Mach Learn (2011) 82: 1–42
DOI 10.1007/s10994-010-5216-5

Editorial survey: swarm intelligence for data mining

David Martens · Bart Baesens · Tom Fawcett

Received: 22 April 2010 / Revised: 24 August 2010 / Accepted: 25 August 2010 /
Published online: 17 September 2010

Abstract This paper surveys the intersection of two fascinating and increasingly popular domains: swarm intelligence and data mining. Whereas data mining has been a popular academic topic for decades, swarm intelligence is a relatively new subfield of artificial intelligence which studies the emergent collective intelligence of groups of simple agents. It is based on social behavior that can be observed in nature, such as ant colonies, flocks of birds, fish schools and bee hives, where a number of individuals with limited capabilities are able to come to intelligent solutions for complex problems. In recent years the swarm intelligence paradigm has received widespread attention in research, mainly as Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO). These are also the most popular swarm intelligence metaheuristics for data mining. In addition to an overview of these nature inspired computing methodologies, we discuss popular data mining techniques based on these principles and schematically list the main differences in our literature tables. Further, we provide a unifying framework that categorizes the swarm intelligence based data mining algorithms into two approaches: effective search and data organizing. Finally, we list interesting issues for future research, hereby identifying methodological gaps in current research as well as mapping opportunities provided by swarm intelligence to current challenges within data mining research.

Editor: Foster Provost.

D. Martens
Department of Business Administration and Public Management, University College Ghent, Ghent University, Ghent, Belgium

D. Martens (✉) · B. Baesens
Department of Decision Sciences & Information Management, K.U. Leuven, Leuven, Belgium
e-mail: David.Martens@econ.kuleuven.be

B. Baesens
School of Management, University of Southampton, Southampton, UK

T. Fawcett
Proofpoint, Inc., Sunnyvale, CA, USA

Keywords Swarm intelligence · Ant colony optimization · Particle swarm optimization · Data mining

1 Introduction

1.1 Swarm intelligence

Swarm intelligence studies the collective behavior of systems composed of many individuals interacting locally with each other and with their environment. Swarms inherently use forms of decentralized control and self-organization to achieve their goals (Dorigo 2007). Researchers in computer science have developed swarm-based systems in response to the observed success and efficiency of swarms in nature to solve difficult problems. In such biological swarms, the individuals (ant, bee, termite, bird or fish) are by no means complete engineers, but instead are simple creatures with limited cognitive abilities and limited means to communicate. Yet the complete swarm exhibits intelligent behavior, providing efficient solutions for complex problems such as predator evasion and shortest path finding.

For example, ants communicate only indirectly through their environment by leaving behind a substance called pheromone which attracts other ants. Based on this indirect communication, shortest paths between the food source and the nest are found, even in the event of changing environments and failure of individual ants. Bees seeking a new location for their beehive is another example. Initially, several scouts are sent out to scope potential locations. Upon returning, they perform a specific dance that encodes the direction of the new found site. The strength of the bee's dance indicates the enthusiasm for the specific location. As soon as enough scouts vote for the same location, the whole swarm moves.

By mimicking nature inspired swarming behavior in computing methodologies, techniques emerge for hard optimization problems that are robust, scalable and easily distributed. A distinction can be made between Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and prey models. The last decade has seen an increasing use of nature inspired computing techniques in engineering applications. Successful applications of swarm intelligence include the modeling of agent behavior (such as the large numbers of fighting individuals in the battle scenes of the movie *Lord of the Rings*[1]), and various optimization problems, such as the routing of packages through networks (Caro and Dorigo 1998), the traveling salesman problem (Dorigo et al. 1996), scheduling (Blum 2005a), robotics (Dorigo 2009) and data mining, the topic of this survey. Figure 1 demonstrates that the number of papers published on these topics (as indexed by the Web of Science[2]) shows a sharp increase since early 2000. Although the paper counts indicate that PSO seems most popular, ACO is also increasing in popularity. Prey models show an increase in popularity as well.

1.2 Swarm Intelligence for Data Mining

The popularity of swarm intelligence has also instigated the development of numerous data mining algorithms, which will be discussed in this overview. Although many differences exist among the proposed techniques, Fig. 2 provides a first attempt for a unifying framework of data mining techniques based on swarm intelligence.

Broadly speaking, we observe two categories: the first category consists of techniques where individuals of a swarm move through a solution space and look for solutions (or so-

[1] www.massivesoftware.com.

[2] www.isiknowledge.com.

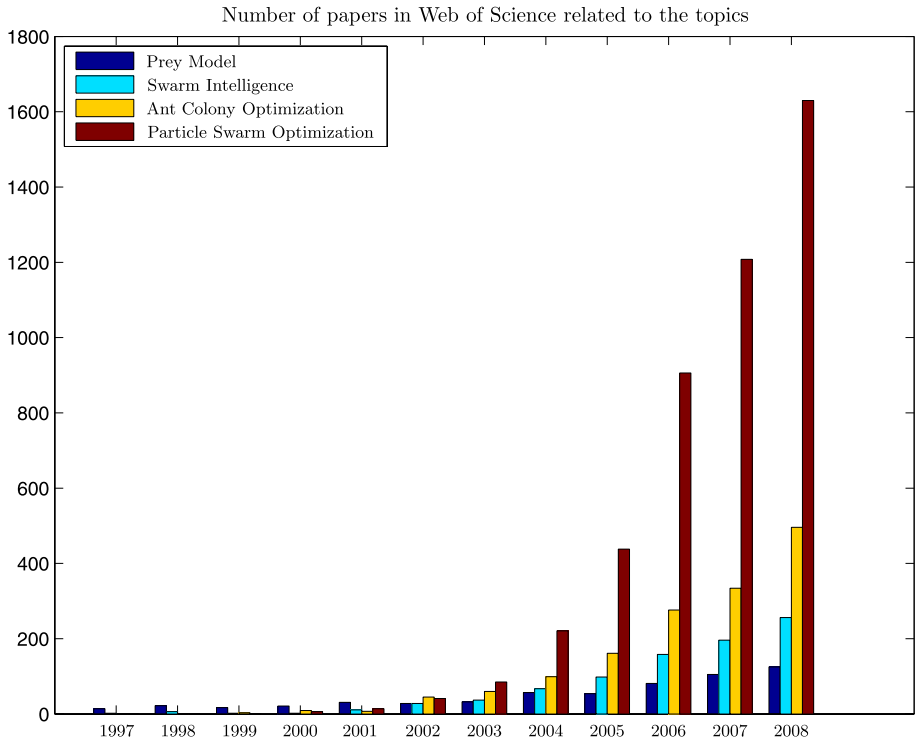

Fig. 1 Number of papers written on the subject, from Web of Science

lution components) for the data mining task at hand, an approach we name *effective search*. The ACO and PSO approaches consist of individuals wandering through the search space in some effective manner that combines exploration and exploitation. For ACO, this search space is discrete and a solution is defined implicitly by the path taken by an ant. For PSO the search space is continuous and the locations within the search space are updated explicitly. In the second category, named *data organizing*, swarms move data instances that are placed on a low-dimensional (typically two-dimensional) feature space in order to come to a suitable clustering or low-dimensional mapping solution of the data. In this category of clustering techniques fall ant-based sorting and prey models.

A high-level algorithmic description of these approaches is described in Algorithms 1 and 2. Both approaches start by defining the environment in which the swarm individuals will operate, followed by an initialization of problem parameters. Such parameters could include a solution instantiation that is gradually constructed in the first approach, or the number of clusters in the second approach. All swarm intelligence based techniques first initialize the search space parameters (such as pheromone levels or ant, particle or prey location on the low dimensional map), after which (some) individuals of the swarm commence their task of creating data mining solutions, optimizing some defined objective function. The objective functions that are optimized vary greatly, even across algorithms of the same approach. The choice made for each algorithm is listed in Tables 1 and 2 of Sect. 5. In Algorithms 1 and 2, the main differences between the two approaches are denoted in italic font. Firstly, the effective search approach allows for swarms to construct solution components, which is not observed in the second approach. The main difference however is the manner

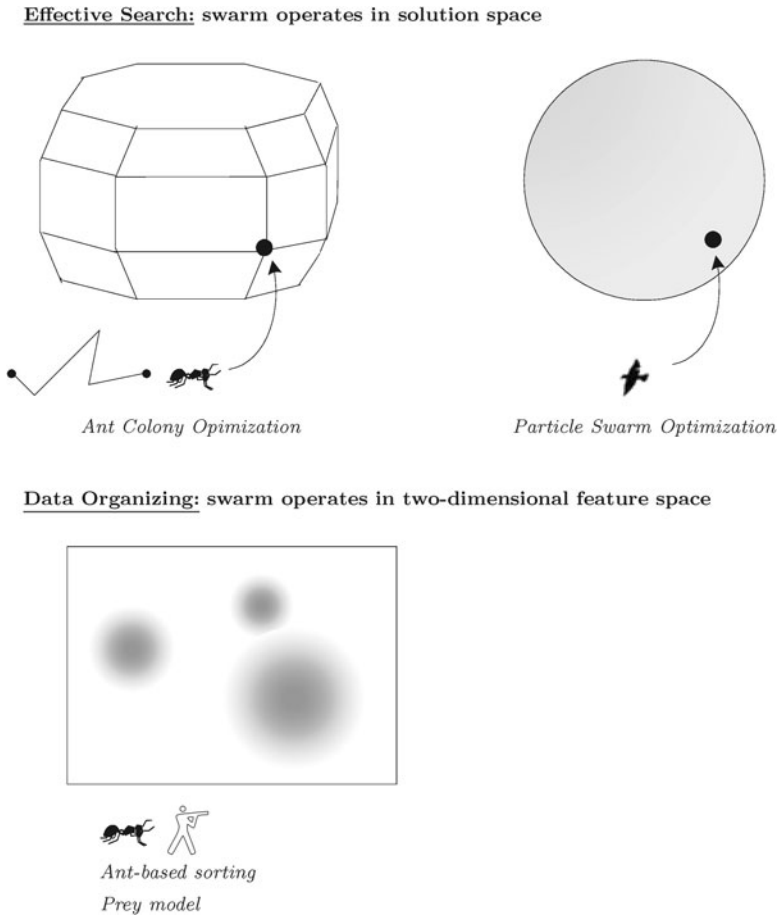

Fig. 2 General framework for swarm intelligence for data mining. The *effective search* approach includes techniques using the ACO and PSO metaheuristic, which comprise of individuals wandering through the search space in some effective manner that combines exploration/exploitation. For ACO, this search space is discrete and a solution is defined implicitly by the path taken by an ant. For PSO the search space is continuous and the locations within the search space are updated explicitly. The second *data organizing* approach moves data items in a two-dimensional feature space, such that similar data items are grouped together. In this category fall ant-based sorting and prey models

in which solution (components) are constructed: either by effectively searching through the solution space, or by organizing data in a low dimensional map.[3]

Many models in computer science have been inspired by nature, but not all of them may be considered swarm intelligence. Swarms generally involve movement of individuals through a representation space, and not all nature inspired algorithms do this. For example, artificial neural networks and evolutionary algorithms are biologically inspired algorithms (based on principles of neuroscience and evolution, respectively), and have been used for data mining, but they are not swarm intelligence. Neural networks are distributed but static; evolutionary algorithms are based on reproduction rather than movement. Similarly, artifi-

[3]Typically this map is two-dimensional to allow for easy visualization.

Algorithm 1 *Effective Search* Approach

```
1: Define search space such that location defines a solution (component)
2: Set problem parameters
3: while no complete solution do
4:    Initialize search space parameters
5:    while no convergence of swarm do
6:       Select some swarm individuals
7:       for all selected individuals in swarm do
8:          Walk through search space effectively exploring new regions while exploiting
            good solutions found so far, optimizing objective function f
9:       end for
10:      Update search space parameters
11:   end while
12:   Add solution (component) and update problem parameters
13: end while
```

Algorithm 2 *Data Organizing* Approach

```
1: Define two-dimensional feature space F
2: Set problem parameters
3: Initialize search space parameters
4: while no convergence of swarm do
5:    Select some swarm individuals
6:    for all selected individuals in swarm do
7:       Update neighboring data item(s)' location(s), optimizing objective function f
8:    end for
9:    Update search space parameters
10: end while
```

cial immune systems are based on lymphocyte cloning and mutation, rather than swarming behavior. All of these nature inspired techniques have been applied to data mining problems but are not included in this survey.[4]

The sections of this overview are organized according to the underlying biological phenomenon: ant colonies in Sect. 2, bird flocking in Sect. 3, and foraging theory in Sect. 4. For each paradigm, we discuss the biological background, followed by the mapping to a computing method, the most popular data mining implementations, and finally challenges for the future. Section 5 summarizes all the techniques with regard to implementation and experimental details in Tables 1 and 2. Based on these, we propose a number of interesting directions for future research in Sect. 5. Finally, Sect. 6 concludes the paper.

Notation-wise, we assume to have a dataset of n data instances and m dimensions. The ith data instance is denoted as $\mathbf{x}_i$, with value for dimension j being x_{ij}. In case of classification, we have c classes. For clustering, k denotes the number of clusters and the centroid of cluster l is $\mathbf{z}_l$.

[4]For these other areas we recommend the following resources. Bishop (1996) has published an excellent overview of neural network techniques for pattern recognition. A survey of evolutionary algorithms for data mining may be found in Freitas (2003). Freitas and Timmis (2007) have surveyed artificial immune systems from a data mining perspective.

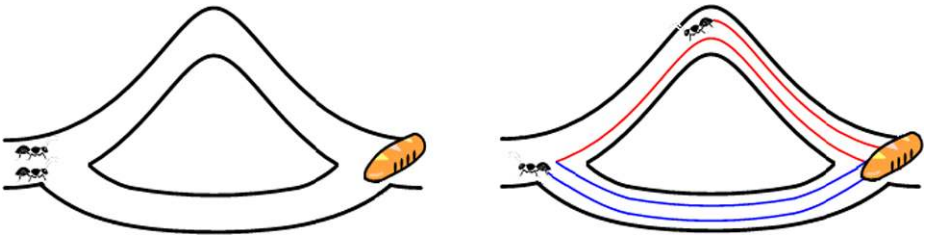

Fig. 3 Path selection with ants using pheromone

2 Ant-based computing

2.1 Foraging ant behavior

2.1.1 Biological background

Foraging ant behavior is an intriguing phenomenon by which ant colonies find the shortest path between food source and nest. Ants communicate not directly with each other, but rather indirectly through their environment. This indirect communication, known as *stigmergy*, allows a colony of ants with limited memory and capabilities to come to intelligent solutions for complex problems. More specifically, ants communicate by dropping a substance called pheromone on their path, thereby providing a feedback mechanism to attract other ants. Paths with higher pheromone levels are more likely to be chosen and thus reinforced. On the other hand, the pheromone trail intensity of paths that are not chosen, is decreased by evaporation.

These principles are illustrated in Fig. 3 for finding the shortest path between a food source (right) and the nest (left). Two ants start from their nest (left) and look for the shortest path to a food source (right). Initially, no pheromone is present on either trails, so there is a 50–50 chance of choosing either of the two possible paths. Suppose one ant chooses the lower trail, and the other one the upper trail. The ant that has chosen the lower (shorter) trail will have returned faster to the nest, resulting in twice as much pheromone on the lower trail as on the upper one. As a result, the probability that the next ant will choose the lower, shorter trail will be twice as high, resulting in more pheromone; thus more ants will choose this trail, until eventually (almost) all ants will follow the shorter path.

