New Algorithms for Generation Decision Trees—Ant-Miner and Its Modifications

Urszula Boryczka and Jan Kozak

Abstract. In our approach we want to ensure the good performance of Ant-Miner by applying the well-known (from the ACO algorithm) two pheromone updating rules: local and global, and the main pseudo-random proportional rule, which provides appropriate mechanisms for search space: exploitation and exploration. Now we can utilize an improved expression of this classification rule discovery system as an Ant-Colony-Miner. Further modifications are connected with the simplicity of the heuristic function used in the standard Ant-Miner. We propose to employing a new heuristic function based on quantitative, not qualitative parameters used during the classification process. The main transition rule will be changed dynamically as a result of the simple frequency analysis of the number of cases from the point of view characteristic partitions. This simplified heuristic function will be compensated by the pheromone update in different degrees, which helps ants to collaborate and is a good stimulant on ants' behavior during the rule construction. The comparative study will be conducted using 5 data sets from the UCI Machine Learning repository.

1 Introduction

Data mining is the process of extracting useful knowledge from real-world data. Among several data mining tasks — such as clustering and classification — this paper focuses on classification. The aim of the classification

Urszula Boryczka
Institute of Computer Science, University of Silesia, Będzińska 39, 41–200 Sosnowiec, Poland, Phone/Fax: (+48 32) 291 82 83
e-mail: urszula.boryczka@us.edu.pl

Jan Kozak
Institute of Computer Science, University of Silesia, Będzińska 39, 41–200 Sosnowiec, Poland, Phone/Fax: (+48 32) 291 82 83
e-mail: j.m.kozak@gmail.com

A. Abraham et al. (Eds.): Foundations of Comput. Intel. Vol. 6, SCI 206, pp. 229–262.
springerlink.com

algorithm is to discover a set of classification rules. One algorithm for solving this task is Ant-Miner, proposed by Parpinelli et al. [66], which employs ant colony optimization techniques [1, 33] to discover classification rules. Ant Colony Optimization is a branch of a newly developed form of artificial intelligence called swarm intelligence. Swarm intelligence is a form of emergent collective intelligence of groups of simple individuals: ants, termits or bees, in which we can observe a form of indirect communication via pheromone. Pheromone values encourage following ants to build good solutions of the analyzed problem and the learning process occurring in this situation is called positive feedback or autocatalysis.

The application of ant colony algorithms to rule induction and classification is a research area that still is not explored and tested very well. The appeal of this approach, similarly to the evolutionary techniques, is that it provides an effective mechanism for conducting a more global search. These approaches are based on a collection of attribute-value terms. Consequently, it can be expected that these approaches will also cope with attribute interactions a little bit better than greedy induction algorithms [35]. What is more, these applications require minimum understanding of the problem domain; the main components are: a heuristic function and an evaluation function, both of which may be employed in the ACO approach in the same form as in the existing literature on deterministic rule induction algorithms.

Ant-Miner is an ant-based system, which it is more flexible and robust than traditional approaches. This method incorporates a simple ant system, in which a heuristic value based on an entropy measure is calculated. Ant-Miner has produced good results when compared with more conventional data mining algorithms, such as C4.5 [69], ID3 and CN2 [14, 15]. Moreover, it is still a relatively recent algorithm, which motivates us to try to amend it. This work proposes some modifications to Ant-Miner to improve it. In the original Ant-Miner, the goal of the algorithm was to produce an ordered list of rules, which was then applied to test data in order in which they were discovered.

The original Ant-Miner was compared to CN2 [14, 15], a classification rule discovery algorithm that uses the strategy for generating rule sets similar to that of the heuristic function used in the main rule of ants' strategy in Ant-Miner. The comparison was made using 6 data sets from the UCI Machine Learning repository that is available at `www.ics.uci.edu/ mlearn/MLRepository.html`. The results were analyzed according to the predictive accuracy of the rule sets and the simplicity of the discovered rule set, which is measured by the number of terms per rule. While Ant-Miner had better predictive accuracy than CN2 on 4 of the data sets and worse on only one of the data sets, the most interesting result is that Ant-Miner returned much simpler rules than CN2. Similar conclusions could also be drawn from the comparison of Ant-Miner to C4.5, a well-known decision tree algorithm [69].

This chapter is organized as follows. Section 1 comprises an introduction to the subject of this work. Section 2 presents the fundamentals of the knowledge

based systems. In section 3, the Swarm Intelligence and Ant Colony Optimization is introduced. In section 4, Ant Colony Optimization in Rule Induction is presented. Section 5 describes the modifications and extensions of original Ant-Miner. In section 6 our further improvements are shown. Then the computational results from five tests are reported. Finally, we conclude with general remarks on this work and outline further research directions.

2 Knowledge-Based Systems

One of the most successful areas of Artificial Intelligence applications are expert systems or, more generally, knowledge-based systems. A popular approach to building knowledge-based systems is using the production system model. There are three main components in the production system model:

- the production rule (long term memory),
- working memory (short term memory),
- the recognize-act cycle.

Knowledge-based systems [13] are usually not able to acquire new knowledge or improve their behavior. It is an important fact, because intelligent agents should be capable of learning. Machine learning is a field of Artificial Intelligence. To deal with such kind of problems, machine learning techniques can be incorporated into knowledge-based systems. Learning, in our considerations, refers to positive changes toward improved performance. When symbol-based machine learning is used, a learner must search the concept space to find the desired concept. These specific improvements must be introduced in programs to direct and order of search, as well as to use of available training data and heuristics to search efficiently. Because of that we will analyze the pheromone trail as a learning possibility.

