kind of a priori information, such as the required number of clusters. Clustering algorithms inspired by the behavior of real ants generally meet these requirements. In this article we propose a novel approach to ant-based clustering, based on fuzzy logic. We show that it improves existing approaches and illustrates how our algorithm can be applied to the problem of Web search results clustering. © 2007 Wiley Periodicals, Inc.

1. INTRODUCTION

Most existing Web search engines respond to a user's query by returning an ordered list of links to Web pages that are considered relevant. The majority of these queries consist of only a few keywords, which are often ambiguous or too general for accurately expressing the user's information need. As a consequence, typically only a small fraction of the search engine results are really relevant. In this way, users are often forced to sift through a long list of search results to find the information they are looking for. Algorithms for clustering Web search results try to overcome this problem by converting the output of an existing search engine to a list of labeled clusters. Well-known clustering algorithms such as k-means and the more general fuzzy c-means depend on an initial estimation of the number of clusters (i.e., k or c). Hence they are not very suitable in the context of search results clustering where such a priori information is not at our disposal. Other clustering algorithms such as agglomerative hierarchical clustering (AHC) are too slow for online clustering of Web search results.

*Author to whom all correspondence should be addressed: e-mail: Steven.Schockaert@UGent.be.
†e-mail: Martine.DeCock@UGent.be.
‡e-mail: Chris.Cornelis@UGent.be.
§e-mail: Etienne.Kerre@UGent.be.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 22, 455–474 (2007)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.20209

Ant-based clustering algorithms are usually inspired by the clustering of dead nestmates, as observed with several ant species under laboratory conditions.[1] Without negotiating about where to gather the corpses, ants manage to cluster all corpses into one or two piles. The conceptual simplicity of this phenomenon together with the lack of centralized control and a priori information are the main motivations for designing a clustering algorithm inspired by this behavior. However, most ant-based clustering algorithms are only suitable for a visual representation of the data on a two-dimensional grid (e.g., Ref. 2) or require a hybridization with a classical clustering algorithm such as *k*-means (e.g., Ref. 3) or AHC (e.g., Ref. 4).

Real ants are, because of their very limited brain capacity, often assumed to reason only by means of rules of thumb.[5] Inspired by this observation, we propose a clustering method in which the desired behavior of artificial ants (and, more precisely, their stimuli for picking up and dropping items) is expressed flexibly by fuzzy IF–THEN rules. In this way, we obtain a genuine clustering algorithm in which hybridization with a classical clustering algorithm becomes superfluous. Moreover, because no a priori information on the number of clusters is needed, our algorithm is a very suitable candidate for the task of clustering the results of a search engine. Note that although we use fuzzy IF–THEN rules, the result of our algorithm is a crisp clustering. This article is an extended version of Refs. 6 and 7. As an extension of our previous work, in this article we compare results of our algorithm with other clustering techniques and, most importantly, we show its applicability for the clustering of Web documents.

The article is organized as follows: In Section 2, we review existing approaches to ant-based clustering, in particular the algorithm of Monmarché, which served as our main source of inspiration. In Section 3 we outline the structure of our clustering algorithm and motivate its key design principles. A comparison of our algorithm with other clustering methods is presented in Section 4. In Section 5, we apply our algorithm to the clustering of Web search results. Finally, Section 6 offers some concluding remarks.

2. RELATED WORK ON ANT-BASED CLUSTERING

Deneubourg et al.[1] proposed an agent-based model to explain the clustering behavior of real ants. In this model, artificial ants (or agents) are moving randomly on a square grid of cells on which some items are scattered. Each cell can only contain a single item, and each ant can move the items on the grid by picking up and dropping these items with a certain probability that depends on an estimation of the density of items of the same type in the neighborhood. Lumer and Faieta[8] extended the model of Deneubourg et al., using a dissimilarity-based evaluation of the local density to make it suitable for data clustering. Unfortunately, the resulting number of clusters is often too high and convergence is slow. Therefore, a number of modifications were proposed, by Lumer and Faieta themselves as well as by others (e.g., Refs. 2 and 9).

Monmarché[3] proposed an algorithm called AntClass in which several items are allowed to be on the same cell. Each cell with a nonzero number of items

corresponds to a cluster. Each (artificial) ant a is endowed with a certain capacity $c(a)$. Instead of carrying one item at a time, an ant a can carry a heap of $c(a)$ items. Let $U = \{x_1, x_2, \ldots, x_n\}$ be the set of all objects to be clustered and d a dissimilarity measure on U, that is, for x and y in U, $d(x, y)$ expresses to what extent x and y are dissimilar. Let H be an arbitrary heap, that is, a nonempty subset of U, with center c_H. We do not specify how to define the center of a heap at this point, because this definition depends on the intended application. Concrete definitions will be given in Sections 4 and 5. We define

$$d_{avg} = \frac{1}{n^2} \sum_{1 \le i, j \le n} d(x_i, x_j)$$

$$d_{max} = \max_{1 \le i, j \le n} d(x_i, x_j)$$

$$\bar{d}(H) = \frac{1}{|H|} \sum_{x \in H} d(x, c_H)$$

$$d^*(H) = \max_{x \in H} d(x, c_H)$$

where d_{avg} is the average dissimilarity between the items in U, d_{max} is the maximal dissimilarity between the items in U, $\bar{d}(H)$ is the average dissimilarity between the items of the heap H and the center of H, and $d^*(H)$ is the maximal dissimilarity between the items of H and the center of H. When an unloaded ant comes to a cell that contains a heap H with center c_H, it will pick up this heap with a probability $P_p(H)$ defined by

$$P_p(H) = \begin{cases} 1 & \text{if } |H| = 1 \\ \min\left(\left(\dfrac{\bar{d}(H)}{d_{avg}}\right)^{k_1}, 1\right) & \text{if } |H| = 2 \\ 1 - 0.9\left(\dfrac{\bar{d}(H) + \epsilon}{d^*(H) + \epsilon}\right)^{k_1} & \text{otherwise} \end{cases}$$

where ϵ is a small positive real constant and k_1 is a positive integer. When an ant, loaded with a heap L, comes to a cell with a heap H, it will drop the heap L onto this cell with a probability $P_d(L, H)$ defined by

$$P_d(L, H) = \begin{cases} 1 & \text{if } d(c_L, c_H) \le d^*(H) \\ 1 - 0.9 \min\left(\left(\dfrac{d(c_L, c_H)}{d_{max}}\right)^{k_2}, 1\right) & \text{otherwise} \end{cases}$$

where k_2 is an integer constant, c_H is the center of H, and c_L is the center of L. Monmarché proposed to apply this algorithm twice. The first time, the capacity of all ants is 1, which results in a high number of tight clusters. Subsequently the

algorithm is repeated with the clusters of the first pass as atomic objects and ants with infinite capacity to obtain a smaller number of large clusters. After each pass, k-means clustering is applied for handling small classification errors. In a similar way, in Ref. 10, an ant-based clustering algorithm is combined with the fuzzy c-means algorithm. Although some work has been done on combining fuzzy rules with ant-based algorithms for optimization problems,[11] to our knowledge until now, fuzzy IF–THEN rules have not yet been used to control the behavior of artificial ants in a clustering algorithm.