2.1.2 Artificial ant colonies and ant colony optimization

The same ideas are applied in artificial ant systems (Dorigo et al. 1991): a number of computational concurrent and asynchronous agents move through their environment and by doing so incrementally construct a solution for the problem at hand. Ants move by applying a stochastic local decision policy based on two parameters: the pheromone value and the heuristic value. The *pheromone* value of a trail is history dependent, and gives an indication of the number of ants that passed through the trail recently. The *heuristic* value is a problem dependent quality value. When an ant comes to a decision point, it is more likely to choose the trail with the higher pheromone and heuristic values. Once the ant arrives at its destination, the solution corresponding to the ant's followed path is evaluated and the pheromone value of the path is updated accordingly. Additionally, evaporation causes the pheromone level of all trails to diminish gradually. Hence, trails that are not reinforced will gradually lose pheromone and will in turn have a lower probability of being chosen by subsequent ants.

Note that real ant colonies only use the pheromone values and no heuristic values exist. However, background problem-dependent information allows us to guide the search with the heuristic values.

The general steps are summarized in the generic Ant Colony Optimization (ACO) metaheuristic shown in Algorithm 3. The implementation of a specific ACO system requires specifying the following aspects:

- An environment that represents the problem domain in such a way that it lends itself to incrementally building a solution for the problem;
- A problem dependent heuristic evaluation function (η), which provides a quality measurement for the different solution components;
- A pheromone updating rule, which takes into account the evaporation and reinforcement of the trails;
- A probabilistic transition rule based on the value of the heuristic function (η) and on the strength of the pheromone trail (τ) that determines the path taken by the ants;
- A convergence criterion that determines when the algorithm terminates.

Algorithm 3 ACO Metaheuristic

```
Set parameters, initialize pheromone trails
while termination condition not met do
    Construct ant solutions
    Apply local search % Optional
    Update pheromones
end while
```

The first algorithm that followed these ACO principles was the Ant System, which was introduced in the context of the traveling salesman problem (TSP) (Dorigo et al. 1996). This NP-complete problem entails finding the shortest path that visits each given city exactly once (Lawler et al. 1985). The construction graph is fully connected and corresponds to the map of cities, with each city corresponding to a node and possible edges being the connection between the cities. Initially an ant starts in a random city. The next city j for an ant to choose (city j), given it is currently in city i and has already visited the cities in partial solution s_p, is given by (1), with α and β being weight parameters that determine the importance of the pheromone and heuristic function respectively. Only cities that have not yet been visited are considered, and constitute the neighborhood $N(s_p)$. The heuristic function $\eta(ij)$ is defined by the inverse distance between cities i and j: $1/d_{ij}$, hence, the closer the city the more probable it is to be chosen by the ant. A good initial pheromone value for all the edges should be slightly higher than the amount added at each iteration, estimated roughly by $\tau_{ij} = \tau_0 = no\_ants/C^{NN}$ with C^{NN} the length of a tour generated by a nearest neighbor heuristic. The quality of the solution s is defined by the evaluation function $f(s) = 1/g(s)$ and measures the inverse of the total length of the tour $g(s)$. After each ant has constructed its solution, pheromone values are updated according to (2), where C is a constant. This equation states that the shorter the path chosen by the ant, the more pheromone is added to each of the edges corresponding to the path.

$$P(j|s_p, i) = \frac{\tau_{ij}^{\alpha} \cdot \eta(ij)^{\beta}}{\sum_{l \in N(s_p)} \tau_{il}^{\alpha} \cdot \eta(il)^{\beta}}, \quad \forall j \in N(s_p) \tag{1}$$

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + f(s) \cdot C \tag{2}$$

Although the performance of Ant System did not compete with traditional algorithms, it is considered to be the basis of many other ACO algorithms that do achieve competitive or superior performance for many applications. The main improvement in subsequent ACO algorithms is the balance between a better exploitation of the best solution and an exploration of the solution space, effectively avoiding early stagnation. For example, the commonly used $\mathcal{MAX}$-$\mathcal{MIN}$ ant system differs from the traditionally proposed Ant System in three aspects (Stützle and Hoos 1996, 2000). First, after each iteration only the best ant is allowed to add pheromone to its trail. This allows for a better exploitation of the best solution found. Also, to avoid early stagnation of the search, the range of possible pheromone trails is limited to an interval $[\tau_{\min}, \tau_{\max}]$. Finally, the initial pheromone value of each trail is set at $\tau_{\max}$. This determines a higher exploration at the beginning of the algorithm.

Other ACO algorithms include Elitist Ant System (Dorigo et al. 1996), Ant-Q (Gambardella and Dorigo 1995), Ant Colony System (Dorigo and Gambardella 1997), ANTS (Maniezzo 1999), Best-Worst Ant System (Cordon et al. 2002), Population-based ACO (Guntsch and Middendorf 2002), Touring ACO (Hiroyasu et al. 2000) and Beam-ACO (Blum 2005b). A detailed overview of these variants can be found in Dorigo and Stützle (2004) and Dorigo and Stützle (2009).

ACO has been applied to a variety of different problems (Dorigo and Stützle 2004), such as vehicle routing (Wade and Salhi 2004; Bullnheimer et al. 1999; Montemanni et al. 2005), scheduling (Colorni et al. 1994; Blum 2005a), timetabling (Socha et al. 2002), traffic light control (de Oliveira and Bazzan 2006), and routing in packet-switched networks (Caro and Dorigo 1998).

2.1.3 Data mining applications of ACO

ACO within the data mining community has been used primarily for supervised classification. Classification is an important data mining task, where the value of a discrete (dependent) variable is predicted based on the values of several independent variables. Although ACO has been used for clustering (discussed at the end of this subsection), the bulk of the research addresses classification. The most notable are the AntMiner rule induction techniques, as AntMiner was the first application of ACO to classification and AntMiner+ reports the overall best results. ACO has also been used in applications such as rule extraction, Bayesian network structure learning, and weight optimization in neural network training. All these techniques are discussed next.

AntMiner: ACO-based rule induction The first application of ACO for the classification task was reported in Parpinelli et al. (2001, 2002), where the authors introduce the AntMiner algorithm for the discovery of classification rules.[5] The aim is to induce simple rules of the form **if** *rule antecedent* **then** *rule consequent*, where the rule antecedent is a conjunction of terms. All attributes are assumed to be categorical; that is, the terms are of the form *Variable* = *Value*, e.g. *Gender* = *male*. AntMiner can be considered a *sequential covering* or *separate-and-conquer* algorithm where the induced classification model is an ordered rule list, meaning that the discovered rules are intended to be evaluated sequentially (Fürnkranz 1999). These algorithms start with learning one rule for the dataset (conquer step), next all data items covered by the rule are removed from the dataset (separate step) and a new rule is induced. This continues until a stopping criterion is met.

[5] A publicly available implementation of this technique is available at http://sourceforge.net/projects/guiantminer.

To apply the ACO metaheuristic, an environment should be defined such that when the ants move they incrementally construct a solution to the domain; in this case, rules for a classification problem. The AntMiner environment is defined as a directed graph, where for each variable there are as many nodes as there are values for that variable. Bidirectional edges exist between all nodes from different variables, resulting in a construction graph as shown in Fig. 4(a).

The AntMiner procedure is described in Algorithm 4. Each ant starts in the *Start* node with an empty rule and chooses an edge to follow, implicitly adding the term corresponding to that node to its rule. By moving through the environment, the ant adds one term at a time to its (partial) rule. This partial rule corresponds to the partial path it has followed so far. The edge to choose, and thus the term to add next, is dependent on the pheromone (τ_{ij}) and the heuristic value (η_{ij}) associated with each term $V_i = Value_{ij}$ and normalized over all possible terms. This probability P_{ij} of going to vertex $v_{i,j}$ is defined by (3), with x_i a binary variable set to 1 if variable V_i was not yet used by the current ant and 0 otherwise. The heuristic function provides a notion of the quality of that term and is chosen as an information theoretic measure in terms of the entropy, as defined by (4) and (5) with c the number of classes, T_{ij} the set of remaining (not yet covered by the rule list) data instances for which the ith variable equals its jth value $V_i = Value_{ij}$, and $|T_{ij}|$ the size of the dataset defined by T_{ij}. The edge to choose is additionally constrained since each variable can occur at most once in a rule to avoid inconsistencies such as $Gender = male$ **and** $Gender = female$.

$$P_{ij}(t) = \frac{\tau_{ij}(t) \cdot \eta_{ij}}{\sum_{k=1}^{m} x_k \sum_{l=1}^{p_k} (\tau_{kl}(t) \cdot \eta_{kl})} \tag{3}$$

$$\eta_{ij} = \frac{\log_2(c) - \mathrm{Info}(T_{ij})}{\sum_{i=1}^{m} x_i \sum_{j=1}^{p_i} (\log_2(c) - \mathrm{Info}(T_{ij}))} \tag{4}$$

$$\mathrm{Info}(T_{ij}) = -\sum_{w=1}^{c} (P(w|T_{ij}) \cdot \log_2 P(w|T_{ij})) \tag{5}$$

$$\tau_{ij}(t+1) = \frac{\tau_{ij}(t) + \tau_{ij}(t) \cdot Q}{\sum_{\forall ij \in rule} \tau_{ij}(t)} \tag{6}$$

The ant keeps adding terms to its partial rule either until all variables have been used in the rule or until adding any term would make the rule cover fewer cases than a user-defined minimum. The consequent of the rule is determined by the majority class of the training data covered by the rule. Finally, the rule is pruned in order to remove irrelevant terms and the pheromone levels are adjusted, increasing the pheromone of the trail followed by the ant and evaporating all others. Within AntMiner, the update rule for the pheromone function adds pheromone according to the rule quality Q, which is taken as the product of sensitivity and specificity, and normalizes over all terms as implicit evaporation mechanism (as shown by (6)). Then, another ant starts with the newly updated pheromone trails to guide its search. This process is repeated until all ants have constructed a rule or when a series of *no_same_rules_allowed* consecutive ants construct the same rule. The best rule among these constructed rules is added to the list of discovered rules and the training cases covered by this rule are removed from the training set. This process is repeated until the number of uncovered training cases is lower than a user-defined threshold.

AntMiner2 (Liu et al. 2002) builds further on AntMiner but uses a simpler, though less accurate, density estimation equation as the heuristic value. This makes AntMiner2 computationally less expensive without a significant degradation of the stated performance.

Algorithm 4 AntMiner algorithm for classification

```
TrainingSet = {all training data}
RuleList = {}
while |TrainingSet| > Max uncovered data instances do
    ant_index = 1
    cvg_index = 1
    Initialize probability, pheromone and heuristic values according to (3), (4) and (5)
    while (ant_index < no_ants) AND (cvg_index < no_same_rules_allowed) do
        Let ant run from source to sink, defining rule R_ant_index
        Prune rule R_ant_index
        Update pheromone on edges of path according to (6)
        if R_ant_index is the same as R_ant_index−1 then
            cvg_index = cvg_index + 1
        else
            cvg_index = 1
        end if
        Kill ant
        ant_index = ant_index + 1
    end while
    Choose best rule R_best among all induced rules R_ant_index
    Add rule R_best to RuleList
    TrainingSet = TrainingSet \ {data instances covered by R_best}
end while
Evaluate performance on test set
```

AntMiner3 also extends AntMiner by introducing two major changes (Liu et al. 2003), resulting in a reported increased accuracy. Firstly, a different update rule is used, defined by (7) with the quality Q of a rule set to the sum of its sensitivity and specificity, and ρ set to 0.1. Secondly, more exploration is encouraged by means of a different transition rule that increases the probability of choosing terms not yet used in previously constructed rules, as implemented by the Ant Colony System. Finally, it is worth noting that an extension of AntMiner that generates fuzzy rules has been proposed in Galea and Shen (2006).

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \left(1 - \frac{1}{1+Q}\right)\cdot\tau_{ij}(t) \tag{7}$$

AntMiner+: a novel environment, ACO algorithm and functions The AntMiner+ algorithm differs from the AntMiner versions in several ways (Martens et al. 2007).[6] The environment is defined as a directed acyclic graph (DAG), so that the ants can choose their paths more effectively (see Fig. 4(b)), while the AntMiner environment is fully connected (with the exception of links between nodes corresponding to the same variable). This means that the ants in AntMiner need to choose among all nodes at each decision point, whereas for AntMiner+ they only need to choose among the nodes corresponding to one variable. By reducing the choices without constraining the rule format, AntMiner+ ants can make their decisions more effectively. Furthermore, to allow for interval rules, the construction graph additionally exploits the difference between nominal and ordinal variables: each nominal

[6] A recent publicly available Matlab implementation of this technique is available at www.antminerplus.com.

variable has one node group (with the inclusion of a dummy vertex indicating the variable does not occur in the rule), but for the ordinal variables however, two node groups are built to allow for intervals to be chosen by the ants. The first node group corresponds to the lower bound of the interval and should thus be interpreted as $V_i \geq Value_{ij}$, the second node group determines the upper bound, giving $V_i \leq Value_{ij}$ (of course, the choice of the upper bound is constrained by the lower bound). This allows fewer rules, which are shorter and actually better. The environment also includes weight parameters for the pheromone and heuristic value, α and β, in the construction graph, which are therefore set by the ants themselves. Finally, the class variable is included as well (with all possible class values except the majority class, which serves as the final default rule) as to allow for multiclass datasets.

The differences in environment between AntMiner and AntMiner+ are clarified with a simplified credit scoring example in Fig. 4. The task at hand is distinguishing good from bad credit customers. Since this is a binary classification problem, the class variable in the AntMiner+ environment has only one value (the majority class is omitted). For the nominal variables 'Sex' and 'Real Estate' property, a dummy node 'any' is added, while for the ordinal variable 'Term', indicating the duration of the loan, two node groups are defined determining the lower and upper bound of the variable. Such a distinction is not made in the construction graph of AntMiner. An ant that has taken the path shown in boldface would have described the rule **if** *Sex* = *male* **and** *Term* $\geq$ *1 year* **and** *Term* $\leq$ *15 years* **then** *class* = *bad*. For the AntMiner environment such a rule would actually require four different rules, defined by the four paths shown in boldface.

Besides these changes in the environment, AntMiner+ also employs the better performing $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (Stützle and Hoos 2000), defines new heuristic and pheromone functions, and applies an early stopping criterion. Using early stopping, the rule induction is stopped as soon as the performance on an independent validation set (not used during rule induction) starts to decrease.

The reported benchmarking experiments show an AntMiner+ accuracy that is superior to that obtained by the other AntMiner versions, and competitive or better than the results achieved by the compared classification techniques, including C4.5, RIPPER and logistic regression (Martens et al. 2007). The main workings of this recent ACO-based classification technique are described in Algorithm 5.