The main idea of machine learning can be demonstrated through inductive reasoning, which refers to the process of deriving conclusions from given facts. A well-known tree induction algorithm adapted from machine learning is ID3 [67], which employs a process of constructing a decision tree in a top-down fashion. A decision tree is a hierarchical representation that can be used to determine the classification of an object by testing its properties for certain values. ID3 has been proven a very useful method, yet there are many restrictions that make this algorithm not applicable in many real world situations. C4.5 was developed to deal with these problems, and can be considered a good solution when using bad or missing data, continuous variables, as well as data of large size.

Many useful forms of this algorithm have been developed, ranging from simple data structures to more complicated ones . We can distinguish four representations of decision trees:

- a binary decision tree,
- linear decision trees,

- classification and regression trees (CART) — a binary decision trees, which split a single variable at each node. CART performs a recursively search on all variables to find an optimal splitting rule for each node.
- a Chi-squared automatic interaction detector (CHAID) — where multiple branches can be produced.

Several methods have been proposed for the rule induction process, such as: ID3 [67], C4.5 [69], CN2 [14, 15], CART [6], and AQ15 [60]. There are two categories of these approaches: sequential covering algorithms and simultaneous covering algorithms. The latter group is represented by: ID3 and C4.5.

Algorithm 1. Algorithm ID3

```
ID3_tree (examples, properties)
if all entries in examples are in the same category of the decision variable then
    return the leaf node labeled with that category;
end
else
    Calculate information gain;
    Select the property P with highest information gain;
    Assign the root of a current tree = P;
    Assign properties = properties - P;
    for each value V of P do
        Create a branch of the tree labeled with V;
        Assign examples_V = the subset of examples with values V for property P;
        Applied ID3_tree (examples_V, properties) to branch V;
    end
end
```

C4.5 is an improved version of ID3. C4.5 follows a, divide and conquer" strategy to build a decision tree through recursive partitioning of a training set. The process of building the decision tree by C4.5 begins with choosing an attribute (a corresponding variablesof this attribute) to split the data set into subsets. Selecting the best splitting attribute is based on the heuristic criteria. This methodology includes: Information Gain [69], Information Gain Ratio [69], Chi-square test [68], and Gini-index [6]. The Information Gain of an attribute A relative to a set of examples S is defined as follows:

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{S_v}{S} Entropy(S_v),$$

where:

- $values(A)$ is a set of all possible values for A,
- S_v is a subset of S for which attribute A has value v.

$Entropy(S)$ is the entropy of S, which characterizes the purity of a set of examples. If a class attribute has k different values, the $Entropy(S)$ is defined as:

$$Entropy(S) = -\sum_{c=1}^{k} \frac{n_c}{N} log_2 \frac{n_c}{N}$$

where:

- n_c is the number of examples for the cth class,
- $N = \sum_{c=1}^{k} n_c$ is the total number of examples in the data set.

The higher the entropy is, more incidentally the k values are distributed all over the data set. Unfortunately, the Information Gain favors attributes with many values over those with a small number of values. So, an alternative criterion — Information Gain Ratio is employed in the C4.5 approach.

The Information Gain Ratio is calculated as follows:

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplittingInfo(S, A)}$$

where $SplittingInformation$ is used to penalize attributes with too many values; we can calculate it as follows:

$$SplittingInfo(S, A) = \sum_{i=1}^{k} \frac{S_i}{S} log_2 \frac{S_i}{S},$$

where S_i is a subset of training set S, which is represented by attribute A.

Our approach is not associated with CN2, which uses accuracy as a heuristic criterion to determine the best rule, but analyzing only the rule accuracy and thereforecan result in less general rules. Consequently, the Laplace error estimate [14] is used to establish generic rules, which cover a large number of examples:

$$LaplaceAccuracy = \frac{n_c + 1}{n_{tot} + k}$$

where:

- k is the number of classes in the domain,
- n_c is the number of examples in the predicted class c covered by the rule,
- n_{tot} is the total number of examples covered by the rule.

3 Swarm Intelligence and Ant Colony Optimization

Ant colonies exhibit very interesting collective behaviors: even if a single ant has only simple capabilities, the behavior of the whole ant colony is highly structured. This is the result of co-ordinated interactions and co-operation between ants (agents) and represents Swarm Intelligence. this term is applied to any work involving the design of algorithms or distributed problem-solving devices inspired by the collective behavior of social insects [4].

3.1 Swarm Intelligence

Swarm Intelligence (SI) is an innovative distributed intelligent paradigm for solving optimization problems that originally took its inspiration from the

biological examples by swarming, self-organizing foraging phenomena in social insects. There are many examples of swarm optimization techniques, such as: Particle Swarm Optimization, Artificial Bee Colony Optimization and Ant Colony Optimization (ACO). The last approach deals with artificial systems, inspired by the natural behaviors of real ants, especially foraging behaviors based on the pheromone substances laid on the ground.

The fundamental concept underlying the behavior of social insects is self-organization. SI systems are complex systems — collections of simple agents that operate in parallel and interact locally with each other and their environment to produce emergent behavior.

The basic characteristic of metaheuristics from nature could be summarized as follows [1]:

- they model a phenomenon in nature,
- they are stochastic,
- in the case of multiple agents, they often have parallel structure,
- they use feedback information for modifying their own parameters — they are adaptive.

Developing algorithms that utilize some analogies with nature and social insects to derive non-deterministic metaheuristics capable of obtaining good results in hard combinatorial optimization problems could be a promising field of research.

The optimization algorithm we propose in this paper was inspired by the previous works on Ant Systems and, in general, by the term – stigmergy. This phenomenon was first introduced by P.P. Grasse [45, 46]. Stigmergy is easily overlooked, as it does not explain the detailed mechanism by which individuals co-ordinate their activities. However, it does provide a general mechanism that relates individual and colony-level behavior: individual behavior modifies the environment, which in turn modifies the behavior of other individuals. The synergetic effect is understood as a result of natural social behavior among individuals connected by the main goal.