For completeness we also mention some approaches to ant-based clustering that are not inspired by the clustering of dead nestmates. For example, in Ref. 12, a clustering algorithm inspired by the chemical recognition system of ants is given, Ref. 13 introduces a clustering algorithm inspired by the self-assembling behavior of certain ant species, and in Ref. 14, a clustering algorithm is given that is inspired by the way ants manage to find the shortest path to a food source.

3. FUZZY ANT-BASED CLUSTERING

3.1. Some Preliminaries from Fuzzy Set Theory

A major asset of humans is their flexibility in dealing with imprecise, granular information, that is, their ability to abstract from superfluous details and to concentrate instead on more abstract *concepts* (represented by words from natural language). One way to allow a machine to mimic such behavior is to construct an explicit interface between the abstract symbolic level (i.e., linguistic terms like "high," "old," etc.) and an underlying, numerical representation that allows for efficient processing; this strategy lies at the heart of fuzzy set theory,[15] which, since its introduction in the 1960s, has rapidly acquired an immense popularity as a formalism for the representation of vague, linguistic information, and which in this article we exploit as a convenient vehicle for constructing commonsense rules that guide the behavior of artificial ants in our clustering algorithm.

Let us recall some basic definitions. A fuzzy set A in a universe U is a mapping from U to the unit interval $[0,1]$. For any u in U, the number $A(u)$ is called the membership degree of u to A; it expresses to what extent the element u exhibits the property A. A fuzzy set R in $U \times V$ is also called a fuzzy relation from U to V. Fuzzy relations embody the principle that elements may be related to each other to a certain extent only. When $U = V$, R is also called a binary fuzzy relation in U. Classical set theory is tightly linked to Boolean logic, in a sense that, for example, the operations of set complement, intersection, and union are defined by means of logical negation, conjunction, and disjunction, respectively. This link is also maintained under the generalization from $\{0,1\}$ to $[0,1]$. For instance, to extend Boolean conjunction, a wide class of operators called t-norms is at our disposal: a t-norm is any symmetric, associative, increasing $[0,1]^2 \to [0,1]$ mapping T satisfying $T(1,x) = x$ for every $x \in [0,1]$. Common t-norms include the minimum and the product in $[0,1]$, but also the Łukasiewicz t-norm T_W, which has several desirable properties (see, e.g., Ref. 16) and which is defined by, for x, y in $[0,1]$,

$$T_W(x, y) = \max(0, x + y - 1) \tag{1}$$

Another prominent contribution of fuzzy set theory is the ability to perform *approximate reasoning*. In particular, we may summarize flexible, generic knowledge in a fuzzy rulebase like

$$\begin{array}{c}\text{IF } X \text{ is } A_1 \text{ and } Y \text{ is } B_1 \text{ THEN } Z \text{ is } C_1\\ \text{IF } X \text{ is } A_2 \text{ and } Y \text{ is } B_2 \text{ THEN } Z \text{ is } C_2\\ \cdots\\ \text{IF } X \text{ is } A_n \text{ and } Y \text{ is } B_n \text{ THEN } Z \text{ is } C_n\end{array}$$

where X, Y, and Z are variables taking values in the respective universes U, V, and W, and where for i in $\{1, \ldots, n\}$, A_i (resp. B_i and C_i) is a fuzzy set in U (resp. V and W). Our aim is then to deduce a suitable conclusion about Z for every specific input of X and Y. This can, of course, be generalized to an arbitrary number of variables in the antecedent and the consequent. Numerous approaches exist to implement this, with varying levels of sophistication; for our purposes, we use the conceptually simple and very efficient Mamdani method.[17]

3.2. Outline of the Algorithm

Our algorithm is in many ways inspired by the algorithm of Monmarché.[3] We will consider however only one ant, because the use of multiple ants on a nonparallel implementation has no advantages. Note, however, that the proposed changes do not exclude the use of multiple ants.

Monmarché's algorithm involves a pass in which ants can only pick up one item as well as a pass during which ants can only pick up an entire heap. In our algorithm the ants are "intelligent" in the sense that they decide for themselves whether to pick up one item or an entire heap. This makes a separation of the clustering in different passes superfluous, hence giving rise to a more elegant algorithm. The underlying principle is a model of division of labor in social insects by Bonabeau et al.[18] In this model, a certain stimulus and a response threshold value are associated with each task a (real) ant can perform. The response threshold value is fixed, but the stimulus can change and represents the need for the ant to perform the task. As will become clear later, we will calculate the values of the stimuli by evaluating fuzzy IF–THEN rules. The probability that an ant starts performing a task with stimulus s and response threshold value θ is given by

$$T_n(s; \theta) = \frac{s^n}{s^n + \theta^n} \tag{2}$$

where n is a positive integer. In fact, this is a slight generalization that was also used in Ref. 2; in Ref. 18 only the case where $n = 2$ is considered. We will assume that $s \in [0, 1]$ and $\theta \in]0, 1]$.

Let us now apply this model to the problem at hand. A loaded ant can only perform one task: dropping its load. Let s_{drop} be the stimulus associated with this

task and θ_{drop} the response threshold value. The probability of dropping the load is then given by

$$P_{drop} = T_{n_i}(s_{drop}; \theta_{drop}) \tag{3}$$

where $i \in \{1, 2\}$ and n_1, n_2 are positive integers. When the ant is only carrying one item n_1 is used; otherwise n_2 is used. An unloaded ant can perform two tasks: picking up one item and picking up all the items. Let s_{one} and s_{all} be the respective stimuli and θ_{one} and θ_{all} the respective response threshold values. The probabilities for picking up one item and picking up all the items are given by

$$P_{pickup\_one} = \frac{s_{one}}{s_{one} + s_{all}} \cdot T_{m_1}(s_{one}; \theta_{one}) \tag{4}$$

$$P_{pickup\_all} = \frac{s_{all}}{s_{one} + s_{all}} \cdot T_{m_2}(s_{all}; \theta_{all}) \tag{5}$$

where m_1 and m_2 are positive integers.