The probability for an ant in vertex $v_{i-1,k}$ to choose the edge to vertex $v_{i,j}$ is defined by AntMiner+ as the pheromone value ($\tau_{(v_{i-1,k},v_{i,j})}$) of the edge, the heuristic value ($\eta_{v_{i,j}}$) of vertex $v_{i,j}$, and normalized over all possible edge choices:

$$P_{ij}(t) = \frac{[\tau_{(v_{i-1,k},v_{i,j})}(t)]^{\alpha}.[\eta_{v_{i,j}}(t)]^{\beta}}{\sum_{l=1}^{p_i}[\tau_{(v_{i-1,k},v_{i,l})}(t)]^{\alpha}.[\eta_{v_{i,l}}(t)]^{\beta}} \tag{8}$$

Also a different heuristic function is chosen, defined as the fraction of training cases that are correctly covered (described) by this term, as defined by (9):

$$\eta_{v_{i,j}}(t) = \frac{|T_{ij}\ \&\ CLASS = class_{\text{ant}}|}{|T_{ij}|} \tag{9}$$

Updating the pheromone trail of the environment of $\mathcal{MAX}$–$\mathcal{MIN}$ Ant Systems is accomplished in two phases: evaporation of all edges and reinforcement of the path of the best performing ant. The reinforcement of the best ant's path should be proportional to the quality of the path Q^+, which we define as the sum of the *coverage* and the *confidence* of the corresponding rule. Coverage measures the fraction of remaining (not yet covered by any

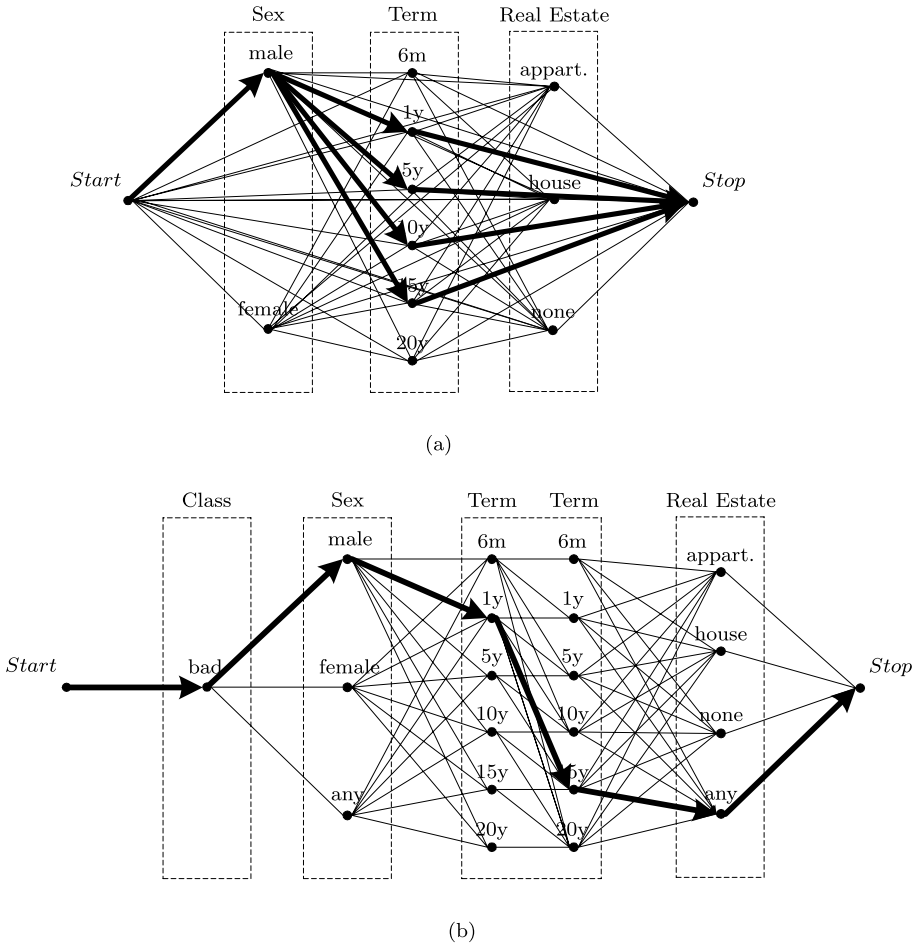

Fig. 4 Example construction graphs for AntMiner (**a**) and AntMiner+ (**b**) for a simplified credit scoring application

of the extracted rules) data points covered by the rule, that are correctly classified. The confidence gives an indication of the overall importance of the specific rule by measuring the number of correctly classified remaining data points over the total number of remaining data points. Note that there is a scaling of this quality measure in (10) to ensure that τ lies between 0 and 1. The evaporation rate is set at 0.15.

$$\tau_{(v_{i,j},v_{i+1,k})}(t+1) = (1-\rho)\cdot\tau_{(v_{i,j},v_{i+1,k})}(t) + \frac{Q^{+}_{\text{best}}}{10} \tag{10}$$

Rule extraction with ACO Instead of a typical rule induction approach, a rule extraction approach is taken by Özbakir et al. (2009) in their TACO-miner algorithm. Rule extraction techniques from neural networks or support vector machines extract rules that mimic these black box models as closely as possible. The aim is either to explain the non-linear model and as such open up the black box, or to improve the performance as noise can be removed

Algorithm 5 AntMiner+ algorithm for classification

```
TrainingSet = {all training data}
RuleList = {}
while not early stopping do
    Initialize probability to τ_max, pheromone and heuristic values according to (8) and (9)
    while not converged do
        Let ants run from source to sink
        Evaporate pheromone of all edges
        Prune rule of best ant R_best
        Add pheromone of R_best
        Adjust pheromone levels if outside boundaries [τ_min, τ_max]
        Kill ants
        Update probabilities of edges
    end while
    Add rule R_best to RuleList
    TrainingSet = TrainingSet \ {data instances covered by R_best}
end while
Evaluate performance on test set
```

in this manner (Martens et al. 2009). Özbakir et al. (2009) use touring ACO (Hiroyasu et al. 2000): after discretization each variable is encoded into a set of binary variables and rules are extracted for each class that maximize the value of the corresponding output node. Experimental work on six UCI datasets (Hettich and Bay 1996) demonstrates superior performance to C4.5 and PART. However, aside the limited experimental setup, it is not clear whether the improved performance comes from the specific ACO-based approach or from the rule extraction approach, as no experiments are conducted with other rule extraction techniques.

ACO for Bayesian network learning A Bayesian network is a graphical representation of the probabilistic relationships among a set of random variables. Building such a network requires two steps (Tan et al. 2006): (1) building a directed acyclic graph to encode the dependence relationships among the variables, and (2) estimating a probability table associating each node to its immediate parent nodes, on which it depends. When no prior knowledge is available to construct the network model, the network needs to be learnt from the data.

de Campos et al. (2002) use the Ant Colony System, proposed by Dorigo and Gambardella (1997), to learn the network structure. This is typically solved in a heuristic manner, which makes the ACO perspective a natural choice and is based on the B Algorithm (Buntine 1991). The B algorithm starts with an arc-less structure and adds that arc that provides the largest gain in terms of a scoring function f. Similarly, using the ACO metaheuristic, each ant starts with an empty graph and adds an arc between two variables at each step until the ants decide to add no more arcs. This ACO-based approach is supplemented with a local optimizer that adds, deletes or swaps arcs for each solution. Experiments on three datasets suggest that the quality of this ACO-B algorithm is better than the included heuristic algorithms: Hill Climbing, Iterated Local Search and Estimation of Distribution Algorithm. Although it performs not as fast, the authors claim the algorithm is still useful for large problems by using distributed computing. Further experiments investigating algorithm details, such as suitable evaluation functions and parameter settings, are considered issues for future research.

Many of the networks that are considered by the ants are equivalent to each other, in the sense that they produce the same score. This results in much redundancy, which can tackled by conducting the search within the space of equivalence classes of the network structures. The ACO-E algorithm proposed by Daly and Shen (2009) takes this approach and allows graphs that may contain both undirected and directed edges, but contains no directed cycles. Seven types of moves are allowed, as defined by (Chickering 2002) for the space of equivalence classes, while optimizing the BDeu scoring function. ACO-E is benchmarked on six datasets against three other techniques: GREEDY-E, which greedily searches in the space of equivalence classes of Bayesian network structures (Chickering 2002), EPQ, an evolutionary approach within the same space (Cotta and Muruzábal 2004), and ACO-B. The results show superior results as in all cases the BDeu score of ACO-E was better than the score of the other algorithms, and this with comparable computational complexity.

Continuous ACO for training artificial neural networks Recently, a continuous version of the ACO metaheuristic has been proposed, aimed at solving continuous optimization problems (Socha 2004; Pourtakdoust and Nobahari 2004; Socha and Blum 2007; Socha and Dorigo 2008). Instead of using a probability function that has one value per node (denoting a single value for the discrete variable), the $ACO_{\mathbb{R}}$ (Socha 2004) version employs a probability distribution consisting of several Gaussian probability density functions that can easily be sampled. This has been applied in a data mining setting for the training of weights in a neural network (Socha and Blum 2007). Based on experiments on three medical diagnosis datasets, the authors find that the hybrid version that combines the commonly used Levenberg-Marquardt algorithm (Bishop 1996) with $ACO_{\mathbb{R}}$ outperforms the backpropagation and Levenberg-Marquardt algorithms, as well as an included genetic algorithm classifier. Although the experimental setting is rather limited, the potential of $ACO_{\mathbb{R}}$ for data mining is promising and will be addressed in Sect. 5 on issues for future direction.

Clustering with ACO All previously discussed data mining techniques use swarm intelligence for classification. The ACO metaheuristic can also be applied to the clustering task. The clustering task is introduced in detail in Sect. 2.2.3 which discusses the use of ant-based sorting for clustering. The approach is quite similar to the use of ACO for the TSP problem, where paths are constructed by ants trying to minimize the total distance. Whereas for the TSP problem nodes corresponded to cities, in this clustering application each node represents a data instance and the distance between them is defined by their dissimilarity. An ant is not obliged to visit all cities: the number of data instances to visit can either be chosen constant (Dorigo et al. 2004), or decreasing over time in a simulated annealing fashion as done in the ACODF technique (Tsai et al. 2004). In this manner each ant will describe a path or series of similar data items, corresponding to the data items it has visited. A final post-processing step breaks up the ant's path into several subpaths, corresponding to the final data clusters. The suggested approach is to break ties between data instances that have a dissimilarity above a certain threshold. Although in Tsai et al. (2004) the results on three synthetic datasets indicate it performs better than SOMs combined with k-means and GKA (genetic k-means algorithm), no comparison with other ant-based approaches is given.

Note that most ant-based clustering techniques use a different approach however, based on sorting behavior rather than foraging behavior of ants, as will be discussed next in Sect. 2.2.

2.1.4 Challenges for ACO-based data mining

Challenges that remain for these ACO-based classification techniques are the need for discretization of all variables and the learning time.

Since ACO is designed to solve discrete optimization problems, each variable can only have a limited number of values. Although AntMiner+ already makes the explicit difference between nominal and ordinal variables, a continuous variable can not be included. This makes the choice of discretization technique quite important. In Otero et al. (2008, 2009) also interval rules are allowed, following the idea of AntMiner+, and a first attempt is made to determine the cutoff value in a dynamic manner for the continuous variables. By choosing a value minimizing the entropy of the corresponding partition, no discretization step is needed. This idea is incorporated in the AntMiner algorithm and is named cAntMiner. The benchmarking results on eight public datasets indicate an improved performance compared to the original AntMiner algorithm. However, no results of this approach for the subsequent AntMiner versions are reported. Note that recently continuous ACO versions have been proposed (see e.g. the work of Socha 2004) and these could provide a more suitable solution for this issue. Section 5, on future research directions, revisits this point.

The learning time of these ACO-based classification techniques is quite long compared to other rule-based techniques as C4.5 and RIPPER (Martens et al. 2007). Although it is stated that for most applications the training duration is not an issue and may last up to hours, for some real-time applications it is of key importance. In these cases explicit parallelization of the inherent distributed system might resolve this challenge. This will also be addressed in Sect. 5.

2.2 Ant-based sorting

2.2.1 Biological background

Whereas the ACO metaheuristic is based on the foraging principles of ants, clustering algorithms have been introduced that mimic the sorting behavior of ants. It has been shown that several ant species cluster dead ants in so-called cemeteries to clean up the nest (Bonabeau et al. 1999). Although the precise dynamics of this behavior is not yet fully understood, a simple model where ants randomly walk around, pick up dead ants and drop them based on local information only (density of other corpses) seems to explain the behavior to a large extent. Figure 5 shows the dynamics of such cemetery forming for the *Messor sancta* type of ant Bonabeau et al. (1999): in a matter of hours, randomly scattered dead ants are clustered in several cemeteries. This sorting behavior has also been observed in the clustering of larvae, with smaller larvae being placed near the center, and larger ones towards the edge of the cluster.

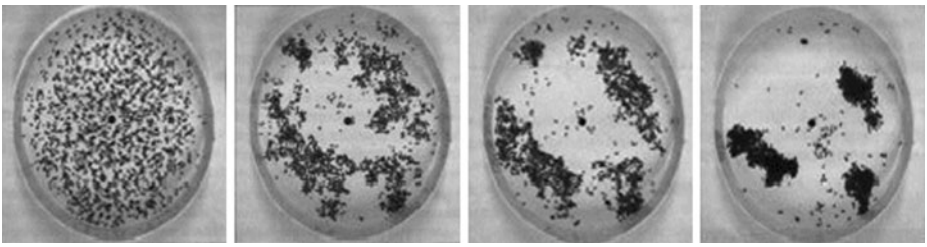

Fig. 5 Randomly scattered ant corpses are clustered in cemeteries in a matter of hours (taken with permission from the authors from Bonabeau et al. 1999). The different stages are reflected 0, 3, 6 and 36 hours after the beginning of the experiment

2.2.2 Clustering artificial ants

The first application that mimics the clustering and sorting behavior of biological ants is reported by Deneubourg et al. (1990) in a robotics context. Ants pick up and drop items based on the similarity with the surrounding items. If an ant carries an item, the more similar the items in the immediate surroundings, the higher the probability that the ant will drop the item. If the ant is not carrying any item, the more dissimilar an item is to its surrounding items, the more likely the ant will pick up that specific item. Ant-based clustering is one of the most popular data mining techniques inspired by swarm intelligence and is discussed in detail next.

2.2.3 Data mining implementation of ant-based sorting

Clustering, multi dimensional scaling and low dimensional mapping Clustering is an unsupervised data mining task whose goal is to group data into homogeneous groups, such that data within a cluster are as similar as possible and data in different clusters are as dissimilar as possible (Witten and Frank 2000). Clustering techniques can be categorized as being exclusive, overlapping, hierarchical or probabilistic. A commonly used clustering technique is k-means clustering, which provides exclusive clusters. Some techniques provide a probability for each instance to belong to one of the clusters. An example family of techniques for probabilistic clustering are mixture models. The expectation maximization (EM) method uses log-likelihood as the objective function to model the distributions. Hierarchical clustering provides clustering at several levels of granularity, and the outcome is typically represented in a tree-like structure. Agglomerative hierarchical clustering techniques work bottom up, in the sense that clusters are formed by recursively merging data points together, while divisive methods recursively divide data into finer groups. In order to merge or split subsets of data points, the distance between individual data points has to be extended to the distance between subsets. Such derived proximity measure is called a linkage metric, whereby popular ones are single link, average link and complete link, calculating the minimum, average and maximum distance metric between each pair of data points across two subsets. An extensive survey on clustering techniques can be found in Berkhin (2002).

Closely related to clustering is multi dimensional scaling, where the data are mapped onto a two- or three-dimensional map that can easily be visualized. The mapping should preserve the topological properties of the input space, such that data items that are near each other in the input space are also close in the low dimensional feature space. A well-known technique is a Self-Organizing Map (SOM), which is a special kind of artificial neural network (Kohonen 2001). A visualization of such a feature map can reveal straightforward clusters. However, if a final clustering solution is sought, an extra postprocessing step is needed, either human-based or automatic, that defines clusters based on the mapping.