Many features of the collective activities of social insects are self-organized. The theories of self-organization (SO) [20], originally developed in the context of physics and chemistry to describe the emergence of macroscopic patterns out of processes and interactions defined at the macroscopic level. They can be extended to social insects to show that complex collective behavior may emerged from interactions among individuals that exhibit simple behavior: in these cases, there is no need to refer to individual complexity to explain complex collective behavior. Recent researches show that SO is indeed a major component of a wide range of collective phenomena in social insects [4].

Self-organization in social insects often requires interaction among insects: such interactions can be either direct and indirect. Direct interactions are the ,,obvious" interactions: antennation, trophallaxis (food or liquid exchange), mandibular contact, visual contact, chemical contact (the odour of nearby nestmates) etc. Indirect interactions are more subtle: two individuals interact

indirectly when one of them modifies the environment and the other responds to the new environment, at a later time. Such an interaction is an example of stigmergy (from Greek stigma: sting, and ergon: work) employed to explain task co-ordination and regulation in the context of nest reconstruction in termites of the genus Macrotermes.

The ant systems – the first version of the ant colony optimization systems mimic the nature and take advantage of various observations made by people, who studied ant colonies. Especially the ACO algorithms were inspired by the experiment run by Goss et al. [44] using a colony of real ants. Deneubourg et al. [4] using a special experimental setup showed that the selection of a path to a food-source in the Argentine ant is based on self-organization. Individual ants deposit a chemical substance called pheromone as they move from a food source to their nest and foragers follow such pheromone trails. The process in which an ant is influenced toward a food source by another ant or by a chemical trail is called recruitment, and recruitment based solely on chemical trails is called mass recruitment.

3.2 Ant Colony Optimization as a New Metaheuristics

In this paper we defined an ant algorithm to be a multi–agent system inspired by the observation of real ant colony behavior exploiting the stigmergic communication paradigm. The optimization algorithm we propose in this paper was inspired by the previous works on Ant Systems and, in general, by the term — stigmergy. This phenomenon was first introduced by P.P. Grasse [45, 46].

The last two decades have been highlighted by the development and the improvement of approximative resolution methods, usually called heuristics and metaheuristics. In the context of combinatorial optimization, the term heuristic is used as a contrast to methods that guarantee to find a global optimum, such as branch and bound or dynamic programming. A heuristic is defined by [70] as a technique which seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is. Often heuristics are problem-specific, so that a method which works for one problem cannot be used to solve a different one. In contrast, metaheuristics are powerful techniques applicable generally to a large number of problems. A metaheuristic refers to an iterative master strategy that guides and modifies the operations of subordinate heuristics by combining intelligently different concepts for exploring and exploiting the search space [43, 64]. A metaheuristic may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The family of metaheuristics includes, but is not limited to, Constraint Logic Programming, Genetic Algorithms, Evolutionary Methods, Neural Networks,

Simulated Annealing, Tabu Search, Non–monotonic Search Strategies, Scatter Search, and their hybrids. The success of these methods is due to the capacity of such techniques to solve in practice some hard combinatorial problems.

Also some traditional and well-established heuristic optimization techniques, such as Random Search, Local Search [3], or the class of Greedy Heuristics (GH) [3] may be considered as metaheuristics.

Considering that Greedy Heuristics are available for most practical optimization problems and often produce good results, it seems, that in these cases, it is less expensive with regard to the development costs to further improve their solution quality by extending them to repetitive procedures rather than to replace them by iterative heuristics which follow completely different optimization strategies. So it seems desirable to have a constructive and repetitive metaheuristics including GH as a special (boundary) case.

An essential step in this direction was the development of Ant System (AS) by Dorigo et al. [25, 31, 28], a new type of heuristic inspired by analogies to the foraging behavior of real ant colonies, which has proven to work successfully in a series of experimental studies. Diverse modifications of AS have been applied to many different types of discrete optimization problems and have produced very satisfactory results [27]. Recently, the approach has been extended by Dorigo and Di Caro [26] to a full discrete optimization metaheuristics, called the Ant Colony Optimization (ACO) metaheuristics.

AS, which was the first ACO algorithm [16, 29, 31], was designated as a set of three ant algorithms differing in the way the pheromone trail was updated by ants. Their names were: ant-density, ant-quantity, and ant-cycle. A number of algorithms, including the metaheuristics, were inspired by ant-cycle, the best performing of the ant algorithms.

The Ant Colony System (ACS) algorithm has been introduced by Dorigo and Gambardella [28, 30] to improve the performance of Ant System [30, 39], which allowed to find good solutions within a reasonable time for small size problems only. The ACS is based on 3 modifications of Ant System:

- a different node transition rule,
- a different pheromone trail updating rule,
- the use of local and global pheromone updating rules (to favor exploration).

The node transition rule is modified to allow explicitly for exploration. An ant k in city i chooses the city j to move to following the rule:

$$j = \begin{cases} \arg\max_{u \in J_i^k} \{[\tau_{iu}(t)] \cdot [\eta_{iu}]^{\beta}\} & \text{if } q \leq q_0 \\ J & \text{if } q > q_0 \end{cases}$$

where q is a random variable uniformly distributed over $[0, 1]$, q_0 is a tunable parameter ($0 \leq q_0 \leq 1$), and $J \in J_i^k$ is a city that is chosen randomly according to a probability:

$$p_{iJ}^{k}(t) = \begin{cases} \dfrac{\tau_{iJ}(t) \cdot [\eta_{iJ}]^{\beta}}{\sum\limits_{l \in J_i^k} [\tau_{il}(t)] \cdot [\eta_{il}]^{\beta}} \end{cases}$$

which is similar to the transition probability used by Ant System. We see therefore that the ACS transition rule is identical to Ant System's one, when $q > q_0$, and is different when $q \leq q_0$. More precisely, $q \leq q_0$ corresponds to the exploitation of the knowledge available about the problem, that is, the heuristic knowledge about distances between cities and the learned knowledge memorized in the form of pheromone trails, whereas $q > q_0$ favors more exploration.