We assume that the objects that have to be clustered belong to some set U, and that E is a binary fuzzy relation in U, which is reflexive (i.e., $E(u,u) = 1$, for all u in U) and T_W-transitive (i.e., $T_W(E(u,v), E(v,w)) \leq E(u,w)$, for all u, v, and w in U). For u and v in U, $E(u,v)$ denotes the degree of similarity between the items u and v.

During the execution of the algorithm, we maintain a list of all heaps. Initially there is a heap, consisting of a single element, for every object in the data set. Picking up an entire heap H corresponds to removing a heap from the list. At each iteration our ant acts as follows:

- If the ant is unloaded, a heap H from the list is chosen at random.
  - If H consists of a single item, this item is always picked up.
  - If H consists of two items, a and b, both items are picked up with probability $E(a,b)^{k_1}$ and one of the two items is picked up with probability $(1 - E(a,b))^{k_1}$.
  - If H consists of more than two items, the probabilities for picking up a single element and for picking up all elements are given by formulas (4)–(5).
- If the ant is loaded, a new heap containing the load L is added to the list of heaps with a fixed probability. Otherwise, a heap H from the list is chosen at random.
  - If H consists of a single item a and L consists of a single item b, L is dropped onto H with probability $E(b,a)^{k_2}$.
  - If H consists of a single item and L consists of more than one item, the ant does nothing. The main reason for separating this special case is efficiency. Because the average similarity $avg(H)$ will always be 1 in this case, the only situation where it would be desirable to merge H and L is when all the items in L are approximately equal to the single element in H. But in this unlikely case, L and H would be merged at a later iteration of the algorithm.
  - If H consists of more than one item, the probability that L is dropped onto H is given by formula (3).

In the above, k_1 and k_2 are small integer constants.

3.3. Computing the Stimuli

For a nonempty heap $H \subseteq U$ with center c in U, we define the average and minimal similarity of H, respectively, by

$$avg(H) = \frac{1}{|H|} \sum_{h \in H} E(h,c) \qquad \min(H) = \min_{h \in H} E(h,c) \tag{6}$$

Furthermore, let $E^*(H_1, H_2)$ be the similarity between the centers of the heap H_1 and the heap H_2.

Dropping items. The stimulus for a loaded ant to drop its load L on a cell that already contains a heap H is based on the average similarity $A = avg(H)$ and an estimation of the average similarity between the center of H and items of L. This estimation is calculated as $B = T_W(E^*(L,H), avg(L))$, which is a lower bound due to our assumption about the T_W-transitivity of E, because

$$\begin{aligned}
T_W(E^*(L,H), avg(L)) &= T_W\left(E(c_L, c_H), \frac{1}{|L|} \sum_{l \in L} E(l, c_L)\right) \\
&= \max\left(0, E(c_L, c_H) + \frac{1}{|L|} \sum_{l \in L} E(l, c_L) - 1\right) \\
&= \max\left(0, \frac{1}{|L|} \sum_{l \in L} (E(c_L, c_H) + E(l, c_L) - 1)\right) \\
&\leq \frac{1}{|L|} \sum_{l \in L} \max(0, E(c_L, c_H) + E(l, c_L) - 1) \\
&= \frac{1}{|L|} \sum_{l \in L} T_W(E(c_L, c_H), E(l, c_L)) \\
&\leq \frac{1}{|L|} \sum_{l \in L} E(l, c_H)
\end{aligned}$$

where c_L (resp. c_H) is the center of L (resp. c_L). Moreover, B can be implemented much more efficiently than the exact value. If B is smaller than A, the stimulus for dropping the load should be low; if B is greater than A, the stimulus should be high. Because heaps should be able to grow, we should also allow the load to be dropped when A is approximately equal to B. Our ant will perceive the values of A and B to be very high (VH), high (H), medium (M), low (L), or very low (VL). The stimulus will be perceived as very very high (VVH), very high (VH), high (H), rather high (RH), medium (M), rather low (RL), low (L), very low (VL), or very very low (VVL). These linguistic terms can be represented by fuzzy sets in $[0,1]$. For example, to represent the linguistic terms for the stimulus value, we used the fuzzy sets in Figure 1. The rules for the stimulus for dropping the load L onto an existing heap H are summarized in Table I.

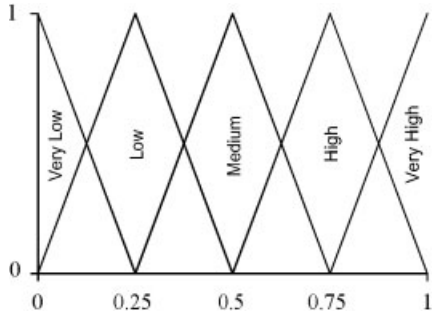

Figure 1. Fuzzy sets in $[0,1]$ representing the linguistic terms for the stimulus value.

Picking up items. An unloaded ant should pick up the most dissimilar item from a heap if the similarity between this item and the center of the heap is far less than the average similarity of the heap. This means that by taking the item away, the heap will become more homogeneous. An unloaded ant should only pick up an entire heap, if the heap is already homogeneous. Thus, the stimulus for an unloaded ant to pick up a single item from a heap H and the stimulus to pick up all items from that heap are based on the average similarity $A = avg(H)$ and the minimal similarity $M = min(H)$. The stimulus for picking up an entire heap can be inferred using the fuzzy rules in Table II; the stimulus for picking up a single item can be inferred using the fuzzy rules in Table III. Because the average similarity A will always be higher than the minimal similarity M, elements above the main diagonal in Tables II and III are empty. Although we can tolerate that an ant wrongly picks up an item from a homogeneous heap once in a while, we should avoid that it picks up a heterogeneous heap. Therefore, the rules in Table III are more strict than those in Table II.

4. COMPARISON WITH OTHER CLUSTERING ALGORITHMS

Before applying our algorithm to the clustering of Web search results, we compare it with other clustering techniques. More in particular we show that our approach is not only more elegant than Monmarché's (because it does not involve

Table I. Fuzzy rules to infer the stimulus for dropping the load.