LF algorithm The data clustering algorithm proposed by Lumer and Faieta (1994) is based on the same ant behavior as modeled by Deneubourg et al. (1990) and is explained in Algorithm 6. First, all data items are placed randomly throughout a two-dimensional grid. The ants are also placed randomly across the grid cells and can walk through the grid by moving to neighboring grid cells. If the ant carries a data item and the grid cell on which it is located is empty, the ant will drop the data item with a probability P_{drop} which increases if the surrounding data items are similar to the data item currently carried by the ants. Similarly, if the ant carries no data item and a data item is located on the ant's grid cell, it will be

Algorithm 6 LF algorithm for ant-based clustering

```
Randomly scatter data items on grid
Randomly place ants on grid
for t = 1 to max_iterations do
    a = random ant
    move ant a randomly over stepsize grid cells
    if ant a carries data item and ant a's grid cell is not occupied by a data item then
        i = data item carried by ant a
        if rand() ≤ P_drop(i) {(12)} then
            ant a drops data item i at current grid cell
        end if
    end if
    if a does not carry a data item and ant a's grid cell is occupied by a data item then
        i = data item at ant a's grid cell
        if rand() ≤ P_pickup(i) {(11)} then
            ant a picks up data item i
        end if
    end if
end for
```

picked up with a probability P_{pickup} which favors picking up data items that are dissimilar to its surrounding data items. As with all clustering algorithms, a neighborhood function $f(i)$ is needed based on the similarity between data item i and its surrounding data items $j \in$ neighborhood $N(i)$.[7] This function is defined by (13) and uses a dissimilarity/distance metric $d(i, j)$ (Lumer and Faieta 1994), which is typically the Euclidean distance for a m-dimensional dataset. This metric provides an outcome between 0 and 1, with the normalizing term s^2 measuring the total number of sites in the local neighborhood (s cells in vertical and horizontal direction) and $\alpha \in [0, 1]$ a parameter to be tuned to the dataset. The extreme case in which all neighboring items are the same, provides a $d(i, j) = 0$ for all j and hence a maximum value for the similarity between data item ii and its neighborhood: $f(i) = 1$. The probability to pick up the item is therefore very low. The other extreme where all neighboring items are maximally dissimilar provides a very small $f(i)$ and thus a high probability to pick up the data item.

$$P_{pickup}(i) = \left(\frac{k_1}{k_1 + f(i)}\right)^2 \tag{11}$$

$$P_{drop}(i) = \begin{cases} 2 \cdot f(i) & \text{if } f(i) < k_2 \\ 1 & \text{otherwise} \end{cases} \tag{12}$$

$$f(i) = \max\left(0, \frac{1}{s^2} \sum_{j \in N(i)} \left(1 - \frac{d(i, j)}{\alpha}\right)\right) \tag{13}$$

In their experiments, the following values were used: $k_1 = 0.1$, $k_2 = 0.15$, $\alpha = 0.5$, $s^2 = 9$, $max\_iterations = 10^6$. The reported experiments cluster a number of synthetic, low-dimensional datasets to demonstrate the functioning of their algorithm. As they observe that

[7] For brevity reasons, within this clustering context we denote instance $\mathbf{x_i}$ by its index i.

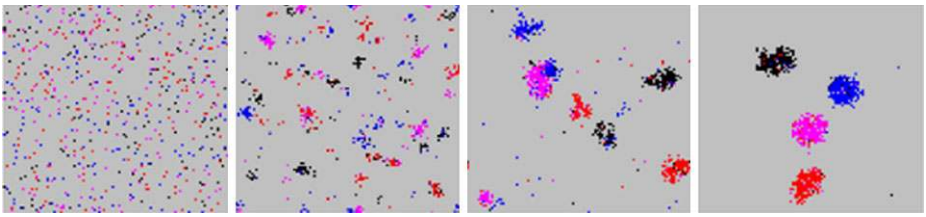

Fig. 6 Different stages of ant-based clustering of data items (taken with permission from the authors from Handl and Meyer (2002)).

typically more clusters are formed than are initially present in the dataset, three modifications are made. Firstly, a speed characteristic for each ant: some ants are able to move over several grid units at a time, forming coarse clusters, while some ants moving more slowly will place the data items with more accuracy. Secondly, each ant has a short term memory, remembering the location and characteristics of the most recently picked up data items (set to 8 in their experiments). Each newly picked up data item is compared to those in memory and instead of moving in a random direction, the ant moves towards the location of the most similar data item in its memory. As such, it is less likely that similar clusters are formed independently. Finally, a self-annealing like mechanism allows ants to destroy existing clusters if no action is performed on them for a given number of steps.

Additions to this basic LF algorithm that have been proposed are an adaptive setting of the parameter (Handl et al. 2006), allowing multiple data items to be transported at once (Li et al. 2005), including pheromones to speed up the process (Ngenkaew et al. 2008) and using fuzzy rules instead of pick up and drop probabilities (Schockaert et al. 2004). Additionally, some parameter suggestions are made as a function of the dataset (Handl et al. 2006): a grid of size $M \times M$ with M set at ceil($\sqrt{10n}$), a step size of $\sqrt{20n}$ and a suggested $2000 \cdot n$ iterations.

A first real-life application entails document clustering, where documents are classified in the form of a topic map, where semantically similar documents appear close to each other on the map (Handl and Meyer 2002). The emerging clusters on the two-dimensional grid at the different stages in that application can clearly be seen in Fig. 6. Further, these ideas have also been proposed for graph partitioning (Kuntz et al. 1997), with applications in VLSI (Very Large Scale Integration) and CAD (Computer Aided Design) technology. However, no quantitative comparison has been provided and the method seems to work only if clusters are present in the graph.

The visualization of the data is quite similar to multi dimensional scaling (MDS) algorithms which map a high dimensional dataset to a low dimensional space so as to facilitate visualization. However, the location of the different clusters constructed by the LF algorithm is arbitrary.

ATTA: an improved data clustering algorithm An improved version of the LF algorithm has been proposed, named ATTA (Adaptive Time-dependent Transporter Ants), which given its reported results and rather extensive changes is worth discussing in more detail (Handl et al. 2006). ATTA has two variants: one which is limited to a topographic mapping, named ATTA-M, and one which actually results in clusters of data (ATTA-C). The main benefits of the proposed ATTA-C technique is an explicit partitioning without need of human intervention and an a priori setting of parameters depending on dataset characteristics. Additionally, the neighborhood function f is adapted to penalize high dissimilarities more heavily: similarity function f is defined by (14), similarly to (13). However, when the distance between

data item i and one of its neighbors j is larger than α, the second condition holds and the similarity is set to 0.

$$f(i) = \begin{cases} \max(0, \frac{1}{s^2} \sum_{j \in N(i)} (1 - \frac{d(i,j)}{\alpha})) & \text{if } \forall j (1 - \frac{d(i,j)}{\alpha}) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Also, the radius of perception of each ant increases over time (linearly from 1 to 5) to save time at the early stages and to allow for the quick formation of clusters in this initial phase of the clustering algorithm, while allowing larger neighborhoods to be investigated later on and as such improve overall clustering quality. Finally, the threshold functions $P_{pickup}(i)$ and $P_{drop}(i)$, as shown by (15) and (16) are also modified to speed up the clustering process. These formulas are derived empirically, where ants will always pick up a data item if the similarity function is less than one, and drop a data item if the similarity function exceeds one.

$$P_{pickup}(i) = \begin{cases} 1 & \text{if } f(i) \leq 1.0 \\ \frac{1}{f(i)^2} & \text{otherwise} \end{cases} \tag{15}$$

$$P_{drop}(i) = \begin{cases} 1 & \text{if } f(i) \geq 1.0 \\ f(i)^4 & \text{otherwise} \end{cases} \tag{16}$$

Experiments are conducted on seven UCI datasets and several synthetic datasets where ATTA is compared with k-means clustering, one-dimensional SOMs, average link agglomerative clustering and a Gap statistic based technique. The results demonstrate good clustering quality, independent of the degree of overlap or size of the data, and the ability to automatically detect the number of clusters inherently present in the dataset. The topographic mapping results are however not so good. The authors state that other techniques, such as multi dimensional scaling and two-dimensional SOMs, are more suitable for this task, with the seemingly only benefit of the ant-based approach being the linear scaling.

A^4C: Ant clustering based on cellular automata A different approach is taken in the A^4C algorithm (Xu and He 2007), in the sense that each ant represents a data instance and uses a cellular automata approach. Cellular automata were originally introduced by von Neumann (1966) and consist of identical and independent cells, arranged regularly in a grid. Each cell has a number of states it can be in, and evolves in discrete time steps according to a predefined set of rules.

The grid of cells in which ants walk around is the same as in the LF algorithm. In their Ants Sleeping Model (ASM) each ant is in either an active or sleeping state. They base themselves on the observation that in nature ants will gather in groups that all have similar characteristics and repel ants that are different. Based on a fitness function that measures the similarity with neighboring ants, ants can transition from one state into the other with a certain probability. If an ant is sleeping and the ant's fitness is low, it feels less secure and has a high probability of becoming active, looking for another location. Once it has found such a location, with high fitness and hence surrounded by similar ants, the probability to go to a sleeping state increases again. An explicit clustering is achieved by the introduction of a clustering rule.

The workings of this Adaptive Artificial Ant Clustering Algorithm (A^4C) are described in Algorithm 7. Similarly to the LF algorithm, ants are scattered throughout the two-dimensional grid, but now each ant represents a data instance. Each ant computes its fitness and based on that determines its probability to change state P_a.

Algorithm 7 A^{4}C algorithm for ant-based clustering

```
Initialize parameters
Randomly place ants on grid
while not termination do
  for each ant do
    compute the ant's fitness and active probability P_a
    if rand() ≤ P_a then {(18)}
      activate ant and move to random, unoccupied neighbor cell
    else
      stay at current cell and remain resting
    end if
    update cluster label: if ant is sleeping, class of majority neighbors
    update cluster label: if ant is active, cluster label remains
  end for
  adaptively update parameters α, β, λ
end while
```

The visible neighborhood of an ant is of size s_X (on the horizontal axis) times s_Y (on the vertical axis). Hence, on the horizontal X dimension, an ant has s_X neighboring cells to its right and s_X neighboring cells to its left (its current cell constitutes the final member of the neighborhood). Similarly for the vertical dimension, an ant has $(2s_Y + 1)$ neighboring cells. Since they opt for a 8-neighbor model (where also diagonal neighbors are considered, next to top, down, left, right), each ant has a total of $(2s_X + 1) \cdot (2s_Y + 1)$ neighboring cells.

The fitness of an ant measures the extent to which the ant fits into its current living environment, and is defined by (17). Note the similarity with the neighborhood function, defined by (13), though now a larger neighborhood size is possible and α as well as λ are adaptively changed.

$$f(i) = \max\left(0, \frac{1}{(2s_X + 1) \cdot (2s_Y + 1)} \sum_{j \in N(i)} \left(1 - \frac{d(i, j)}{\alpha_i}\right)\right) \tag{17}$$

$$P_a(i) = \frac{\beta^\lambda}{\beta^\lambda + f(i)^\lambda} \tag{18}$$

Xu and He (2007) report their clustering results on the two-dimensional synthetic dataset consisting of four Gaussian distributions with little overlap, also used by Lumer and Faieta (1994), and the Iris, Wine, Glass, Thyroid and Soybean UCI datasets, which have respectively 150, 178, 214, 215 and 47 data instances. The grid size is chosen as in the LF algorithm experiments, being a grid of size 100×100.

Their experiments indicate that the A^{4}C algorithm provides clusters of higher quality than those provided by the LF algorithm and also at a far less computational cost. Even with the parameters not changing dynamically but set fixed, A^{4}C performs better than LF. The computational expense of LF is due to three things: (1) ants take a long time to find proper locations to deposit data instances, definitely in the later stages, (2) assigning isolated data instances to a suitable cluster can take a long time and such data instances may even never be dropped, and finally (3) the parameters of LF are hard to set and are not set to change adaptively. The change in paradigm by representing data instances with ants circumvents these issues, as no data instances have to be looked for and most of the computational time is simply spent on the poorly adapted ants.

Benchmarking ant-based clustering techniques A major methodological gap in clustering research is the absence of proper and rigorous analysis of the clustering quality. One of the major reasons is that most ant-based clustering techniques provide a spatial distribution of the data, and not an explicit clustering of the data, as pointed out by Handl et al. (2003). Although the clustering solution is rather obvious from a human perspective, the automatic clustering is not that straightforward. This analysis issue is addressed by Handl et al. (2003) where ant-based clustering is augmented with an additional explicit clustering step, and benchmarked with commonly used clustering techniques such as k-means clustering and agglomerative average link clustering. For these techniques, a priori the correct number of clusters is assumed (which is not the case for the ant-based clustering technique). The clustering results on three artificial and three UCI datasets are evaluated using the F-measure, Rand Index, Inner Cluster Variance and Dunn Index. Note that the ATTA technique is developed by the same authors and is based on this benchmarking study.

The benchmarking experiments show that ant-based clustering performs very well compared to the other techniques: for the synthetic datasets it always ranks second, and achieves results that are only slightly less than the results of k-means clustering. On all but one real datasets, ant-based clustering even achieves the best results for all performance metrics, while not having information on the optimal number of clusters (unlike for the other two techniques). For the Yeast dataset, the results are less conclusive, with all three techniques achieving the best result for some evaluation metric, though the results are all quite poor. With respect to necessary runtime, although ant-based clustering is slower on the small datasets, since it scales linearly with the number of data instances, it outperforms the other techniques on the larger datasets (from around 2000 data instances).

The extensive benchmarking studies by Handl et al. (2003, 2006) demonstrate the following advantages of ant-based clustering:

- It scales linearly, like k-means clustering, but unlike agglomerative hierarchical clustering, which scales quadratically.
- It is fairly robust to outliers in the data.
- It is applicable to any kind of data that can be described in symmetric dissimilarity.
- It makes no assumption about cluster shape, unlike k-means which works well only for spherical clusters.
- It can automatically determine the number of clusters, unlike k-means and agglomerative hierarchical clustering.

To conclude, ant-based clustering is very promising and able to deal with important clustering aspects, such as finding the optimal number of clusters and dealing with different types of data. However, several challenges still remain.

2.2.4 Challenges for ant-based clustering techniques

Although these results seem promising, a large scale benchmarking study with high dimensional data, real-life problems and other traditionally used clustering techniques as well as ant-based clustering techniques is missing. For example the A^4C clustering technique has only been applied to one synthetic data and four (small) UCI datasets, with performance measured using the class labels provided in the dataset. Benchmarking the main ant-based techniques (i.e. LF, ATTA, A^4C and ACODF) with each other and popular clustering techniques would reveal which technique and paradigm works best for clustering. Also, we

would like to point out that making the implementations available would facilitate such experiments to a great extent. As the results seem very promising and worthwhile, an implementation in the Weka workbench (Witten and Frank 2000) would definitely add to the visibility and use of this research.

3 Particle swarm optimization

3.1 Biological background

Flocks of birds and schools of fish demonstrate fascinating coordinated collective behavior. Birds randomly look for food, while also keeping an eye on one another and following the one closest to food. Flocking has several advantages for the individuals: increased effectiveness in searching for food, better predator avoidance and evasion, and improved mating opportunities. With each individual using only local information, the overall swarm exhibits a fluid coherent motion which seems perfectly synchronized (Kennedy and Eberhart 1995). Reynolds (1987) shows that such flock-like motion can be modeled with three basic behavioral principles that use only information from neighbors: (1) avoid collisions with nearby flockmates, (2) match velocity with nearby flockmates, and (3) attempt to stay close to nearby flockmates.