In Ant System all ants are allowed to deposit pheromone after completing their tours. By contrast, in the ACS only the ant that generated the best tour since the beginning of the trail is allowed to globally update the concentrations of pheromone on the branches. The updating rule is:

$$\Delta_{ij}(t+n) = (1-\alpha) \cdot \tau_{ij}(t) + \alpha \cdot \Delta\tau_{ij}(t, t+n)$$

where (i, j) is the edge belonging to T^+, the best tour since the beginning of the trail, α is a parameter governing pheromone decay, and:

$$\Delta\tau_{ij}(t, t+n) = \frac{1}{L^+}$$

where L^+ is the length of the T^+.

The local update is performed as follows: when, while performing a tour, ant k is in city i and selects city $j \in J_i^k$ to move to, the pheromone concentration on edge (i, j) is updated by the following formula:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_0$$

The value of τ_0 is the same as the initial value of pheromone trails and it was experimentally found that setting $\tau_0 = (n \cdot L_{nn})^{-1}$, where n is the number of cities and L_{nn} is the length of a tour produced by the nearest neighbor heuristic, produces good results [28, 39].

3.3 Hybrid Approaches in ACO

A single algorithm cannot be the best approach to solve quickly every atomization problem. One way to deal with this issue is to hybridize an algorithm with more standard procedures, such as greedy methods or local search procedures. Individual solutions obtained so far can be improved using another searching procedures, and then placed back in competition with other solutions not improved yet. There are so many ways to hybridize that there is a common tendency to overload hybrid systems with too many components.

Some modern hybrid systems contain fuzzy-neural-evolutionary components, together with some other problem-specific heuristics. Ant Colony Optimization algorithms are extremely flexible and can be extended by incorporating diverse concepts and alternative approaches. For example Stüzle and Hoos [75] have implemented version of AS-QAP based on their MAX-MIN Ant System (MMAS). They discussed the possibility of incorporating a local search (2-opt algorithm and short Tabu Search runs) into the Ant Algorithm in the case of the QAP. Ant Colony Optimization technique can also be enhanced by including new points in a standard version of this algorithm. Gambardella et al. [4] proposed a new Ant Colony Optimization algorithm, called the hybrid Ant System (HAS-QAP), which differs significantly from the ACO algorithm. There are two novelties:

- ants modify solutions, as opposed to building them,
- pheromone trails are used to guide the modifications of solutions, and not as an aid to direct their construction.

In the initialization phase, each ant $k, k = 1, \ldots, m$, is given a permutation Pi^k which consists of a randomly generated permutation to its local optimum by the local search procedure. This procedure examines all possible ways in a random order and accepts any swap that improves the current solution. Gambardella and Dorigo [39] have developed the Hybrid Ant System (called HAS-SOP), which is a modified version of ACS plus a local optimization procedure specifically designed to solve the sequential ordering problem (SOP). Results obtained with HAS-SOP were very useful (effective). HAS-SOP, which was tested on an extensive number of problems, outperforms existing heuristics in terms of solution quality and computational time. Also, HAS-SOP has improved many of the best known results on problems maintained in TSPLIB.

The ACO metaheuristics has been successfully applied to many discrete optimization problems, as listed in tab. 1 [4]. ACO algorithms results turned out to be competitive with the best available heuristic approaches. In particular, the results obtained by the application of ACO algorithms to the TSP are very encouraging. They are often better than those obtained using other general purpose heuristics like Evolutionary Computation or Simulated Annealing. Also, when adding to ACO algorithms Local Search procedures based on 3-opt, the quality of the results is close to that obtainable by other state-of-the-art methods. ACO algorithms are currently one of the best performing heuristics available for the particularly important class of quadratic assignment problems which model real world problems. AntNet, an ACO algorithm for routing in packet switched networks, outperformed a number of state-of-the-art routing algorithms for a set of benchmark problems. AntNet-FA, an extension of AntNet for connection oriented network routing problems, also shows competitive performance. HAS-SOP, an ACO algorithm coupled to a local search routine, has improved many of the best known results on a wide set of benchmark instances of the sequential ordering problem (SOP), i.e. the

Table 1 Current applications of ACO algorithms

Problem name	Algorithm	Main references
Traveling salesman	AS	Dorigo [31]
		Dorigo [25]
		Dorigo [32]
	Ant-Q	Gambardella [37]
	ACS and	
	ACS-3-opt	Dorigo [30]
		Dorigo [29]
		Gambardella [38]
	AS	Stützle [75]
		Stützle [74]
	ASrank	Bullheimer [8]
Quadratic assignment	AS-QAP	Maniezzo [58]
	HAS-QAP	Gambardella [41]
		Gambardella [42]
	MM AS-QAP	Stützle [76]
	ANTS-QAP	Maniezzo [54]
	AS-QAP	Maniezzo [56]
Scheduling problems	AS-JSP	Colorni [17]
	AS-FSP	Stützle [73]
	ACS-SMTTP	Bauer [2]
	ACS-SMTWTP	Denbesten [19]
Vehicle routing	AS-VRP	Bullheimer [10]
		Bullheimer [7]
		Bullheimer [9]
	HAS-VRP	Gambardella [40]
Connection-oriented network routing	ABC	Schoonderwoerd [72]
		Schoonderwoerd [71]
	ASGA	White [81]
	AntNet-FS	Di Caro [23]
	ABC-smart ants	Bonabeau [5]
Connection-less network routing	AntNet and	
	AntNet-FA	Di Caro [21]
		Di Caro [22]
		Di Caro [24]
	Regular ants	Subramanian [77]
	CAF	Heusse [47]
	ABC-backward	Vanderput [78]
Sequential ordering	HAS-SOP	Gambardella [39]
Graph coloring	ANTCOL	Costa [18]
Shortest common supersequence	AS-SCS	Michel [61]
		Michel [62]

Table 1 (*continued*)

Problem name	Algorithm	Main references
Frequency assignment	ANTS-FAP	Maniezzo [55] Maniezzo [57]
Generalized assignment	MMAS-GAP	Ramalhinho [53]
Multiple knapsack	AS-MKP	Leguizamon [50]
Optical networks routing	ACO-VWP	Navarro [79]
Redundancy allocation	ACO-RAP	Liang [51]

problem of finding the shortest Hamiltonian circle on a graph which satisfies a set of precedence constraints on the order in which cities are visited. ACO algorithms have also been applied to a number of other discrete optimization problems like the shortest common supersequence problem, the multiple knapsack, single machine total tardiness, and others, with very promising results.