	A				
B	VH	H	M	L	VL
VH	RH	H	VH	VVH	VVH
H	L	RH	H	VH	VVH
M	VVL	L	RH	H	VH
L	VVL	VVL	L	RH	H
VL	VVL	VVL	VVL	L	RH

Table II. Fuzzy rules to infer the stimulus for picking up a heap.

	A				
M	VH	H	M	L	VL
VH	VVH	—	—	—	—
H	M	VH	—	—	—
M	L	RL	H	—	—
L	VVL	VL	L	RH	—
VL	VVL	VVL	VVL	VL	M

passes or hybridization with other clustering techniques), but it is also able to generate improved results. As test cases for evaluating our algorithm, we use the same artificial and real-world data sets as Monmarché. In these data sets the n objects to be clustered are characterized by m numerical attributes, that is, $U = \{u_1, \ldots, u_n\}$ with $u_i \in \mathbb{R}^m$, $i = 1, \ldots, n$. We define the center of a heap $H = \{h_1, h_2, \ldots, h_p\}$, with $h_i = (h_i^1, h_i^2, \ldots, h_i^m)$ for $1 \leq i \leq p$, as the center of gravity c_H, that is,

$$c_H = \left(\frac{1}{p} \sum_{i=1}^{p} h_i^1, \frac{1}{p} \sum_{i=1}^{p} h_i^2, \ldots, \frac{1}{p} \sum_{i=1}^{p} h_i^m \right)$$

To compute the similarity between vectors, we use the fuzzy relation E in U defined by, for u_i and u_j in U,

$$E(u_i, u_j) = 1 - \frac{d(u_i, u_j)}{d_{max}} \tag{7}$$

where d represents Euclidean distance and d_{max} is (an estimation of) the maximal distance between objects from U, that is, $d_{max} = diam\ U$. Note that we have to use an approximation of d_{max}, because the calculation of the exact value requires a number of steps that is quadratic in the size of the data set. Initially, this approximation is set to the maximal distance between n randomly chosen pairs of objects from U. Every time a higher distance is calculated during the execution of the algorithm, the approximation is adjusted. Note that E is indeed reflexive because d is reflexive, and T_W-transitive because for arbitrary a, b, and c in $[0, 1]$ we have

Table III. Fuzzy rules to infer the stimulus for picking up a single item.

	A				
M	VH	H	M	L	VL
VH	M	—	—	—	—
H	H	RH	—	—	—
M	VVH	VH	H	—	—
L	VVH	VVH	VVH	VH	—
VL	VVH	VVH	VVH	VVH	VVH

$$
\begin{aligned}
T_W(E(a,b),E(b,c)) &= \max(0, E(a,b)+E(b,c)-1) \\
&= \max\left(0, 1-\frac{d(a,b)}{d_{max}}+1-\frac{d(b,c)}{d_{max}}-1\right) \\
&= \max\left(0, 1-\frac{d(a,b)+d(b,c)}{d_{max}}\right) \\
&\leq \max\left(0, 1-\frac{d(a,c)}{d_{max}}\right) \\
&= 1-\frac{d(a,c)}{d_{max}} \\
&= E(a,c)
\end{aligned}
$$

where we have used the triangle inequality $d(a,b)+d(b,c) \geq d(a,c)$.

All response threshold values were set to 0.5, that is, the modal value of the fuzzy set representing the linguistic term "medium." The parameters k_1 and k_2 should be given a small integer value; we choose $k_1 = k_2 = 5$. Clearly, the most important parameters are the parameters n_1, n_2, m_1, and m_2, which reflect the degree of randomness of the algorithm. For very large values of these parameters, the corresponding tasks are performed almost for certain when the stimulus value is higher than 0.5. Because picking up an item or a heap has less drastic consequences than dropping a heap or an item, we can tolerate more randomness for m_1 and m_2 than for n_1 and n_2. Moreover, because dropping an item has less drastic consequences than dropping an entire heap, we can tolerate more randomness for n_1 than for n_2. Therefore, we impose $m_1 = m_2 < n_1 < n_2$; we will use $(m_1, m_2, n_1, n_2) = (5,5,10,20)$ in the remainder of this article.

To evaluate the algorithm, we compare the obtained clusters with the correct classification of the objects. For u in U, let $k(u)$ be the (unique) class that u belongs to and $c(u)$ the heap u was put in after algorithm execution. Following Monmarché,[3] we define the classification error F_c by

$$
F_c = \frac{1}{|U|^2} \sum_{1 \leq i,j \leq n} \epsilon_{ij} = \frac{2}{|U|(|U|-1)} \sum_{1 \leq i < j \leq n} \epsilon_{ij} \tag{8}
$$

with

$$
\epsilon_{ij} = \begin{cases} 0 & \text{if } (k(u_i) = k(u_j) \text{ and } c(u_i) = c(u_j)) \text{ or } (k(u_i) \neq k(u_j) \text{ and } c(u_i) \neq c(u_j)) \\ 1 & \text{otherwise} \end{cases} \tag{9}
$$

As an important benefit, this evaluation criterion strongly penalizes a wrong number of clusters.[3]

Table IV. Artificial data sets.

Name	Structure
ART1	$(100;\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2))$, $(100;\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2))$, $(100;\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2))$, $(100;\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2))$
ART2	$(500;\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2))$, $(500;\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2))$
ART3	$(500;\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2))$, $(500;\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2))$, $(50;\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2))$, $(50;\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2))$
ART4	$(100;\mathcal{U}(-1,1),\mathcal{U}(-10,10))$, $(100;\mathcal{U}(2,3),\mathcal{U}(-10,10))$
ART5	$(100;\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2))$, $(100;\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2))$, $(100;\mathcal{N}(1.4,0.2),\mathcal{N}(0.2,0.2))$, $(100;\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2))$, $(100;\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2))$, $(100;\mathcal{N}(1.4,0.2),\mathcal{N}(0.8,0.2))$, $(100;\mathcal{N}(0.2,0.2),\mathcal{N}(1.4,0.2))$, $(100;\mathcal{N}(0.8,0.2),\mathcal{N}(1.4,0.2))$, $(100;\mathcal{N}(1.4,0.2),\mathcal{N}(1.4,0.2))$
ART6	$(100;\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.2,0.2))$, $(100;\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2))$, $(100;\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.2,0.2))$, $(100;\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2),\mathcal{N}(0.8,0.2))$

Let $(n;\mathcal{N}(a_1,b_1),\mathcal{U}(a_2,b_2),\ldots,\mathcal{N}(a_m,b_m))$ denote the structure of a class of n objects that are represented as m-dimensional vectors. The first component of this class has a Gaussian distribution with a mean value of a_1 and a standard deviation of b_1, the second component is uniformly distributed in the interval $[a_2,b_2]$, and so forth. The structure of the artificial data sets ARTi is summarized in Table IV. In Table V we compare our algorithm with k-means, AHC (agglomerative hierarchical clustering), and some other ant-based clustering algorithms: AntClass

Table V. Classification error F_c for our fuzzy ant algorithm after 10^6 iterations and some other clustering algorithms for artificial data sets.