3.2 Particle swarm optimization

In Particle Swarm Optimization (PSO) the birds are represented by a population of particles, called a swarm (Kennedy and Eberhart 1995). Although PSO was initially designed to graphically simulate the choreography of bird flocking, its conceptual development finally led to a continuous optimization meta-heuristic. Whereas ACO is designed to solve discrete optimization problems, PSO is designed to solve continuous ones. Each particle has a certain location and velocity within the search space and as such represents a solution. Like birds wandering through their environment looking for food or evading predators, particles fly through the search space in search of high quality solutions. This search is guided by the best location it has found so far and the best location found by all neighboring particles.

The basic PSO algorithm is described in Algorithm 8. Initially the particles are scattered randomly throughout the input space. Each particle stores the personal best location it has ever visited: $\mathbf{p}_i$, with fitness value $pbest_i$. Additionally, the best of all these personal best solutions of all particles in its neighborhood $\mathbf{p}_g$ is also retained, with fitness value $pbest_{global}$: such that $pbest_{global} = f(\mathbf{p}_g) \geq f(p_j), \forall j \in N(\mathbf{x}_i)$. Usually the neighborhood is chosen as the complete space, in which case the local best corresponds to the global best solution. PSO algorithms update the particles' location and velocity according to (19) and (23), where $\mathbf{U}(0, \phi_1)$ and $\mathbf{U}(0, \phi_2)$ represent vectors of random numbers which are uniformly distributed in $[0, \phi_1]$ and $[0, \phi_2]$ respectively. At each iteration, these numbers are generated randomly for each particle. As such, particles move stochastically in the direction of their own best solution and the neighborhood's best solution. Supposing that the random numbers are 1, the update of a particle's location and velocity is illustrated in Fig. 7.

Note that in this original PSO algorithm, the velocity of each particle is kept within the range $[-V_{\max}, +V_{\max}]$ to avoid these numbers growing to extremely large values. Further updates of the particles are typically halted when sufficiently good performance is obtained

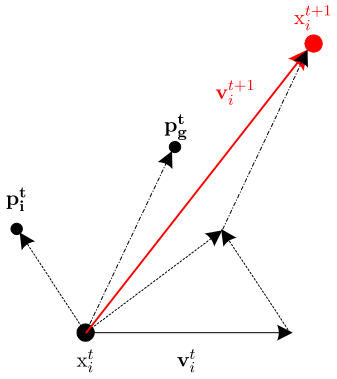

Fig. 7 Illustration of PSO update of a particle's location and velocity

or when a maximum number of iterations has occurred.

$$\text{PSO:} \quad \mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \mathbf{U}(0, \Phi_1)^t \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + \mathbf{U}(0, \Phi_2)^t \cdot (\mathbf{p}_g^t - \mathbf{x}_i^t) \tag{19}$$

$$\text{LDPSO:} \quad \mathbf{v}_i^{t+1} = \mathbf{w}^t \cdot \mathbf{v}_i^t + \mathbf{U}(0, \Phi_1)^t \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + \mathbf{U}(0, \Phi_2)^t \cdot (\mathbf{p}_g^t - \mathbf{x}_i^t) \tag{20}$$

$$\text{CPSO:} \quad \mathbf{v}_i^{t+1} = \chi \cdot \left(\mathbf{v}_i^t + \mathbf{U}(0, \Phi_1)^t \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + \mathbf{U}(0, \Phi_2)^t \cdot (\mathbf{p}_g^t - \mathbf{x}_i^t)\right) \tag{21}$$

$$\text{CLPSO:} \quad \mathbf{v}_i^{t+1} = \mathbf{w}^t \cdot \mathbf{v}_i^t + \mathbf{U}(0, \Phi_1)^t \cdot (\mathbf{C}\mathbf{p}_i^t - \mathbf{x}_i^t) \tag{22}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{23}$$

Algorithm 8 Particle Swarm Optimization

```
Initialize a population of particles with random positions and velocities, throughout the input space
while not sufficiently good performance or maximum number of iterations do
  for each particle i do
    f_p = f(x_i)
    if f_p > pbest_i then
      pbest_i = f_p
      p_i = x_i
    end if
    g = {j | f(p_j) = max(f(p_k), k ∈ N(x_i))}
    Update velocity v_i according to (19)
    Update velocity x_i according to (23)
  end for
end while
```

Since the introduction of PSO, several variants have been introduced (see e.g. Montes de Oca et al. 2009), each defining different velocity update rules (19) to (20). A discrete PSO variant (DPSO) is proposed by Kennedy and Eberhart (1997) where a particle's position is discrete but its velocity is continuous. The linearly decreasing PSO (LDPSO) is introduced by Shi and Eberhart (1998), where a time-decreasing function is added as a multiplicator in

the first term of the velocity update rule. The constricted PSO (CPSO) adds a constriction factor in the velocity update rule to avoid unlimited growth of the particles' velocity (Clerc and Kennedy 2002). Zheng et al. (2003) show that in some cases better results are obtained with a time-increasing function. The fully informed particle swarm (FIPS) is introduced by Mendes et al. (2004) in which information from all the neighbors is used, not only the best one. Ratnaweera et al. (2004) added local search behavior by removing the first (inertia) term from the velocity update rule, reinitializing the velocity of a particle to a large value when its velocity approaches zero, and adapting the acceleration coefficients linearly. In the adaptive hierarchical PSO, the topology of the neighborhood is changed at runtime (Janson and Middendorf 2005). Comprehensive learning PSO (CLPSO) combines all particles' historical best information to update a particle's velocity (Liang et al. 2006). This strategy is aimed at avoiding premature convergence by enabling the diversity of the swarm to be preserved. The combined best position of the particles, denoted as $\mathbf{Cp}_i^t$, is chosen probabilistically for each dimension based on the fitness function. The chaotic PSO uses a chaotic number generator each time a random number is needed by the classical PSO algorithm, in order to improve global searching capability by escaping the local solutions (Alatas et al. 2009). Based on an empirical analysis of several PSO variants, a new algorithm is proposed in Montes de Oca et al. (2009) that combines the algorithmic components that show distinct advantages in performance. The resulting PSO algorithm is named Frankenstein's PSO and includes both the FIPS mechanism for updating a particle's velocity, as well as a decreasing inertia weight. For more details on PSO, its variants and applications, we refer to a recent overview in Poli et al. (2007).

3.3 Popular data mining applications of PSO

PSO for rule-based classification models A first application of PSO to classification is reported by Sousa et al. (2004). A sequential covering algorithm is employed, with each rule being a conjunction of terms. Rule quality is measured as the product of sensitivity and specificity, just as with AntMiner. When the quality of a rule remains constant for a number of iterations, set to 30, the swarm is considered to have converged and the best solution (rule) found by the swarm is added to the rule list. Once a rule is discovered, pruning removes any attribute that does not contribute to the performance. Training is stopped when a user-defined percentage of the data has been covered by the rules, typically set to 90% in their work. The final default rule predicts the remaining majority class.

Three PSO variants have been tested: discrete PSO, linear decreasing weight PSO and constricted PSO. All variables are discretized into binary variables: for each variable as many binary variables are created as needed to encode the possible values ceil($\log_2$(1 + number of values)), and an extra bit is added to denote the indifference state (similar to the dummy value in AntMiner+). The value of the binary variable is 1 if the corresponding variable has the specific value encoded by that binary variable. In a second phase, interval rules are allowed by creating multiple binary variables at once.

In their experiments, the constriction coefficient χ is set to 0.73, and ϕ_1, ϕ_2 are set to 2.05. The developed technique is benchmarked with C4.5 on three UCI datasets. On two datasets, a superior performance is reported in terms of test accuracy, though no significance testing is performed. For all three datasets, fewer or as many rules as C4.5 are obtained and hence an improved comprehensibility is provided. A drawback however is the increase in execution time.

In Holden and Freitas (2008), a combination of both ACO and PSO is used to build rule-based classification models that can handle both nominal and continuous variables.[8] While for PSO implementations nominal variables need to be encoded in binary variables, and ACO implementations require discretization, the proposed PSO/ACO2 technique directly deals with both types of variables. This technique extends and refines a first version by the same authors that was proposed in Holden and Freitas, Holden and Freitas (2005, 2007).

A sequential covering approach is also taken, where a rule is first induced using nominal variables only, which serves as the basis for adding terms with continuous variables. As in the technique by Sousa et al. (2004), the constricted PSO is used with the constriction coefficient set to 0.73 and ϕ_1, ϕ_2 set to 2.05. Two dimensions are considered for each continuous variable: one for the lower bound and one for the upper bound. The quality measure, taken as sensitivity $\times$ specificity, seems to generate rules with sometimes very few covered data instances. Therefore the Laplace corrected precision evaluation metric is used for rule quality assessment. The same rule pruning and stopping criteria as in AntMiner are used. The induced rules are ordered according to their quality, which can be different to the order in which they were induced from the data.

Holden and Freitas (2008) compared their PSO/ACO2 technique to the PART algorithm on 27 UCI datasets. The results indicate that the performance of the proposed technique is competitive in terms of accuracy, but far more comprehensible as the provided rule sets are much smaller.

In Alatas and Akin (2009), rule mining is considered as a multi-objective optimization problem with predictive accuracy and comprehensibility objectives. Each rule is evaluated according to multiple rule evaluation criteria: accuracy and comprehensibility. The chaotic PSO algorithm searches for Pareto optimal classification rules.

Each particle is represented as a concatenation of real-valued elements in the range [0, 1]. The size of a particle is $2 \cdot m$: for each variable an attribute-existence part indicates whether the variable occurs in the rule (if its value is larger than 0.5), and if so, the attribute-value part encodes the value. Note that the encoding scheme makes a distinction between nominal and ordinal types of variables.

The accuracy is measured as the number of data instances that satisfy both the rule antecedent and consequent—0.5, divided by the number of examples satisfying all the conditions in the antecedent. Comprehensibility is estimated as one minus the relative number of variables that occur in the rule. Weights for accuracy and comprehensibility are set at 0.7 and 0.3 respectively. The minimal and maximal velocity are set to 0 and 1, learning rates ϕ_1 and ϕ_2 are set to 2. Experiments on four UCI datasets (mushroom, zoo, monk and nursery) show that the algorithm is competitive with the included algorithms Ridor, OneR, Prism, NNge and PART while also demonstrating good comprehensibility.

Although the multi-objective approach is surely interesting from a research perspective (see also Coello Coello et al. (2009)), no comparison is made with other PSO rule learners, nor are any experiments done with single-objective approaches (followed by all other proposed techniques). Since the induced rule sets from other PSO learners typically don't have an abundant number of variables in the rule antecedent, the additional value of the multi-objective approach seems limited.

Nearest-neighbor like classification A second application of PSO to classification is developed by De Falco et al. (2007), with a different model representation and with extended benchmarking experiments.

[8] A publicly available implementation of this technique is available at http://sourceforge.net/projects/psoaco2.

Considering a classification problem with c classes and m variables, De Falco et al. (2007) see the classification problem as finding the optimal coordinates of c centroids in the m-dimensional search space. Hence each particle has $2 \cdot c \cdot m$ components, encoding for each class the m-dimensional centroid location and velocity. Given a solution, a data instance is assigned to the centroid that is nearest. The linearly decreasing weight PSO variant is used in their implementation.

Experiments are conducted on 13 UCI datasets, where the proposed technique is compared to 9 popular classification techniques, including ANN, Bayesian Networks and Bagging. The PSO based classification technique is quite competitive, yielding the best results on three datasets, and obtaining an average ranking of four, while having an execution time in the same order of magnitude as the other included techniques. It seems that the technique is particularly suitable for binary classification problems but less for multiclass problems. Finally, it should be noted that the model is a nearest neighbor like model, and as such is surely less comprehensible than rule-based models.

PSO as optimizer within other learning algorithms PSO is used by Holden and Freitas (2009) to learn hierarchical classification models, specifically aimed at bioinformatics classification problems. The technique generates hierarchical ensembles of hierarchical rule sets (HEHRS), where the weights for combining the rule sets are optimized with PSO. This kind of classification model is specifically suitable for protein function classification and is applied to six datasets from that domain. Of the included four hierarchical classification techniques, the PSO optimized technique performs best.

PSO has been used to train both ANN (Kennedy and Eberhart 1995) and SVM models (Paquet and Engelbrecht 2003). In Samanta and Nataraj (2009), this is combined with PSO-based feature selection.

Clustering with PSO van der Merwe and Engelbrecht (2003) applied PSO for data clustering. Each solution is represented by the coordinates of a user pre-defined number of cluster centroids. Each data instance is assigned to the nearest cluster centroid, using Euclidean distance. The fitness function used is the quantization error, which can be seen as the average distance from a data point to its cluster centroid, averaged over the different clusters. Equation 24 formally defines the quantization error, for a clustering problem with k clusters, n data instances, u_{li} a binary variable which is 1 if instance i is assigned to cluster l, $\mathbf{x}_i$ denoting instance i, $\mathbf{z}_l$ the centroid of cluster l and $d()$ the Euclidean distance.

The computational steps are described in Algorithm 9. The algorithm is further refined using k-means to provide initial cluster centroids.

$$f = \frac{\sum_{l=1}^{k} \left(\sum_{i=1}^{n} u_{li} \cdot d(\mathbf{x}_i, \mathbf{z}_l) \right) / |C_l|}{k} \tag{24}$$

Using two artificial and four UCI datasets, the authors demonstrate the potential in terms of improved quantization error. A drawback is that, as with k-means clustering, the number of clusters needs to be predefined. The experiments are limited to low dimensional data, an issue that is addressed in the following PSO based technique.

Clustering high dimensional data with soft projected clustering Clustering high dimensional data is addressed with PSO by Lu et al. (2010). Due to the curse of dimensionality, traditional data clustering algorithms typically fail at providing good quality solutions. In soft projected clustering, a weight vector is assigned to each cluster, which denotes the relevance of the dimensions to that cluster (Domeniconi et al. 2007). The objective function

Algorithm 9 PSO-based clustering

```
Initialize each particle with k randomly selected cluster centroids
for t = 1 to max_iterations do
  for each particle do
    for i = 1 to n do
      for cluster l = 1 to k do
        if d(x_i, z_l) = min_{c=1:k}{d(x_i, z_c)} then
          data instance i is assigned to cluster l: u_li = 1
        else
          u_li = 0
        end if
      end for
    end for
  end for
  Update personal and global best positions
  Update particle's positions and velocities
end for
```

to minimize is the sum of the within-cluster distances of each cluster, weighted across the dimensions. So if a variable contributes strongly to the identification of a data instance in a cluster, the weight for that dimension for that cluster should be high. Such clustering algorithms are driven by the objective function and the search strategy used to minimize it. The proposed PSOVW algorithm (PSO for variable weighting) uses the CLPSO variant to optimize the objective function from (25), with x_{ij} the value of instance i in dimension j and z_{lj} the centroid of cluster l in dimension j (the other symbols are as before). The β parameter is user defined and set to 8 by the authors, making the function more sensitive to large weights, allowing them to play a stronger role.

$$f(W) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{li} \cdot \left(\frac{w_{lj}}{\sum_{q=1}^{m} w_{lq}} \right)^{\beta} \cdot d(x_{ij}, z_{lj}) \tag{25}$$

$$\text{such that} \quad \begin{cases} 0 \leq w_{ij} \leq 1 \\ \sum_{l=1}^{k} u_{il} = 1 \quad u_{il} \in \{0, 1\}, 1 \leq i \leq n \end{cases} \tag{26}$$

The objective function is a generalization of the ones used in previous soft projected clustering, such as LAC, W-k-means, EWKM and even k-means. The normalization of the weights allows the equality constraint, stating that the weights should sum to one, to be transformed into a less complicated boundary constraint.