Multiple Ant Colony System for Vehicle Routing Problem with Time Windows (MACS-VRPTW) is a new ACO based approach to solve vehicle routing problems with time windows. The general idea of adapting ACS for multiple objectives in VRPTW is to define two colonies, each dedicated to the optimization of a different objective function. In the MACS-VRPTW algorithm the minimization of the number of tours (or vehicles) and the minimization of the total travel time are optimized simultaneously by two ACS based colonies — ACS-VEI and ACS-TIME. The goal of the first colony is to reduce the number of vehicles used, whereas the second colony optimises the feasible solutions found by ACS-VEI. Each colony uses its own pheromone trails but collaborates with another one by sharing information about the best results obtained so far. MACS-VRPTW is shown to be competitive with the best existing methods both in terms of solution quality and computation time. Moreover, MACS-VRPTW improves good solutions for a number of problem instances known from the literature.

4 Ant Colony Optimization in Rule Induction

The adaptation of ant colony optimization to rule induction and classification is a research area still not well explored and examined. Ant-Miner is a sequential covering algorithm that merged concepts and principles of ACO and rule induction. Starting from a training set, Ant-Miner generates a set

of ordered rules through iteratively finding an appropriate rule that covers a subset of the training data, adds the formulated rule to the induced rule list and then removes the examples covered by this rule until the stopping criteria are reached.

ACO has a number of features that are important for computational problem solving [34]:

- it is relatively simple and easy to understand and then to implement
- it offers emergent complexity to deal with other optimization techniques
- it is compatible with the current trend towards greater decentralization in computing
- it is adaptive and robust and it is enables coping with noisy data.

There are numerous characteristics of ACO which are crucial in data mining applications. ACO, in contrary to deterministic decision trees or rule induction algorithms, during rule induction tries to alleviate the problem of premature convergence to local optima because of a stochastic element which prefers a global search in problem's search space. Secondly, ACO metaheuristic is a population–based one. It permits the system to search in many independently determined points in the search space concurrently and to use the positive feedback between ants as a search mechanism [65].

Ant-Miner was invented by Parpinelli et al. [66, 65]. It was the first Ant algorithm for rule induction and it has been shown to be robust and comparable with the CN2 [14] and C4.5 [69] algorithms for classification. Ant-Miner generates solutions in the form of classification rules. The original Ant–Miner has a limitation that it can only process discrete values of attributes.

We present a short review of the main aspects of the rule discovery process by Ant-Miner together with the description of the Ant-Miner algorithm. Ant-Miner is a sequential covering approach to discover a list of classification rules, by discovering one rule at a time until all or almost all the examples in the training set are covered by the discovered rules. When the algorithm begins to work, the training set holds all the examples and the discovered rule list is empty. Each ant builds one rule. At the end of the **While** loop, the best rule is added to the discovered rule list. Examples having the class predicted by the rule, are removed from the training set before the next iteration of the **While** loop. The process of rule discovering is repeated as long as the number of uncovered examples in the training set is less than a user-specified threshold.

Three main phases may be distinguish in every iteration of the **Repeat — Until** loop: rule construction, rule pruning and pheromone updating. In the first step, Ant_t starts with an empty rule without term in the antecedent, and adds one term at time until one of the two criteria is satisfied:

- there is a term added to the current rule R_t, which makes the rule cover a number of examples less than a user specified parameter — *MinExamplesPerRule*,

- there are no more terms which can be added to the rule antecedent by the current Ant_t. Please notice that no rule can contain any attribute twice (with different values).

During the construction of the partial rule, Ant_t builds a path, and every term added to the partial rule represents the direction of how the path is being extended. The next term will be added to the partial rule according to the probability of a term being selected. This value depends on the value of the heuristic function and the amount of the pheromone associated with the term. After this phase, pruning will be performed. The aim of this process is to remove all irrelevant terms and improve the rule R_t. The importance of this process is due to the probabilistic mechanism performed during the construction of the rule. We want to eliminate the drawback of ignoring interactions between attributes. Rule pruning iteratively remove one term at a time from the rule as long as this improves the quality of the rule. This process will be performed till there is only one term in the rule, or no term improves the quality of the rule.

The next phase — pheromone updating is performed according to the quality of the rule (predictive accuracy) — each term in the antecedent of the just-pruned rule is affecting this value. The process of reducing the value of pheromone will be performed for other terms, which are not present in the rule antecedent.

The main loop is repeated until one of the following termination criteria is achieved:

- the number of constructed rules is equal or greater than the number of ants,
- the rule constructed by Ant_t is equal to $No\_rules\_converg-1$ rules (defined by the user).

All cells in the pheromone table are initialized equally to the following value:

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^{a} b_i}$$

where:

- a — the total number of attributes,
- b_i — the number of values in the domain of attribute i.