	ART1	ART2	ART3	ART4	ART5	ART6
Fuzzy ants	0.15	0.07	0.12	0.24	0.23	0.01
AntClass	0.15	0.41	0.35	0.29	0.08	0.11
AntClust	0.22	0.07	0.15	0.23	0.26	0.05
AntTree$_{Local}$	0.15	0.36	0.25	0.26	0.21	0.03
AntTree$_{Stoch}$	0.18	0.19	0.21	0.26	0.21	0.34
AntTree$_{No\text{-}Threshold}$	0.19	0.33	0.24	0.13	0.16	0.12
k-means	0.11	0.04	0.23	0.00	0.07	0.00
AHC	0.15	0.04	0.14	0.00	0.42	0.00

Note: Unlike the other algorithms, k-means is given the correct number of clusters and the best possible initial partitioning.

Table VI. Real-world data sets.

	Wine	Iris	Glass	Soybean	Thyroid	Pima
Number of classes	3	3	2/6	4	3	2
Number of attributes	12	4	9	35	5	8
Number of objects	178	150	214	47	215	768

(Monmarché's algorithm), AntClust,[12] and three different variants of the AntTree algorithm.[13] All results are averaged over 50 runs. The results for *k*-means and Monmarché's approach are those reported in Ref. 3. The initial partitioning of the *k*-means algorithm corresponds to the correct classification of the data set. Although the other algorithms do not know the number of clusters or an initial partitioning, *k*-means is given the correct number of clusters and the best possible initial partitioning. The results for AntClust are those reported in Ref. 12, and the results for AntTree and AHC are those reported in Ref. 13. Because both AHC and AntTree are hierarchical clustering algorithms, heuristics are necessary to obtain a flat clustering; for more details we refer to Ref. 13.

For most of the data sets *k*-means and AHC give the best results. For ART3, however, the fuzzy ants provide the best clustering. The results for our algorithm are better than those of Monmarché's algorithm, except for ART5. For ART5, some of the nine clusters are taken together by our algorithm. Given the structure of this data set, this seems to us as a valid alternative clustering, rather than a mistake. Moreover, for most of the data sets the results of our algorithm are better than the results of AntClust and AntTree.

As real-world data sets, we took the "Wine," "Iris," "Glass," "Soybean," "Thyroid," and "Pima" data sets from the UCI Machine Learning Repository.[19] The main characteristics of these data sets are shown in Table VI. The results are presented in Table VII. For the "Glass" data set, the classification error was computed with respect to all six classes. The classification error with respect to the two main

Table VII. Classification error F_c for our fuzzy ant algorithm after 10^6 iterations and some other clustering algorithms for real-world data sets.

	Wine	Iris	Glass	Soybean	Thyroid	Pima
Fuzzy ants	0.13	0.16	0.36	0.11	0.18	0.44
AntClass	0.51	0.19	0.40	0.54	0.22	0.47
AntClust	n.a.	0.22	0.36	0.46	0.07	0.16
AntTree_{Local}	0.18	0.18	0.33	0.12	0.40	0.50
AntTree_{Stoch}	0.32	0.25	0.42	0.07	0.25	0.43
$\text{AntTree}_{No\text{-}Threshold}$	0.32	0.17	0.26	0.02	0.45	0.51
k-means	0.28	0.13	0.32	0.00	0.18	0.44
AHC	0.07	0.22	0.40	0.00	0.16	0.48

Note: Unlike the other algorithms, *k*-means is given the correct number of clusters and the best possible initial partitioning.

classes for our algorithm is 0.12, which shows that our algorithm discovers the two main clusters rather than all six subclusters. Clearly, the results for our algorithm are an improvement over Monmarché's algorithm. Moreover, our results are comparable with AHC and the ideal results for k-means. Recall that the computational complexity of AHC is too high for most real-world applications and k-means was given the correct number of clusters.

5. CLUSTERING WEB SEARCH RESULTS

Cutting et al.[20] introduced the idea to use a document clustering algorithm as a tool for representing search results in an organized way. Because it is unlikely that a user is prepared to wait more than a few seconds, algorithms for clustering Web search results must rely solely on the short summaries that are returned by the search engine. Zamir and Etzioni[21] showed that using only the snippets returned by a search engine, instead of using full documents, results in a small decrease in performance only. A fast linear-time document clustering algorithm called STC-clustering is given. This algorithm, however, cannot discover the number of clusters in a reliable way. Instead, the algorithm returns a fixed number of clusters. An algorithm for visualizing search engine results on a topic map using an ant-based clustering algorithm was suggested in Ref. 9. In Ref. 22, a similar ant-based clustering algorithm for visualizing a collection of documents is given. Because these algorithms have to use some kind of grid representation to present the results, they are not suitable for large document collections. In Ref. 13, the AntTree algorithm is applied to the automatic generation of portal sites. However, only the clustering of full documents is considered.

5.1. Similarity of Snippets

Each snippet returned by a search engine can be treated as a small document. As a consequence, standard document similarity measures can be used to calculate the similarity of two search results. Let R be a fuzzy relation from the (finite) universe $\mathcal{D}$ of documents to the (finite) universe $\mathcal{T}$ of terms. For d in $\mathcal{D}$ and t in $\mathcal{T}$, $R(d, t)$ denotes the importance of term t in document d. The importance of a term t in a document d can be calculated in a lot of different ways, most of which incorporate the frequency of occurrence of t in d, and the inverse of the document frequency, that is, the number of documents that contain t. We will, however, use a binary weighting scheme, in which $R(d, t) = 1$ if d contains t, and $R(d, t) = 0$ otherwise. Indeed, using the frequency of occurrence of a term does not make much sense in this context, due to the very small size of the documents; penalizing terms that occur in many documents does not make much sense either, because we are trying to discover the most important topics in the document collection. We will assume that no documents in $\mathcal{D}$ are empty, that is, for all d in $\mathcal{D}$, there exists a term t in $\mathcal{T}$ such that $R(d, t) > 0$. Furthermore, we will assume that every term is contained in at least one document. To improve performance, the Porter stemmer (http://www.tartarus.org/~martin/PorterStemmer/) is used to remove some common morphological and inflectional endings from terms. Moreover, we use a list

of stopwords to discard some very common words (e.g., *he*, *and*, *have*, etc.) and discard terms that occur in only one or two snippets.