A particle's location is represented by the $k \times m$ weight matrix W, the weight vectors for each of the clusters. Data instances are assigned to the nearest cluster centroid, taking into account the weights. The particle swarm wanders through the search space using the PSO principles, attempting to optimize (26).

Experiments are conducted on 36 synthetic datasets, with different dimensions (100, 1000 and 2000), and different data and cluster subspace overlap. Compared to the LAC, W-k-means, EWKM and k-means algorithms, PSOVW performs best on all 36 datasets both in clustering quality and cluster stability. Further experiments show that the improved performance is related to both the PSO search strategy, which unlike the other local search techniques is not easily trapped in a local minimum, and the new objective function. Empirical evaluation on three UCI datasets corroborate these findings. PSOVW is however more

time consuming, though still acceptable: for the dataset with 2000 dimensions, the algorithm takes about 5 minutes.

The algorithm is extended for text clustering by using the Jaccard coefficient instead of Euclidean distance (and corresponding revised objective function), and a document representation in term space. Using four datasets from 20 newsgroups (from UCI data repository), and four large text datasets from the CLUTO clustering toolkit, a superior clustering performance is also shown using the F-measure and Entropy.

As the technique is a k-means based technique, once again the number of clusters needs to be defined beforehand. An automated determination of k is considered an issue for future research.

3.4 Challenges

Current research for PSO-based classification aims at a wide variety of model types: rule-based models, nearest neighbor classifiers and black box SVMs, and this even for different kind of problems: binary to multiclass problems and even hierarchical classification. This is probably driven by the larger potential of optimization problems in the continuous domain. However, for both classification and clustering, a systematic investigation of the reasons for the (improved) performance is largely missing. Since the use of PSO requires many choices, the contribution of each needs to be studied. For example, if better performance is achieved, is it because of the use of the PSO metaheuristic, specific PSO variant chosen, objective function or data characteristics. We will come back to this issue in Sect. 5.

Since recently continuous ACO has been proposed to solve continuous optimization problems, it is worthwhile investigating when either of the metaheuristics is to be preferred.

4 Prey model

4.1 Biological background

In animal life we typically observe a biological environment consisting of foragers and preys. Upon encountering a prey, a forager needs to decide whether to attack the prey or to continue searching. According to the prey model of (Stephens and Krebs 1986), a forager compares the potential energy gain to the potential opportunity of finding an item of superior type. By obtaining the necessary energy to survive, extra time becomes available to perform activities as fighting and reproducing.

4.2 Foraging agents maximizing energy intake

In the foraging prey model, the average energy gain can be mathematically expressed in terms of the expected time, energy intake, encounter rate and attack probability for each type of prey. By maximizing this energy intake with respect to the probability to attack, a forager should either attack a prey type each time it is encountered, or never at all. The prey types to actually attack are the j top item types, ranked according to the energy intake per time unit to process, such that when a forager encounters an item of type $j+1$ or inferior, it benefits more from searching for items of types 1 through j than to process the item. For a formal derivation and details, we refer to Stephens and Krebs (1986) and Giraldo et al. (2010).

4.3 Dimensionality reduction for visual exploration

The prey model is used for dimensionality reduction in Giraldo et al. (2010). The data items are mapped to a two-dimensional grid such that clusters in the input space become visible in this two-dimensional grid. As with the LF algorithm discussed earlier, the data items are scattered randomly throughout the grid, of size $M \times M$ with M set at ceil $(\sqrt{10n})$, as suggested by Handl et al. (2006) for the ant-based clustering grid. The agents (three in this case) are placed on occupied grid positions. They can decide to either pick up a data item and replace it or to continue to another occupied position. Once the forager has decided to pick up/process a data item/prey, it moves to a randomly chosen free position and decides whether to drop it there or move to another free position. The decision to pick up and drop an item are related to the similarity between the considered data item and neighboring data items, where a squared neighborhood is chosen that contains a fixed set of 9 neighbors. The similarity function is a weighted function dependent on 1—the Euclidean distance, weighted such that items nearby receive higher weights than those further away.

There are a user-defined number of types of data items, set to 100 by the authors. Each type has the same energy intake, with data item types ranked from highest energy intake to lowest. In pick up mode, higher energy gains are obtained for data instances with low average similarity. Hence, the type number is an integer, low for low similarities, and high for high similarities. Similarly for foragers in the drop mode.

A first visual evaluation is performed on three and two-dimensional datasets, showing that the resulting clusters are indeed well-separated. The results are compared to the ISOMAP (Tenenbaum and Silva 2000), t-SNE (van der Maaten and Hinton 2008) and the previously discussed ant-based LF algorithms. Real-life datasets include two UCI datasets (iris and wine) and the 4096 dimensional Olivetti face dataset. The complete-link algorithm is used to partition and label the two-dimensional data instances (King 1967). For the iris and Olivetti datasets, the prey model performs best in terms of clustering accuracy. For the wine dataset, the prey model comes second, after t-SNE.

4.4 Challenges for prey model based dimensionality reduction

The main challenge for prey models within data mining comes from its novelty, as very limited research has been conducted on the topic. The methodology employed by the prey model for data organizing knows many similarities with the ant-based sorting techniques. As for ant-based sorting, a large scale benchmarking study with high dimensional data, real-life problems and other traditionally used clustering techniques is currently missing. A further investigation of the (dis)similarities between ant-based sorting and prey models for data clustering, and the optimal conditions to use either one of them, is currently lacking.

5 Directions for future research

Tables 1 and 2 provide an overview of the ant and PSO based data mining techniques. For each of the techniques, we list the metaheuristic variant that was used, the objective function to optimize, the type of the resulting data mining model, the solution component representation, the techniques which are used as benchmark, the datasets used in the benchmarking experiments, and the approach from our unifying framework that is taken (effective search versus data sorting). Note that for the artificial (arti) datasets, we also list the dimension (m) and number of data instances (n). The techniques are ordered according to the model output type and year.

Table 1 Literature overview table of data mining techniques based on ACO and ant-based sorting

	Authors and year	Metaheuristic variant	Objective function	Model type	Solution component representation	Benchmarked techniques	Datasets	Approach
AntMiner	Parpinelli et al. (2002)	AS	sens × spec	Rule list	Path ant: rule	CN2	UCI (6): bcw, chd, derm, hep, lbc, ttt	ES
AntMiner2	Liu et al. (2002)	AS	sens × spec	Rule list	Path ant: rule	AntMiner	UCI (1): bcw	ES
AntMiner3	Liu et al. (2003)	ACS	sens × spec	Rule list	Path ant: rule	AntMiner	UCI (2): bcw, ttt	ES
AntMiner+	Martens et al. (2007)	MMAS	cov + conf	Rule list	Path ant: rule	AntMiner, AntMiner2, AntMiner3, RIPPER, C4.5, 1NN, logit, SVM	UCI (11): aus, bal, bcw, car, cmc, ger, iris, lbc, tae, ttt, wine arti (1): replay ($m = 2$, $n = 1000$)	ES
cAntMiner	Otero et al. (2008) Otero et al. (2009)	AS	sens × spec	Rule list	Path ant: rule	AntMiner C4.5	UCI (8): aus, bcw, chd, crx, glass, hep, ion, wine,	ES
TACO-miner	Özbakir et al. (2009)	TACO	ANN output value	Rule list	Path ant: rule	NBTree, Decision Table, PART, C4.5	UCI (6): bcw, ecg, iris, lbc, nurs, pima	ES
ACO-B	de Campos et al. (2002)	ACS	decomposition scoring function	Bayesian network	Path ant: network	ILS, UMDA, PBIL, HCST	alarm, insurance, boblo	ES
ACO-E	Daly and Shen (2009)	ACS	BDeu scoring function	Bayesian network	Path ant: network	ACO-B, EPQ, GREEDY-E	alarm, barley, diabetes, hailfinder, mildew, win95pts	ES
$ACO_{\mathbb{R}}$ *LM* $ACO_{\mathbb{R}}$ *BP*	Socha and Blum (2007)	$ACO_{\mathbb{R}}$	square error percentage	ANN	Path ant: weights setting	BP, LM, Random Search	proben1 (3): cancer1, diabetes1, heart1	ES ES
LF	Lumer and Faieta (1994)	ant sorting	neighborhood similarity function	Topological map	Location ant: location on topographic map	–	arti (1): ($m = 2$, $n = 400$)	DS
ACODF	Tsai et al. (2004)	AS	tour length	Clusters	Path ant: tour	GKA, FSOM+k-means	arti (2): ($m = 2$, $n = 579$); ($m = 2$, $n = 300$) own (1): ($m = 8$, $n = 732$)	DS
ATTA-TM *ATTA-C*	Handl et al. (2006)	ant sorting	neighborhood similarity function	Topological map Clusters	Location ant: location on topographic map	k-means, 1dSOM, avg link, Gap MDS, SOM, lower bound	arti (12): ($m = 2$, $n = 1000$) UCI (7): bcw, derm, dig, iris, wine; yeast, zoo	DS DS
A^4C	Xu and He (2007)	ASM	neighborhood similarity function	Clusters	Location ant: location on topographic map	LF	arti (2): ($m = 2$, $n = 800$) UCI (5): glass, iris, soy, thy, wine	DS

Table 2 Literature overview table of data mining techniques based on PSO

	Authors and year	Metaheuristic variant	Objective function	Model type	Solution component representation	Benchmarked techniques	Datasets	Approach
PSO-Sousa	Sousa et al. (2004)	DPSO LDWPSO CPSO	sens × spec	Rule list	Location particle: rule antecedent	C4.5	UCI (3): bcw, lbc, zoo	ES
PSO/ACO	Holden and Freitas (2005)	AS CPSO	sens × spec	Rule list	Path ant: rule Location particle: rule antecedent	DPSO	Prosite	ES
PSO/ACO2	Holden and Freitas (2008)	AS CPSO	Laplace-corrected precision	Rule list	Path ant: rule Location particle: rule antecedent	PART	UCI (26): aus, auto, bal, bcw, chd, crx, diab, ger, hstat, ion, iris, irisd, krkp, lbc, lym, mush, prom, seg, sonar, soy, ttt, veh, vow, wis, zoo, spl	ES
PSO-HEHR	Holden and Freitas (2009)	PSO with inertia	sens × spec	Hierarchical ensemble of hierarchical rule sets	Location particle: rule weights	Rule EMM, HEHRS, Hierarchical RIPPER	Enzyme (3): Prints, Pfam, Prosite GPCR (3): Prints, Interpro Prosite	ES
PSO-vdMerwe	van der Merwe and Engelbrecht (2003)	PSO with inertia	QE	Clusters	Location particle: cluster centroids	k-means	arti (2): ($m = 2$, $n = 400$), ($m = 2$, $n = 600$) UCI (4): auto, bcw, iris, wine	DS
PSO-DeFalco	(De Falco et al. 2007)	LDWPSO	PCC avg distance to cluster centroid Combination of both	Clusters	Location particle: cluster centroids	PSO, Bayes Net, MLP ANN, RBF ANN, Kstar, Bagging, MultiBoost, NBTree, Ridor, VFI	UCI (12): aus, bal, derm, dia, Ecoli, glass, chd, horse, iris, thy, bcw, wine	DS

From these tables, we can observe that many techniques have been proposed, with little consensus on optimal choice for any of the categories, even within the same data mining task. Most techniques use a different metaheuristic variant to apply, objective function to optimize, datasets to use and other techniques to benchmark with. Surprisingly, few researchers compare their newly proposed techniques with existing swarm intelligence algorithms. These observations lead to a number of directions for future research in this area, which are discussed next. These can be categorized in the following four groups and are discussed in more detail in what follows:

1. Broad-based experimental studies
   - Systematic study into (hyper-)parameters used and metaheuristics adopted
   - Need for real-life applications
   - Need for open source implementations
2. Exploiting swarm intelligence advances
   - Applying SI to other data mining tasks
   - Using new SI techniques
   - Combining approaches
3. Meeting data mining specific requirements
   - Incorporating domain knowledge
   - Interpretability
4. Exploiting the robust, dynamic and distributed nature of SI-based solutions
   - Distributed – Privacy preserving data mining
   - Distributed – Using networked data
   - Dynamic and Robust – Real time applications

5.1 Broad-based experimental studies

5.1.1 Systematic study into (hyper-)parameters used and metaheuristics adopted

Researchers in this area often fail to provide a systematic empirical study where the explicit contribution of their proposed data mining technique is investigated. Benchmarking on some datasets with popular data mining techniques to see how the technique performs is commonly done (although this is already arguable when looking at the literature tables). However, investigating whether the improvement in performance comes from the chosen metaheuristic, specific variant, objective function to optimize, included functions, parameters settings or data characteristics is most often missing. Of the proposed techniques, few actually employ several metaheuristic variants to empirically investigate which performs best in the specific data mining domain. The same applies for the other mentioned algorithm building blocks. For example, most of the classification techniques use a sequential covering approach to induce rule sets. The evaluation functions used for a single rule are quite diverse, though most techniques base themselves on the product of sensitivity and specificity. An empirical study of which rule quality metric to use and why, as done for example by Janssen and Fürnkranz (2010), could surely improve the results, as well as provide insights in the workings and explicit contributions of the algorithm.

Since this research area is rather new, a comparison with other swarm-based methods that use the same metaheuristic would also be useful. A reason this is not done is probably

the lack of overview of existing techniques, one of the contributions we hope to make with this paper. Even across metaheuristics, similarities exist in some of the discussed data mining techniques which are worth investigating empirically. For example, the ACO-based and prey model-based technique for dimensionality reduction both operate in a two-dimensional grid where agents move data instances according to some similarity-derived function. Investigating where and how these techniques differ in clustering behavior could provide interesting insights in the metaheuristics themselves as well.

Whereas ACO is traditionally used for discrete optimization problems, PSO is used for continuous optimization. However, with the recent introduction of continuous ACO variants, a comparison between both approaches using continuous search spaces is more than ever needed. Also, one could investigate whether the discretization step needed to apply ACO adds or decreases the performance.

Based upon an extensive empirical study with a component-wise investigation, the most relevant building blocks for a new technique can be investigated, combining the best of each previously proposed technique. Such an approach, similar to the development of the Frankenstein's PSO (Poli et al. 2007) where based on an empirical study of various PSO variants a new composite PSO variant is developed, could provide a high performing Frankenstein Miner.

5.1.2 Need for real-life applications

Although the UCI data are commonly used for benchmarking and testing out new algorithms, there is a real danger of repository overfitting, whereby marginal improvements on frequently analyzed data sets are considered to be a clear sign of algorithmic superiority (Soares 2003). Hence, it would be interesting to see more real-life applications of swarm intelligence in well-known data mining domains, e.g. credit scoring, customer relationship management, bio-informatics, text mining, etc. Comparing swarm intelligence based techniques to other well-known machine learning techniques on real-life data will provide insight into their behavior and performance.

5.1.3 Need for open source implementations

Most of the proposed techniques are not publicly available, making comparisons across benchmarks difficult. The aforementioned direction of extensive empirical investigations of all algorithmic building blocks would surely benefit from open source implementations. Only few have been made publicly available, which are listed below. The addition of high performing swarm based techniques to the Weka or RapidMiner workbenches should definitely be encouraged.