The probability is calculated for all of the attribute–value pairs, and the one with the highest probability is added to the rule. The transition rule in Ant-Miner is given by the following equation:

$$p_{ij} = \frac{\tau_{ij}(t) \cdot \eta_{ij}}{\sum_{i}^{a} \sum_{j}^{b_i} \tau_{ij}(t) \cdot \eta_{ij}}, \forall i \in I$$

where:

Algorithm 2. Algorithm Ant-Miner

```
1  TrainingSet = {all training examples};
2  DiscoveredRuleList = [ ]; /* rule list is initialized with an empty list */
3  while TrainingSet > MaxUncoveredExamples do
4      t = 1;/* ant index */
5      j = 1;/* convergence test index */
6      Initialize all trails with the same amount of pheromone;
7      repeat
8          Ant_t starts with an empty rule and incrementally constructs a classification rule
           R_t by adding one term at a time to the current rule;
9          Prune rule R − t;
10         Update the pheromone amount of all trails by increasing pheromone in the trail
           followed by Ant_t (proportional to the quality of R_t) and decreasing pheromone
           amount in the other trails (simulating pheromone evaporation);
11         /* update convergence test */
12         if R_t is equal to R_t − 1 then
13             j = j + 1;
14         end
15         else
16             j = 1;
17         end
18         t = t + 1;
19     until (t ≥ No_of_ants) OR (j ≥ No_rules_converg) ;
20     Choose the best rule R_best among all rules R_t constructed by all the ants;
21     Add rule R_best to DiscoveredRuleList;
22     TrainingSet = TrainingSet - (set of examples correctly covered by R_best);
23 end
```

- η_{ij} is a problem-dependent heuristic value for each term,
- τ_{ij} is the amount of pheromone currently available at time t on the connection between attribute i and value j,
- I is the set of attributes that are not yet used by the ant,
- Parameter β is equal to 1.

In Ant-Miner, the heuristic value is supposed to be an information theoretic measure for the quality of the term to be added to the rule. For prefering the quality is measured in terms of entropy term to the others, and the measure is given as follows:

$$\eta_{ij} = \frac{log_2(k) - InfoT_{ij}}{\sum_i^a \sum_j^{b_j} (log_2(k) - InfoT_{ij})}$$

where the function $Info$ is similar to another function employed in C4.5 approach:

$$InfoT_{ij} = -\sum_{w=1}^{k} \left[\frac{freqT_{ij}^w}{T_{ij}}\right] log_2 \left[\frac{freqT_{ij}^w}{|T_{ij}|}\right]$$

where: k is the number of classes, $|T_{ij}|$ is the total number of cases in partition T_{ij} (the partition containing the cases, where attribute A_i has the value V_{ij}), $freqT_{ij}^w$ is the number of cases in partition T_{ij} with class w, b_i is a number of values in the domain of attribute A_i (a is the total number of attributes).

The higher the value of $InfoT_{ij}$, is the less likely is that the ant will choose $term_{ij}$ to add to its partial rule.

Please note that this heuristic function is a local method and it is sensitive to attribute interaction. The pheromone values assigned to the term have a more global nature. The pheromone updates depend on the evaluation of a rule as a whole, i.e. we must take into account interaction among attributes appearing in the rule.

The heuristic function employed here comes from the decision tree world and it is similar to the method used in algorithm C4.5. There are many other heuristic functions that can be adapted and used in Ant-Miner. We can derive them from information theory, distance measures or dependence measures.

The rule pruning procedure iteratively removes the term whose removal will cause the maximum increase in the quality of the rule. The quality of a rule is measured using the following formula:

$$Q = \left(\frac{TruePos}{TruePos + FalseNeg}\right) \cdot \left(\frac{TrueNeg}{FalsePos + TrueNeg}\right)$$

where:

- TruePros – the number of cases covered by the rule and having the same class as the one predicted by the rule,
- FalsePros – the number of cases covered by the rule and having a different class from the one predicted by the rule,
- FalseNeg – the number of cases that are not covered by the rule while having a class predicted by the rule,
- TrueNeg – the number of cases that are not covered by the rule which have a different class from the class predicted by the rule.

The quality measure of a rule is determined by:

$$Q = sensitivity \cdot specificity.$$

We can say that accuracy among positive instances determines sensitivity, and the accuracy among negative instances determines specificity. Now we take into account only the rule accuracy, but it can be changed to analyze the rule length and interestingness.

Once each ant completes the construction of the rule, pheromone updating is carried out as follows:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot Q, \forall term_{ij} \in \text{the rule}$$

The amounts of pheromones of terms belonging to the constructed rule R are increased in proportion to the quality of Q. To simulate pheromone evaporation τ_{ij}, the amount of pheromone associated with each $term_{ij}$ which does not occur in the constructed rule must be decreased. The reduction of pheromone of an unused term is performed by dividing the value of each

τ_{ij} by the summation of all τ_{ij}. The pheromone levels of all terms are then normalized.

5 Modifications

The authors of Ant-Miner [66, 65] suggested two directions for future research:

1. Extension of Ant-Miner to cope with continuous attributes;
2. The investigation of the effects of changes in the main transition rule:
   a. the local heuristic function,
   b. the pheromone updating strategies.

Recently, Galea [36] proposed a few modifications in Ant-Miner. Firstly, the population size of ants was changed (from 1 to 50 ants). Secondly, new stopping criteria were examined. Compared to the classical Ant-Miner, the implementation of Ant-Miner proposed by Galea [36] produces slightly lower predictive accuracy then Parpinelli et al.'s. Galea also examined the effect of employing the pseudo random proportional rule. Galea suggests to study the relationship between the characteristics of the dataset, such as:

- the distribution of the data set and correlation between attributes,
- the selection of algorithms: stochastic or deterministic algorithms.

The Ant-Miner modifications and extensions have been presented in many articles, as listed in Tab.2.

This subsection gives a brief overview of Ant-Miner extensions. Some of the propositions are relatively simple and, as a result, give the same type of classification rules discovered by this algorithm. Another modifications cope with the problem of attributes having ordered categorical values, some of them improve the flexibility of the rule representation language. Finally, more sophisticated modifications have been proposed to discover multi-label classification rules and to investigate fuzzy classification rules. Certainly there are still many problems and open questions for future research.