Although standard similarity measures, such as the cosine similarity and weighted Jaccard similarity, have proven effective for document clustering, they are not suitable for our purpose because of the small size of the documents. The snippets of two very similar documents may contain entirely different terms. To overcome this limitation we propose an alternative measure, inspired by the notion of upper approximation from fuzzy rough set theory.[23] The idea is that before comparing two documents, we first add all terms that are more specific (to some degree) than terms that already occur in the documents. First we define the binary fuzzy narrower than relation $N^{\mathcal{T}}$ in the universe of terms $\mathcal{T}$:

$$N^{\mathcal{T}}(t_1, t_2) = \frac{\sum_{d \in \mathcal{D}} \min(R(d, t_1), R(d, t_2))}{\sum_{d \in \mathcal{D}} R(d, t_1)}$$

for all t_1 and t_2 in $\mathcal{T}$. The underlying idea is that term t_1 is more specific than term t_2 in the degree to which documents that contain t_1 also contain t_2. Next we extend the original document–term relation R to obtain a new fuzzy relation $\hat{R}$:

$$\hat{R}(d, t) = \sup_{t' \in \mathcal{T}} \min(R(d, t'), N^{\mathcal{T}}(t, t'))$$

for all d in $\mathcal{D}$ and t in $\mathcal{T}$. For the comparison of snippets d_1 and d_2 we use

$$N^{\mathcal{D}}(d_1, d_2) = \frac{\sum_{t \in \mathcal{T}} \min(\hat{R}(d_1, t), \hat{R}(d_2, t))}{\sum_{t \in \mathcal{T}} \hat{R}(d_1, t)} \tag{10}$$

Note that Equation (10) is an inclusion measure, rather than a similarity measure, in the sense that two documents will be clustered together if they are similar or if one of them is more specific than the other. Another alternative is given by

$$E^{\mathcal{D}}(d_1, d_2) = \frac{\sum_{t \in \mathcal{T}} \min(\hat{R}(d_1, t), \hat{R}(d_2, t))}{\sum_{t \in \mathcal{T}} \max(\hat{R}(d_1, t), \hat{R}(d_2, t))} \tag{11}$$

In fact, Equation (11) is the weighted Jaccard similarity, which is known to be T_W-transitive,[24] calculated with respect to the extended document–term relation. In practice, however, we found that Equation (10) resulted in a better overall performance than its symmetrical, T_W-transitive counterpart. This is in accordance

with the observations made in Ref. 25, where the use of asymmetrical measures, instead of symmetrical measures for document and term clustering is advocated. We conclude this section with a small example to illustrate the usefulness of our measure. For experimental results we refer to Section 5.3.

Example. Consider the following five snippets:

Rem Koolhaas - Great Buildings Online
Rem Koolhaas oma, netherlands architect in the Great Buildings Online.

1999 Laureate Announcement
Rem Koolhaas of The Netherlands Is the Pritzker Architecture Prize ... Los Angeles, CA—Rem Koolhaas, a 56 year old architect from the Netherlands, ...

Who Is Rem Koolhaas
The controversial Dutch architect Rem Koolhaas has won the presitigious Pritzker Prize, but who is he? Join us on a virtual tour of his life and works, ...

Rem Koolhaas [en]
Information on Rem Koolhaas and his Office for Metropolitan Architecture (OMA) archived at the ArchINFORM database.

Faculty Profile
... Rem Koolhaas is Professor in Practice of Architecture and Urban Design. ... In 1975, Rem Koolhaas founded the Office for Metropolitan Architecture with ...

Removing stopwords, query terms ("Rem Koolhaas"), and terms that occur only once, we obtain

$$d_1 = \{\text{oma, netherlands, architect}\}$$

$$d_2 = \{\text{netherlands, pritzker, architecture, prize, architect}\}$$

$$d_3 = \{\text{architect, pritzker, prize}\}$$

$$d_4 = \{\text{office, metropolitan, architecture, oma}\}$$

$$d_5 = \{\text{architecture, office, metropolitan}\}$$

For the simplicity of this example, we did not apply stemming. Also, we did not discard terms that occur only twice. Using traditional similarity measures *sim* such as the cosine similarity, $sim(d_1, d_5) = 0$ because d_1 and d_5 do not have a single term in common. However, all five documents are related to Rem Koolhaas. The extended document–term relation $\hat{R}$ is shown in Table VIII; using Equation (10) we obtain

$$N^{\mathcal{D}}(d_1, d_5) = 0.578$$

Table VIII. An example of the extended document–term relation $\hat{R}$.

	t							
d	oma	netherlands	architect	pritzker	prize	architecture	office	metropolitan
d_1	1	1	1	1	1	0.33	0.5	0.5
d_2	0.5	1	1	1	1	1	1	1
d_3	0.5	1	1	1	1	0.33	0	0
d_4	1	0.5	0.33	0.5	0.5	1	1	1
d_5	0.5	0.5	0.33	0.5	0.5	1	1	1

5.2. Center of a Heap

We want to choose the most general document of a heap H as the center of this heap. A straightforward approach is to choose the document d for which

$$\sum_{d' \in H} N^{\mathcal{D}}(d', d)$$

is maximal. Unfortunately, this approach leads to an overall execution time of the algorithm that is at least quadratic in the number of search results to be clustered. Therefore, we propose another definition of the center, inspired by the notion of leader value.[25] For t in $\mathcal{T}$, we define the leader value $l^{\mathcal{T}}(t)$ of t as

$$l^{\mathcal{T}}(t) = \sum_{t' \in \mathcal{T}} N^{\mathcal{T}}(t', t)$$

$l^{\mathcal{T}}(t)$ reflects the generality of term t in the document collection. The leader value $l^{\mathcal{D}}(d)$ of a document d is defined as

$$l^{\mathcal{D}}(d) = \sum_{t \in \mathcal{T}} R(d, t) \cdot l^{\mathcal{T}}(t)$$

As the center of a heap, we choose the document for which the leader value is maximal, that is, the document that contains the most general terms.