- AntMiner: http://sourceforge.net/projects/guiantminer
- PSO/ACO2: http://sourceforge.net/projects/psoaco2
- AntMiner+: www.antminerplus.com

5.2 Exploiting swarm intelligence advances

5.2.1 Applying SI to other data mining tasks

The literature tables reveal that swarm intelligence initiatives within data mining are limited to the classification and clustering tasks. Several other data mining tasks have not yet been addressed with the discussed nature inspired techniques. Given the success of existing swarm based techniques for data mining, similar promising results could be expected in the following areas:

– *Regression*: to the best of our knowledge, no swarm intelligence approach for regression has yet been proposed. The first avenue in this research will probably be the adaptation of ant-based classification techniques with continuous ACO or PSO in order to move from a discrete to a continuous solution space which includes the target variable. Moving towards continuous data also allows for continuous variables to be included in the classification techniques (without the need for a discretization step), a direction in which initial steps have been taken by among others cAntMiner and PSO/ACO2.
– *Association/sequence rule mining*: given that many techniques build rule-based classification models, approaches that induce association or sequence rules seem a natural extension.
– *Semi-supervised learning*: Data mining techniques based on swarm intelligence are either employed for clustering or classification. An interesting increasingly popular data mining task can be found at the intersection of both: semi-supervised learning (Chapelle et al. 2006). It has been found that clustering results can significantly be improved by including labels for some data instances.

5.2.2 *Using new SI techniques*

New PSO and ACO variants, such as Frankenstein's PSO, are being proposed at a constant rate nowadays. A modular approach where new variants can be plugged in would be of great value in such a research arena. Interesting is also the application of other existing swarm based approaches such as the gravitational search algorithm (GSA) (Rashedi et al. 2009) and group search optimizer (He et al. 2009). GSA is based on the law of gravity where agents are represented by objects with a mass corresponding to their fitness value. More suitable solutions have higher masses and attract other objects. Intuitively, one sees the similarities with PSO. As such, data mining techniques employing this GSA idea can be devised.

As we are constantly learning more fascinating ways in which nature is able to deal with complex problems, computational descriptions of these phenomena become the basis of new metaheuristics. The mathematical modeling of such behavior can lead to groundbreaking advances in all engineering applications, including data mining. Whereas most of the current literature can be divided according to the metaheuristics ACO and PSO, we foresee many other swarm-based metaheuristic implementations in the near future.

5.2.3 *Combining approaches*

The successful swarm-based data mining approaches typically take some kind of hybrid approach. Some combine a swarm intelligence technique with some conventional optimization technique, such as the PSO-based clustering technique of van der Merwe and Engelbrecht (2003) that starts with the solution obtained by k-means clustering, and the $\text{ACO}_{\mathbb{R}}$ LM technique that combines conventional Levenberg Marquardt with continuous ACO; while others combine several swarm-based approaches, such as the PSO/ACO2 technique of Holden and Freitas (2008). This motivates a closer look at combining swarm intelligence with other state-of-the-art data mining techniques.

In many applications of ACO, adding local search improves the performance (Dorigo and Stützle 2004). However, none of the discussed techniques actually adds local search as well. In our experiments with AntMiner+ this did not reveal any additional performance improvements. A closer investigation on where local search improves data mining models and why constitutes another area for future research.

5.3 Meeting data mining specific requirements

5.3.1 Incorporating domain knowledge

As with most data mining algorithms, the discussed techniques rely solely on modeling repeated patterns that occur in the data. However, in some cases it might be beneficial to include background domain knowledge during the pattern learning process to express known facts and/or relationships within a given problem domain. Two example settings where this could be especially useful are biased data sets and small data sets. Biased data sets frequently occur in data mining because of historical policies adopted by firms. A popular example in credit scoring is reject inference, whereby no performance information is available upon the past rejected applicants due to the historical acceptance policy. Ignoring them during the estimation will result in a biased sample, and hence domain knowledge can be used to limit the effect of the potential bias. Data mining algorithms are also being more and more used to analyse small data sets. Examples are data about new products or less common products (e.g. project finance in a financial context). In these settings, it is of key importance to come up with data mining algorithms that exploit the small volume data set to its fullest extent possible, and combine the learnt patterns with domain knowledge for improved decision making

Domain knowledge can come in multiple forms or shapes. In predictive modeling, a popular example is univariate constraints, whereby the impact of a variable on the target is specified beforehand. Multivariate constraints express known relationships between variables that should be satisfied during the learning process. In descriptive modeling, domain knowledge can express, e.g., which objects should or should not belong to the same cluster, a minimal separation distance between clusters, the need for balanced clusters, etc. Also, it should be possible to allow the domain constraints to be respected in a hard, absolute way, or in a soft, preferred way, depending upon the statistical strength of the correlations and/or patterns in the data.

Incorporating domain knowledge is an important issue for data mining in general, and swarm intelligence provides a unique way to deal with the academically challenging problem of consolidating the automatically generated data mining knowledge with the knowledge reflecting experts' domain expertise. As individuals of a swarm explicitly look for solution (components) in the defined search space, domain knowledge could be included by guiding the individuals towards those regions that are of interest to the domain expert. Potential approaches to do so are adapting the heuristic values (as done in Martens et al. 2006), explicitly enforcing constraints on the agents' movements or including the domain knowledge using some penalty addition to the objective functions. An alternative way of incorporating domain knowledge could be by additional data generation and/or labeling, possibly guided by the swarm intelligence model itself.

5.3.2 Interpretability

In domains where validation of the underlying model is required, e.g. credit scoring and medical diagnosis, a clear insight into the reasoning made by the decision model is necessary. Some data mining techniques, such as neural networks and support vector machines, are quite powerful but their use is often limited in practice because they produce complex, opaque, non-linear models. Experience has shown that decision makers are often reluctant to use black box data mining models. Hence, in order to successfully deploy data mining models in business settings, it becomes of key importance to have transparent, white box

models which at the same time also provide satisfactory statistical performance. Note that this will often amount to choosing an optimal trade-off between model performance and model interpretability, especially since the latter is somewhat subjective in nature.

Until recently, many swarm intelligence algorithms largely focused on estimating models that perform optimally from a purely statistical perspective. The interpretability issue now adds a different dimension to it. One popular example to come up with interpretable models are classification rule induction or rule extraction techniques. Preliminary results of using swarm intelligence algorithms for both purposes have already been reported, such as the AntMiner algorithms and the work by Özbakir et al. (2009). More research is however needed to better quantify the added value of swarm intelligence algorithms, compared to traditional/other rule induction and/or extraction techniques. Graphical models, such as Bayesian networks, are also often considered as probabilistic white box models with a high degree of transparency. Also here, swarm intelligence algorithms might have a role to play by, e.g., offering new ways of learning graphical decision models from data.

5.4 Exploiting the robust, dynamic and distributed nature of SI-based solutions

Most applications of swarm intelligence explicitly use the robust, dynamic and distributed characteristics as being key to the solutions. Previous data mining research focuses on the performance in terms of predictive capability and clustering quality, although robust, dynamic and distributed behavior are also very relevant for specific data mining applications.

5.4.1 Distributed—parallelization

As originally intended, data mining applications more and more work on huge datasets having millions of observations of high-dimensionality. Two popular examples of this are bio-informatics and RFID data. Most of the proposed techniques are computationally very expensive and running them on a single processor computer is not always possible. Furthermore, datasets often are distributed, and centralizing the data can come at high data transfer and communication costs.

Using the distributed nature of ACO has proven to be of great value. One of the most distinctive applications that explicitly uses the distributed characteristic of ACO can be found in AntNet, an algorithm for routing packages in an IP network (Caro and Dorigo 1998). The algorithm shows excellent results, outperforming most of the state-of-the-art existing algorithms. A similar approach could be used here.

Parallelization of the inherently distributed data mining algorithms discussed previously, such that the agents' tasks can be executed concurrently, can provide an answer to the tremendous expansion of data, not only in size but also in location. In this case the work of the agents is processed at several sites, and the agents carry the processed information with them. Thinking for example of supermarket chains where large amounts of data are gathered at each store: instead of sending the data to a centralized warehouse, one could use swarm individuals that visit each of the distributed data sets.

5.4.2 Distributed—privacy preserving data mining

In many settings, data are naturally distributed, yet offer unprecedented potential for advanced and integrated data analysis when aggregated. This is especially relevant in the case of small data sets, whereby a firm only has limited data about a certain product and/or service. By pooling the data with other industry partners, it will become more feasible to

apply data mining techniques at the pooled level. A popular example of this are low default portfolios where banks only have limited observations of a particular target class (i.e. the defaulters) and try to increase the default observations using pooling initiatives with other banks and/or data poolers. By doing this, a much larger data set is created, facilitating the use of data mining techniques. Pooling initiatives have also been proposed for appropriately quantifying operational risk (Baud et al. 2002). Other applications can be thought of whenever a joint venture between firms is set up. Participating companies will surely be interested in analyzing all data available, without actually wanting to share their own data.

Within a pooling context, privacy becomes a major issue. Privacy has increasingly become one of the concerns of data providers in a data mining exercise (Agrawal and Srikant 2000). This surely is an issue when several parties would like to run data mining algorithms on their joint datasets without revealing the data to each other. As the mentioned swarm intelligence techniques typically work with simple creatures with limited memory that can work on distributed data (for example a distributed construction graph within ACO), the application of them in this setting might reveal an additional, increasingly important benefit.

5.4.3 Distributed—using networked data

Using networked data is a very promising research avenue within data mining. Previous studies reveal the predictive power of using such data in supervised learning (see e.g. the work by Mackassy and Provost (2007), Hill et al. (2006)). As Mackassy and Provost (2007) motivate, a network learner typically consists of a local model which only considers object-specific attributes, a relational model which takes into account relationships between objects, and a collective inferencing procedure specifying how objects influence each other mutually. Also here, swarm intelligence techniques might offer interesting new insights and/or perspectives. E.g. one can think of such data as an extreme form of distributed data, with the number of data instances being in the order of the number of data locations. This data typically is first centralized, after which the analysis takes place. However, since the data are explicitly distributed, a swarm based approach that skims this environment could be better suited as a scalable solution. As the future provides us even larger networks, the question becomes whether we will be able to centralize the (necessary) data anyway, and decentralized solutions as those provided by swarm intelligence will become necessary.

5.4.4 Dynamic and robust—real-time applications

In real-time applications, datasets change on a continuous basis. Such applications include fraud detection, real-time data mining based intrusion detection systems (Lee et al. 2001) and counter terrorism. To genuinely be able to exploit the possibility to deal with such dynamic environments, the combination with distributed environments should be considered. Similarly, swarm based techniques can deal with the case where the networked environment is faulty, in the sense that some links within the network can go down. This has been proven to be a major advantage of swarm intelligence based algorithms in the AntNet application, where the failure of routers or connections are dealt with dynamically by the ants (Caro and Dorigo 1998). Further investigating the practical problem domains where these distributed, dynamic and robust solutions are of superior value constitutes a final promising issue for future research.

6 Conclusion

Swarm intelligence is a relatively new domain within AI research. Since its introduction however, it has drawn increasing attention by the research community with applications in a wide variety of engineering problems. The proposed data mining techniques that incorporate these swarm principles often show performance results that are competitive with traditional techniques. Yet, many challenges and also opportunities exist. Since the swarm intelligence research itself is still in its infancy, the research on the application thereof is also continuously in motion. We have set up a framework in which the existing techniques are categorized in two approaches: effective search and data organizing. We envision that this framework provides a useful guide for researchers on comparing or placing new swarm intelligence based data mining techniques within the existing literature.

Although the techniques show very promising results, the topic is still in its infancy. Surveying this topic, we listed some challenges that constitute promising research directions. Firstly, there is a need for more large scale benchmarking studies among the swarm intelligence based techniques, as well as real life applications. Open source implementations in data mining workbenches such as Weka or RapidaMiner will surely speed up such experimental designs. This can prove the potential of such techniques as well as facilitate the use thereof for data mining researchers and practitioners. From that perspective, adapting the existing algorithms to meet specific data mining requirements will accelerate this adoption even further. Secondly, the use of other swarm intelligence techniques besides ACO and PSO, as well as the application to other data mining tasks besides (rule-based) classification and clustering, are yet to be investigated. The good results of the existing techniques motivate such research. Finally, the main advantages of swarm intelligence have not yet been fully used in a data mining context. Whereas other applications often rely on the solutions being robust and able to deal with a dynamic and distributed environment, data mining techniques based on swarm intelligence have not yet fully leveraged these properties. Doing so may show great potential for dealing with emerging challenges in data mining such as handling distributed, networked and real-time data.

Acknowledgements We would like to thank the reviewers of the submissions for the special issue on swarm intelligence for data mining, as well as editor-in-chief Foster Provost for the invitation and guidance in the editing process. Furthermore, we are grateful to Thomas Stützle, Alex Freitas and Bart Minnaert for their insightful comments and suggestions. David Martens and Bart Baesens acknowledge the Flemish Research Council for financial support (FWO postdoctoral research grant, Odysseus grant B.0915.09).

References

Agrawal, R., & Srikant, R. (2000). Privacy-preserving data mining. *SIGMOD Records*, *29*(2), 439–450.

Alatas, B., & Akin, E. (2009). Multi-objective rule mining using a chaotic particle swarm optimization algorithm. *Knowledge-Based Systems*, *22*(6), 455–460.

Alatas, B., Akin, E., & Ozer, A. B. (2009). Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons & Fractals*, *40*(4), 1715–1734.

Baud, N., Frachot, A., & Roncalli, T. (2002). Internal data, external data and consortium data for operational risk measurement: how to pool data properly? *Working paper*, Credit Lyonnais.

Berkhin, P. (2002). Survey of clustering data mining techniques. *Tech. rep.*, Accrue Software Inc.

Bishop, C. (1996). *Neural networks for pattern recognition*. Oxford: Oxford University Press.

Blum, C. (2005a). Beam-ACO—hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Computers & Operations Research*, *32*(6), 1565–1591.

Blum, C. (2005b). Beam-ACO: hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Computers and Operations Research*, *32*(6), 1565–1591.

Bonabeau, E., Dorigo, M., & Tharaulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. Oxford: Oxford University Press.

Bullnheimer, B., Hartl, R., & Strauss, C. (1999). Applying the ant system to the vehicle routing problem. In: Voss, S., Martello, S., Osman, I., Roucairol, C. (Eds.), *Meta-heuristics: advances and trends in local search paradigms for optimization* (pp. 285–296).

Buntine, W. (1991). Theory refinement on Bayesian networks. In *Proceedings of the 17th conference on uncertainty in artificial intelligence* (pp. 52–60). San Mateo: Morgan Kaufmann.

de Campos, L., Fernández-Luna, J., Gámez, J., & Puerta, J. (2002). Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, *31*(3), 291–311.

Caro, G. D., & Dorigo, M. (1998). Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, *9*, 317–365.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Cambridge: MIT Press.

Chickering, D. (2002). Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, *2*, 445–498.

Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73.

Coello Coello, C., Satchidananda, S. D., & Ghosh, S. (2009). *Swarm intelligence for multi-objective problems in data mining*. Berlin: Springer.

Colorni, A., Dorigo, M., Maniezzo, V., & Trubian, M. (1994). Ant system for job-shop scheduling. *Journal of Operations Research, Statistics and Computer Science*, *34*(1), 39–53.

Cordon, O., de Viana, I. F., & Herrera, F. (2002). Analysis of the best-worst ant system and its variants on the QAP. In M. Dorigo, G. Di Caro, & M. Sampels (Eds.), *Lecture notes in computer science: Vol. 2463. Proceedings of the third international workshop on ant algorithms (ANTS'2002)* (pp. 228–234). Brussels: Springer.