6 Experiments and Remarks

An Ant Colony Optimization technique is in essence, a system based on agents which simulate the natural behavior of ants, incorporating a mechanism of cooperation and adaptation, especially via pheromone updates. When solving different problems with the ACO algorithm we have to analyze three major functions. Choosing these functions appropriately helps to create better results and prevents stucking in local optima of the search space.

The first function is a problem-dependent heuristic function (η) which measures the quality of terms that can be added to the current partial rule. The

Table 2 Modifications of the Ant-Miner algorithm

1	Modifications or extractions: The class is known during the rule construction. All ants construct rules predicting the same class [12, 36, 59, 48]	
Authors	Year	Open questions:
Chen, Chen & He, Galea & Shen, Martens et al., Smaldon & Freitas,	2006 2006 2006 2006	This approach leads to: a new heuristic function and new pheromone update strategies.
2	Modifications or extensions: A new heuristic function. The replacement of the entropy reduction heuristic function by a simpler heuristic function [12, 52, 59, 63, 48, 80]	
Authors	Year	Open questions:
Chen, Chen & He, Liu, Abbas & Mc Kay, Martens et al., Oakes, Smaldon & Freitas, Wang & Feng,	2006 [12] 2004 2006 2004 2006 2004	There is no quarantee that the use of pheromones would completely compensate the use of a less effective function. The heuristic function can be computed with respect to the number of cases and attributes (only once in the initialization of the algorithm).
3	Modifications or extensions: Using a pseudo random proportional transition rule. Using parameter q_0 [12, 52, 80]	
Authors	Year	Open questions:
Chen, Chen & He, Liu, Abbas & Mc Kay, Wang & Feng,	2006 2004 2004	This transition rule has an advantage that it allows the user to have an explicit control over the exploitation versus exploration trade-of. It requires to choose a good value for the parameter q_0 in empirical way.
4	Modifications or extensions: A new rule Quality measure. Ant-Miner's rule quality is based on the product of sensitivity and specificity. The replacement of the rule quality function by measures that are essentially based on the confidence and coverage of a rule [12, 59]	
Authors	Year	Open questions:
Chen, Chen & He, Martens et al.,	2006 2006	It would be interesting to perform extensive experiments comparing the effectiveness of these kinds of rule quality measures.
5	Modifications or extensions: New pheromone updating rules. The use of an explicit pheromone evaporation rate. A self adaptive parameter. Different procedures to update pheromone as a function of the rule quality [52, 59, 48, 80]	

Table 2 (*continued*)

Authors	Year	Open questions:
Liu, Abbas & Mc Kay, Martens et al., Smalton & Freitas, Wang & Feng,	2004 2006 2006 2004	It is important to change the original Ant-Miner's formula to update pheromone in order to cope better with law-quality rules. We treat evaporation trails as a form of a learning factor.
6 Modifications or extensions: Ant-Miner can cope with ordered values of categorical attributes (not continuous) [63, 59]		
Authors	Year	Open questions:
Oakes Martens et al.,	2004 2006	It is important to point out that the connection with the Rough Sets theory in the contrast to Fuzzy Sets is a new augmentation.
7 Modifications or extensions: Dropping the rule pruning procedure. The removal of pruning makes the Ant-Miner+ significantly faster [59]		
Authors	Year	Open questions:
Martens et al.,	2006	It would be interesting to evaluate, whether for the discovered by Ant-Miner+ rules the predictive accuracy could be increased by the use of a deterministic rule pruning procedure driven by the rule quality.
8 Modifications or extensions: Discovering Fuzzy Classification Rules. FRANTIC-SRL maintains multiple populations of ants, each of them discovers rules predicting a different class (fixed for all ants belongs to one colony) [36]		
Authors	Year	Open questions:
Galea & Shen	2006	The evaluation of the rule sets could be reduced by considering only the combinations of the best rules (the delaying reinforcement (perhaps to a late) — the update of the pheromones).
9 Modifications or extensions: Discovering rules for multi-label classification. The simultaneous prediction of the value of two or more class attributes rather than just 1 class attribute [11]		
Authors	Year	Open questions:
Chan & Freitas	2006	MuLAM — each ant constructs a set of rules. MuLAM uses a pheromone matrix for each of the class attributes. Authors update the pheromone matrix of the class attributes (depending on the terms in its antecedent) predicted by the rule. The model of communication via pheromone on different levels is worth to analyze.

heuristic function stays unchanged during the algorithm run in the classical approach. We want to investigate whether the heuristic function depends on the previous well-known approaches in the data-mining area (C4.5, CN2) and can influence the behavior of the whole colony, or not. A rule for pheromone updating, which specifies how to modify the pheromone trail (τ_{ij}) and guarantee the communication between ants, is an important aspect of learning via pheromone values. Consequently the changing scheme of this value may play a significant role in establishing a suitable form of cooperation. Finally, the probabilistic transition rule based on the value of the heuristic function and on the contents of the pheromone trail matrix (that is used to iteratively construct rules) can also be analyzed as a form of specific two-way switch between deterministic and stochastic rule induction in Ant-Miner. These problems will be analyzed carefully in the following experimental studies.

6.1 Data Sets Used in Our Experiments

The evaluation of the performance behavior the behavior of Ant-Miner was performed using 5 public-domain data sets from the UCI (University of California at Irvine) data set repository available from: `http://www.ics.uci.edu.~mlearn/MLRepository.html`. Table 3 shows the main characteristics of the data sets, which were the data sets similar to ones used to evaluate the original Ant-Miner. Please note that Ant-Miner cannot cope directly with continuous attributes (i.e. continuous attributes have to be discretized in a preprocessing step, using the RSES program (`http://logic.mimuw.edu.pl/~rses/`). For the data sets marked with an asterisk in table 3 we used the discretized version of the data. In the original Ant-Miner and Galea implementation [35], the discretization was carried out using a method called C4.5-Disc [49]. C4.5-Disc is an entropy-based method that applies the decision-tree algorithm C4.5 to obtain discretization of the continuous attributes.