5.3. Experimental Results

We have implemented our approach to Web search results clustering as a part of the Carrot^2 framework. Carrot^2 is an open source Web search results clustering framework, developed by Dawid Weiss at the university of Poznań. We refer readers to the project of Carrot^2 (http://carrot2.sourceforge.net/) for the source code and an online demonstration of our algorithm.

To allow a hierarchical clustering, we apply our algorithm recursively to all clusters. To assign a label to a cluster C, we determine the most representative term t of the snippets in the cluster, that is, the term t in $\mathcal{T}$ for which

$$\sum_{d \in C} \hat{R}(d, t) \tag{12}$$

is maximal. Moreover, we add extra terms to the right (resp. left) if these terms occur at the right (resp. left) of t in at least 75% of the occurrences of t. To obtain the experimental results in Tables V and VII, we have used a fixed number of iterations. In practice, however, the ideal number of iterations will be dependent on both the nature and size of the data set. Experimental results have indicated that $c \cdot n$ is a good estimation of the number of iterations that is required, where n is the size of the data set and c is a parameter that is dependent on the nature of the data set. For clustering Web search results, we used $c = 5000$.

In Figure 2 the resulting cluster structure of our algorithm for some sample queries is shown. For each query, the first 500 snippets returned by Google (http://www.google.com) were clustered. For the query "rem" among others, three clusters are found concerning the rock band REM ("music," "rock," and "lyrics"), two clusters concerning the rem-statement that is found in many programming languages ("syntax rem" and "program"), one cluster concerning the rem sleep, and

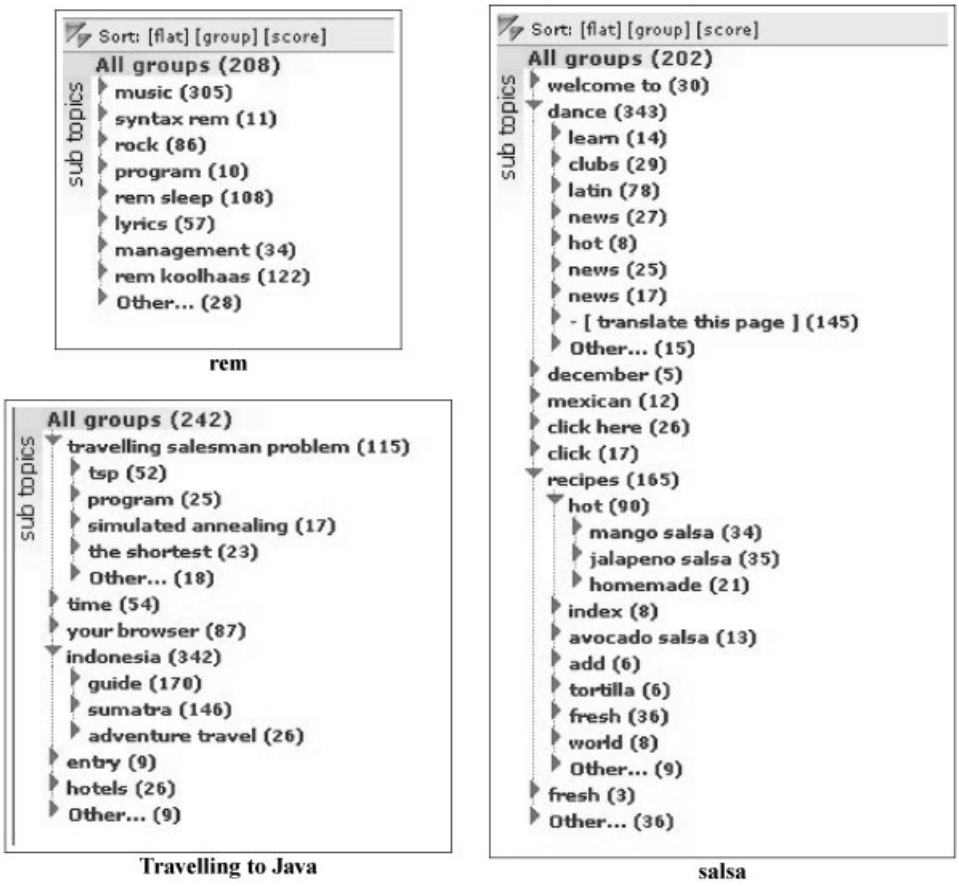

Figure 2. Cluster structure for some sample queries.

one cluster about a famous architect called Rem Koolhaas. The query "Travelling to Java" results in two important types of search results: results concerning implementations of the Traveling Salesman Problem in the programming language Java and results concerning travelling information about the island of Java in Indonesia. The query "Salsa" can refer to salsa cooking and salsa dancing.

The execution time for the query "Salsa" as a function of the number of snippets to be clustered is shown in Figure 3. This test was performed on a Pentium 4, 1.4-GHz PC with 512 MB of internal memory. From this we can see that our algorithm is scalable to large data sets and sufficiently fast for online clustering. Similar results were found for other queries.

6. CONCLUSIONS

We have presented a clustering algorithm, inspired by the behavior of real ants simulated by means of fuzzy IF–THEN rules. Like all ant-based clustering algorithms, no initial partitioning of the data is needed, nor should the number of clusters be known in advance. The machinery of approximate reasoning from fuzzy set theory endows the ants with some intelligence. As a result, throughout the whole clustering process, they are capable of deciding for themselves to pick up either one item or a heap. Hence the two phases of Monmarché's original idea are smoothly merged into one, and k-means clustering becomes superfluous. To apply our algorithm to the problem of Web search results clustering, we introduced an appropriate measure to assess the relationship between two documents, inspired by the notion of upper approximation from fuzzy rough set theory and a definition of the center of a heap that can be calculated in an efficient way. The usefulness of clustering Web search results in the presence of ambiguity was illustrated by some sample queries.