Cotta, C., & Muruzábal, J. (2004). On the learning of Bayesian network graph structures via evolutionary programming. In: Lucas, P. (Ed.), *Proceedings of the second workshop on probabilistic graphical models*, Leiden, The Netherlands (pp. 65–72).

Daly, R., & Shen, Q. (2009). Learning Bayesian network equivalence classes with ant colony optimization. *Journal of Artificial Intelligence Research*, *35*(1), 391–447.

De Falco, I., Della Cioppa, A., & Tarantino, E. (2007). Facing classification problems with particle swarm optimization. *Applied Soft Computing*, *7*(3), 652–658.

Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chrétien, L. (1990). The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on from animals to animats* (pp. 356–363). Cambridge: MIT Press.

Domeniconi, C., Gunopulos, D., Ma, S., Yan, B., Al-Razgan, M., & Papadopoulos, D. (2007). Locally adaptive metrics for clustering high dimensional data. *Data Mining and Knowledge Discovery*, *14*(1), 63–97.

Dorigo, M. (2007). Editorial. *Swarm Intelligence*, *1*(1), 1–2.

Dorigo, M. (2009). Swarm-bots and swarmanoid: Two experiments in embodied swarm intelligence. In *Web intelligence* (pp. 2–3). New York: IEEE.

Dorigo, M., & Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge: MIT Press.

Dorigo, M., & Stützle, T. (2009). Ant colony optimization: overview and recent advances. *Tech. rep. TR/IRIDIA/2009-013*, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). Positive feedback as a search strategy. *Tech. rep. 91016*, Dipartimento di Elettronica e Informatica, Politecnico di Milano, IT.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant System: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, *26*(1), 29–41.

Dorigo, M., Birattari, M., Blum, C., Gambardella, L. M., Mondada, F., & Stützle, T. (Eds.) (2004). *Lecture notes in computer science: Vol. 3172. Ant colony optimization and swarm intelligence, Proceedings, 4th international workshop, ANTS 2004, Brussels, Belgium, September 5–8, 2004*. Berlin: Springer.

Freitas, A., & Timmis, J. (2007). Revisiting the foundations of artificial immune systems for data mining. *IEEE Transactions on Evolutionary Computation*, *11*(4), 521–540.

Freitas, A. A. (2003). A survey of evolutionary algorithms for data mining and knowledge discovery. In *Advances in evolutionary computing: theory and applications* (pp. 819–845). New York: Springer.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, *13*, 3–54.

Galea, M., & Shen, Q. (2006). Simultaneous ant colony optimization algorithms for learning linguistic fuzzy rules. In *Swarm intelligence in data mining* (pp. 75–99).

Gambardella, L. M., & Dorigo, M. (1995). Ant-Q: a reinforcement learning approach to the traveling salesman problem. In A. Prieditis & S. Russell (Eds.), *Proceedings of the twelfth international conference on machine learning* (pp. 252–260). Morgan Kaufmann: Palo Alto.

Giraldo, L. F., Lozano, F., & Quijano, N. (2010). Foraging theory for dimensionality reduction of clustered data. *Machine Learning*.

Guntsch, M., & Middendorf, M. (2002). A population based approach for ACO. In *Proceedings of the applications of evolutionary computing on EvoWorkshops 2002* (pp. 72–81). London: Springer.

Handl, J., & Meyer, B. (2002). Improved ant-based clustering and sorting in a document retrieval interface. In G. Goos, J. Hartmanis, & Jv. Leeuwen (Eds.), *Lecture notes in computer science: Vol. 2439. Proceedings of the 7th international conference on parallel problem solving from nature PPSN VII* (pp. 913–923). Berlin: Springer.

Handl, J., Knowles, J. D., & Dorigo, M. (2003). On the performance of ant-based clustering. In *Design and application of hybrid intelligent systems* (pp. 204–213). Amsterdam: IOS Press.

Handl, J., Knowles, J., & Dorigo, M. (2006). Ant-based clustering and topographic mapping. *Artificial Life*, *12*(1), 35–61.

He, S., Wu, Q., & Saunders, J. (2009). Group search optimizer: an optimization algorithm inspired by animal searching behavior. *IEEE Transactions on Evolutionary Computation*, *13*(5), 973–990.

Hettich, S., & Bay, S. D. (1996). The uci kdd archive. http://kdd.ics.uci.edu.

Hill, S., Provost, F., & Volinsky, C. (2006). Network-based marketing: identifying likely adopters via consumer networks. *Statistical Science*, *22*, 256–276.

Hiroyasu, T., Miki, M., Ono, Y., & Minami, Y. (2000). Ant colony for continuous functions. *Tech. rep.*, The Science and Engineering Doshisha University.

Holden, N., & Freitas, A. (2005). A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In *Swarm intelligence symposium, 2005, SIS 2005, Proceedings 2005* (pp. 100–107). New York: IEEE.

Holden, N., & Freitas, A. (2008). A hybrid PSO/ACO algorithm for discovering classification rules in data mining. *Journal of Artificial Evolution and Applications*, *2008*, 11 pages. doi:10.1155/2008/316145

Holden, N., & Freitas, A. (2009). Hierarchical classification of protein function with ensembles of rules and particle swarm optimisation. *Soft Computing*, *13*(3), 259–272.

Holden, N. P., & Freitas, A. A. (2007). A hybrid PSO/ACO algorithm for classification. In *Proceedings of the GECCO-2007 workshop on particle swarms: the second decade* (pp. 2745–2750). New York: ACM.

Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, *35*(6), 1272–1282.

Janssen, F., & Fürnkranz, J. (2010). On the quest for optimal rule learning heuristics. *Machine Learning*, *78*(3), 343–379.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *International conference on neural networks* (pp 1942–1948).

Kennedy, J., & Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. In *IEEE international conference on computational cybernetics and simulation* (Vol. 5, pp. 4104–4108).

King, B. (1967). Step-wise clustering procedures. *Journal of the American Statistical Association*, *69*, 86–101.

Kohonen, T. (2001). *Springer series in information sciences*: *Self-organizing maps*, 3rd edn. Berlin: Springer.

Kuntz, P., Layzell, P., & Snyers, D. (1997). A colony of ant-like agents for partitioning in VLSI technology. In *Proceedings of the 4th international conference on artificial life (ECAL97)*. Cambridge: MIT Press.

Lawler, E. L., Lenstra, J. K., Kan, A. H. G. R., & Shmoys, D. B. (1985). *The Travelling salesman problem*. Chichester: Wiley.

Lee, W., Stolfo, S., Chan, P., Eskin, E., Fan, W., Miller, M., Hershkop, S., & Zhang, J. (2001). Real time data mining-based intrusion detection. In: *DARPA information survivability conference & exposition II, 2001, DISCEX'01, Proceedings* (Vol. 1, pp. 89–100).

Li, Q., Shi, Z., Shi, J., & Shi, Z. (2005). Swarm intelligence clustering algorithm based on attractor. In L. Wang, K. Chen, & Y. S. Ong (Eds.), *Lecture notes in computer science: Vol. 3612. ICNC (3)* (pp. 496–504). Berlin: Springer.

Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, *10*(3), 281–295.

Liu, B., Abbass, H. A., & McKay, B. (2002). Density-based heuristic for rule discovery with ant-miner. In *6th Australasia-Japan joint workshop on intelligent and evolutionary systems* (AJWIS2002), Canberra, Australia.

Liu, B., Abbass, H. A., & McKay, B. (2003). Classification rule discovery with ant colony optimization. In *IAT* (pp. 83–88). Los Alamitos: IEEE Computer Society.

Lu, Y., Wang, S., Li, S., & Zhou, C. (2010). Particle swarm optimizer for variable weighting in clustering high-dimensional data. *Machine Learning*.

Lumer, E. D., & Faieta, B. (1994). Diversity and adaptation in populations of clustering ants. In *SAB94: Proceedings of the third international conference on simulation of adaptive behavior: from animals to animats 3* (pp. 501–508). Cambridge: MIT Press.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

Mackassy, S. A., & Provost, F. (2007). Classification in networked data: a toolkit and a univariate case study. *Journal of Machine Learning Research*, *8*, 935–983.

Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal on Computing*, *11*(4), 358–369.

Martens, D., De Backer, M., Haesen, R., Baesens, B., Mues, C., & Vanthienen, J. (2006). Ant-based approach to the knowledge fusion problem. In *Lecture notes in computer science: Proceedings of the fifth international workshop on ant colony optimization and swarm intelligence* (pp. 85–96). Berlin: Springer.

Martens, D., De Backer, M., Haesen, R., Snoeck, M., Vanthienen, J., & Baesens, B. (2007). Classification with ant colony optimization. *IEEE Transaction on Evolutionary Computation*, *11*(5), 651–665.

Martens, D., Van Gestel, T., & Baesens, B. (2009). Decompositional rule extraction from support vector machines by active learning. *IEEE Transactions on Knowledge and Data Engineering*, *21*(2), 178–191.

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, *8*(3), 204–210.

Montemanni, R., Gambardella, L. M., Rizzoli, A. E., & Donati, A. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, *10*(4), 327–343.

Ngenkaew, W., Ono, S., & Nakayama, S. (2008). The deposition of multiple pheromones in ant-based clustering. *International Journal of Innovative Computing, Information and Control*, *4*(7), 1349–4198.

Montes de Oca, M., Stützle, T., Birattari, M., & Dorigo, M. (2009). Frankenstein's PSO: A composite particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, *13*(5), 1120–1132.

de Oliveira, D., & Bazzan, A. L. C. (2006). Traffic lights control with adaptive group formation based on swarm intelligence. In M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, & T. Stützle (Eds.), *Lecture notes in computer science: Vol. 4150. ANTS workshop* (pp. 520–521). Berlin: Springer.

Otero, F. E., Freitas, A. A., & Johnson, C. G. (2008). cant-miner: an ant colony classification algorithm to cope with continuous attributes. In *ANTS '08: proceedings of the 6th international conference on ant colony optimization and swarm intelligence* (pp. 48–59). Berlin: Springer.

Otero, F. E. B., Freitas, A. A., & Johnson, C. G. (2009). Handling continuous attributes in ant colony classification algorithms. In *CIDM* (pp. 225–231). New York: IEEE.

Özbakir, L., Baykasoglu, A., Kulluk, S., & YapIcI, H. (2009). TACO-miner: an ant colony based algorithm for rule extraction from trained neural networks. *Expert Systems with Applications*, *36*(10), 12,295–12,305.

Paquet, U., & Engelbrecht, A. (2003). Training support vector machines with particle swarms. In *International joint conference on neural networks* (pp. 1593–1998). New York: IEEE Press.

Parpinelli, R. S., Lopes, H. S., & Freitas, A. A. (2001). An ant colony based system for data mining: applications to medical data. In *Proceedings of the genetic and evolutionary computation conference (GECCO-2001)* (pp. 791–797). San Francisco: Morgan Kaufmann.

Parpinelli, R. S., Lopes, H. S., & Freitas, A. A. (2002). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, *6*(4), 321–332.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, *1*(1), 33–57.

Pourtakdoust, S., & Nobahari, H. (2004). An extension of ant colony system to continuous optimization problems. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, & T. Stützle (Eds.), *Lecture notes in computer science: Vol. 3172. Proceedings of fourth international workshop on ant colony optimization and swarm intelligence—ANTS 2004* (pp. 294–301). Berlin: Springer.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information Sciences*, *179*(13), 2232–2248. Special section on high order fuzzy sets.

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, *8*(3), 240–255.

Reynolds, C. W. (1987). Flocks, herds and schools: a distributed behavioral model. In *SIGGRAPH'87: proceedings of the 14th annual conference on computer graphics and interactive techniques* (pp. 25–34). New York: ACM.

Samanta, B., & Nataraj, C. (2009). Application of particle swarm optimization and proximal support vector machines for fault detection. *Swarm Intelligence*, *3*(4), 303–325.

Schockaert, S., Cock, M. D., Cornelis, C., & Kerre, E. E. (2004). Fuzzy ant based clustering. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, & T. Stützle (Eds.), *Lecture notes in computer science: Vol. 3172. Ant colony optimization and swarm intelligence, 4th international workshop, ANTS 2004, Brussels, Belgium, September 5–8, 2004, Proceedings* (pp. 342–349). Berlin: Springer.

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. In *Evolutionary computation proceedings, 1998*. IEEE World congress on computational intelligence (pp. 69–73).

Soares, C. (2003). Is the uci repository useful for data mining? In F. M. A. S. Pires (Ed.), *Lecture notes in artificial intelligence: Vol. 2902. Proceedings of the 11th Portuguese conference on artificial intelligence (EPIA'03)* (pp. 209–223). Beja: Springer.

Socha, K. (2004). Extended ACO for continuous and mixed-variable optimization. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, & T. Stützle (Eds.), *Lecture notes in computer science: Vol. 3172. Proceedings of fourth international workshop on ant colony optimization and swarm intelligence—ANTS 2004* (pp. 25–36). Berlin: Springer.

Socha, K., & Blum, C. (2007). An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. *Neural Computing and Applications*, *16*(3), 235–247.

Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operational Research*, *185*(3), 1155–1173.

Socha, K., Knowles, J. D., & Sampels, M. (2002). A $\mathcal{MAX}$-$\mathcal{MIN}$ ant system for the university timetabling problem. In M. Dorigo, G. Di Caro, & M. Sampels (Eds.), *Lecture notes in computer science: Vol. 2463. Proceedings of ANTS 2002—third international workshop on ant algorithms* (pp. 1–13). Berlin: Springer.

Sousa, T., Silva, A., & Neves, A. (2004). Particle swarm based data mining algorithms for classification tasks. *Parallel Computing*, *30*(5–6), 767–783.

Stephens, D., & Krebs, J. (1986). *Monographs in behavior and ecology. Foraging theory*. Princeton: Princeton University Press.

Stützle, T., & Hoos, H. H. (1996). Improving the ant-system: a detailed report on the $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. *Tech. rep. AIDA 96-12*, FG Intellektik, TU Darmstadt, Germany, URL sherry.ifi.unizh.ch/st96improving.html.

Stützle, T., & Hoos, H. H. (2000). $\mathcal{MAX}$-$\mathcal{MIN}$ ant system. *Future Generation Computer Systems*, *16*(8), 889–914.

Tan, P. N., Steinbach, M., & Kumar, V. (2006). *Introduction to data mining*. Boston: Pearson Education.

Tenenbaum, J. B., Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Tsai, C. F., Tsai, C. W., Wu, H. C., & Yang, T. (2004). ACODF: a novel data clustering approach for data mining in large databases. *Journal of Systems and Software*, *73*(1), 133–145.

van der Merwe, D., & Engelbrecht, A. P. (2003). Data clustering using particle swarm optimization. In *IEEE congress on evolutionary computation (1)* (pp. 215–220). New York: IEEE.

von Neumann, J. (1966). *Theory of self reproducing cellular automata*. Champaign: University of Illinois Press.

Wade, A., & Salhi, S. (2004). An ant system algorithm for the mixed vehicle routing problem with backhauls. In *Metaheuristics: computer decision-making* (pp. 699–719). Norwell: Kluwer Academic.

Witten, I. H., & Frank, E. (2000). *Data mining: practical machine learning tools and techniques with Java implementations*. San Francisco: Morgan Kaufmann.

Xu, X., Chen, L., & He, P. (2007). A novel ant clustering algorithm based on cellular automata. *Web Intelligence and Agent Systems*, *5*(1), 1–14.

Zheng, Y., Ma, l., Zhang, L., & Qian, J. (2003). On the convergence analysis and parameter selection in particle swarm optimization. In *Proceedings of international conference on machine learning and cybernetics* (pp. 1802–1807).