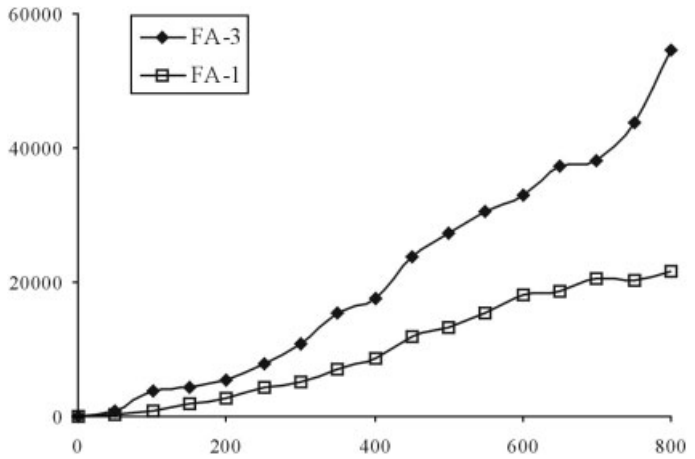

Figure 3. Execution time for the query "salsa" for a nonrecursive version of the algorithm (FA-1) and for a recursive version with depth limit 3 (FA-3).

Acknowledgments

Steven Schockaert and Chris Cornelis would like to thank the Research Foundation–Flanders for funding their research.

References

1. Deneubourg JL, Goss S, Franks N, Sendova-Franks A, Detrain C, Chrétien L. The dynamics of collective sorting robot-like ants and ant-like robots. In: From Animals to Animats: Proc First Int Conf on Simulation of Adaptive Behaviour, Paris, 1990. Cambridge, MA: MIT Press; 1991. pp 356–363.
2. Ramos V, Muge F, Pina P. Self-organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies. In: Abraham A, Ruis-del-Solar J, Köppen M, editors. Soft computing systems: Design, management and applications. Amsterdam: IOS Press; 2002. pp 500–509.
3. Monmarché N. Algorithmes de Fourmis Artificielles: Applications à la Classification et à l'Optimisation. Ph.D. thesis. Tours, France: Université François Rabelais; 2000.
4. Handl J. Ant-based methods for tasks of clustering and topographic mapping: Extensions, analysis and comparison with alternative techniques. Master thesis. Nuremberg, Germany: University of Erlangen–Nuremberg; 2003.
5. Hölldobler B, Wilson EO. The ants. Heidelberg: Springer-Verlag; 1990.
6. Schockaert S, De Cock M, Cornelis C, Kerre EE. Efficient clustering with fuzzy ants. In: Ruan D, D'Hondt P, De Cock M, Nachtegael M, Kerre EE, editors. Applied computational intelligence. Singapore: World Scientific; 2004. pp 195–200.
7. Schockaert S, De Cock M, Cornelis C, Kerre EE. Fuzzy ant based clustering. 4th Int Workshop on Ant Colony Optimization and Swarm Intelligence, Brussels, Belgium. Lecture Notes in Computer Science 3172. Berlin, Heidelberg, Germany: Springer-Verlag; 2004. pp 342–349.
8. Lumer ED, Faieta B. Diversity and adaptation in populations of clustering ants. In: From animals to animats 3: Proc Third Int Conf on the Simulation of Adaptive Behaviour, Brighton, UK, 1994. Cambridge, MA: MIT Press; 1994. pp 501–508.
9. Handl J, Meyer B. Improved ant-based clustering and sorting in a document retrieval interface. In: Merelo JJ, Adamidis P, Beyer H-G, editors. Proc Seventh Int Conf on Parallel Problem Solving from Nature, Granada, Spain. Lecture Notes in Computer Science 2723. Berlin, Heidelberg, Germany: Springer-Verlag; 2002. pp 913–923.
10. Kanade PM, Hall LO. Fuzzy ants as a clustering concept. In: Proc 22nd Int Conf of the North American Fuzzy Information Processing Society, Chicago, IL; 2003. pp 227–232.
11. Lučić P. Modelling transportation systems using concepts of swarm intelligence and soft computing. Ph.D. thesis. Blacksburg, VA: Virginia Polytechnic Institute and State University; 2002.
12. Labroche N, Monarché N, Venturini G. AntClust: Ant clustering and web usage mining. In: Proc Genetic and Evolutionary Computation Conf, Chicago, IL. Lecture Notes in Computer Science 2723. Berlin, Heidelberg, Germany: Springer-Verlag; 2003. pp 25–36.
13. Azzag H, Venturini G, Oliver A, Guinot C. A hierarchical ant based clustering algorithm and its use in three real-world applications. Eur J Oper Res 2007;179:906–922.
14. Tsai C-F, Wu H-C, Tsai C-W. A new data clustering approach for data mining in large databases. In: Proc Int Symp on Parallel Architectures, Algorithms and Networks, Makati City, Metro Manila, Philippines; 2002. pp 315–320.
15. Zadeh LA. Fuzzy sets. Inform Control 1965;8:338–353.
16. Klement EP, Mesiar R, Pap E. Triangular norms. Dordrecht: Kluwer Academic Publishers; 2002.
17. Mamdani EH, Assilian S. An experiment in linguistic synthesis with a fuzzy logic controller. Int J Man Mach Stud 1975;7:1–13.

18. Bonabeau E, Sobkowski A, Theraulaz G, Deneubourg JL. Adaptive task allocation inspired by a model of division of labor in social insects. Working paper 98-01-004; 1998. Available at http://ideas.repec.org/p/wop/safiwp/98-01-004.html.
19. Blake CL, Merz CJ. UCI repository of machine learning databases. University of California; 1998. Available at: http://www.ics.uci.edu/~mlearn/MLRepository.html.
20. Cutting DR, Karger DR, Pedersen JO, Tukey JW. Scatter/gather: A cluster-based approach to browsing large document collections. In: Proc 15th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval, Copenhagen, Denmark; 1992. pp 318–329.
21. Zamir O, Etzioni O. Web document clustering: A feasibility demonstration. In: Proc 21st Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval, Melbourne, Australia; 1998. pp 46–54.
22. Bin W, Yi Z, Shaohui L, Zhongzhi S. CSIM: A document clustering algorithm based on swarm intelligence. In: Proc 2002 Congress on Evolutionary Computation, Honolulu, Hawaii; 2002. pp 477–482.
23. Radzikowska AM, Kerre EE. A comparative study of fuzzy rough sets. Fuzzy Set Syst 2002;126:137–156.
24. De Baets B, De Meyer H. The Frank t-norm family in fuzzy similarity measurement. In: Proc Second EUSFLAT Conf, Leicester, UK; 2001. pp 249–252.
25. Krishna K, Krishnapuram R. A clustering algorithm for asymmetrically related data with applications to text mining. In: Proc 10th Int Conf on Information and Knowledge Management, Atlanta, GA; 2001. pp 571–573.