Both the original Ant-Miner and our proposal have some parameters. The first one — the number of ants will be examined during the experiments. The rest of parameters are presented in tab. 4. There are the following values of the parameters:

- the minimum number of cases per rule,
- the maximum number of uncovered cases in the training set,
- the number of rules used to test convergence of the ants.

In order to obtain reliable performance estimates ten-fold cross-validations were carried out to produce each of the statistics in the tables below. As some of the tests required changes to the parameters for each of the data sets, for example the number of ants or q_0-value was changed during appropriate tests. The results obtained are summarized after 300 ten-fold cross-validations, summarized and discussed below. Possible follow-up investigations suggested by the results are also mentioned.

The authors of the original version of Ant-Miner used 2 sets of evaluation criteria for the comparison with other rule induction algorithms: C4.5 and CN2. We usually used the first criterion: a predictive accuracy. The second criterion determines the rule set size:

- the number of rules in a rule set,
- an individual rule size (the number of terms per rule).

Table 3 The properties of data sets

Dataset	Number of Instances	Number of Attributes	Number of Values of Attributes	Number of Decision Classes
Breast cancer	296	9+1	51	2
Wisconsin breast cancer	699	9+1	90	2
Dermatology	366	34+1	*137	6
Hepatitis	155	19+1	*71	2
Tic-tac-toe	958	9+1	27	2

Table 4 Original parameters in data sets

Test Set	50%
Training Set	50%
Min. Cases per Rule	5
Max. Uncovered Cases	10
Rules for Convergence	10
Number of Iterations	100

6.2 TEST 1: Changing the Population Size

This test was carried out to see whether a variable number of ants as compared to a single ant in Ant-Miner has an influence on the predictive accuracy. The number of ants was changed from 1 to 50, with the step equal to 5. The aim of this experiment was to observe the obvious effect of a changing of the number of ants for an iteration and how does it affct the predictive accuracy.

Two tables: 5 and 6 show the results obtained for all of the data sets and compare them with the result obtained with a population size equal to 1. Similarly to Galea results [35], in two of the data sets — Wisconsin Breast Cancer

Table 5 Comparing the predictive accuracy for the different number of ants' population. The numbers next to the ,,±" symbol are the standard deviations of the corresponding average predictive accuracies

Dataset	Best Achieved result		Predictive Accuracy with a Population Size of one Ant (%)
	Population size (%)	Predictive Accuracy (%)	
Breast cancer	5	74.69 (± 1.76)	74.36 (± 1.69)
Wisconsin breast cancer	1	92.13 (± 1.06)	92.13 (± 1.06)
Dermatology	40	94.02 (± 1.77)	87.84 (± 3.20)
Hepatitis	50	79.17 (± 2.64)	78.79 (± 3.16)
Tic-tac-toe	1	73.80 (± 1.43)	73.80 (± 1.43)
	10	73.80 (± 1.61)	

Table 6 The simplicity of the rule sets generated by Ant-Miner, C4.5 and CN2. The numbers next to the ,,±" symbol are the standard deviations of the corresponding average predictive accuracies

Dataset	Average No. of Rules		
	Ant Miner	C4.5	CN2
Breast cancer	5.48 (± 0.36)	6.2 (± 3.21)	55.4 (± 2.07)
Wisconsin breast cancer	9.92 (± 0.62)	11.1 (± 1.45)	18.6 (± 1.45)
Dermatology	7.11 (± 0.43)	23.2 (± 1.99)	18.5 (± 0.47)
Hepatitis	4.42 (± 0.31)	4.4 (± 0.93)	7.2 (± 0.25)
Tic-tac-toe	7.90 (± 0.86)	83.0 (± 14.1)	39.7 (± 2.52)

and Tic-Tac-Toe the algorithm achieved the best average predictive accuracy for one ant only. In the rest of the analyzed data sets the improvements can be seen when the population size was increased to the greater values, close to 50. This number of ants depends on the particular characteristics of the data sets.

During these investigations, one thing to note is that choosing the best rule from all of the created within one iteration comes a superficial and purposeless task. It can be helpful only for later ants (using pheromone updates), which could create different rules that might also could have an improved quality.

It is worth to make further experiments to check the pheromone updating rules force the ants to explore different regions in the problem search space

and create different rules. This may be accomplished by testing different strategies of pheromone reinforcements or evaporations according to local and global rules. The pheromone evaporation create a possibility, or situation, when other ants in the same iteration select different terms and hence create different, more valuable rules. More detailed investigation were carried out in the next tests.

6.3 TEST 2 — Changing the Pheromone Values

There are various ways in which a pheromone updating can be performed — the initialization of the pheromone and setting it at minimum/maximum bounds of the levels reached. In the original Ant-Miner the rule induction is performed with a population of one ant. In the ant colony algorithm, where the population size is greater than one, choices need to be made as to how to update the pheromone values. All the ants may have to be reinforced or only the k best ants are used, in short, the elitist strategy should be used for pheromone updating, ensuring that exploration is more channelized. As it was mentioned before in the original Ant-Miner algorithm the pheromone values of terms were updated after each ant create the solution while in our implementation the pheromone valueis changed at the end of an iteration based on the best ant from the current iteration (a global updating rule). Meanwhile, the gathered pheromone values evaporate according to the local pheromone updating rule. This makes the pheromone updating of terms more prejudicing and therefore allows more control over the search of ants in successive iterations.

$$\Delta\tau_{ij}(t) = (1-\alpha)\tau_{ij}(t-1) + \tau_{ij}(t-1)\cdot Q, \forall i,j \in R,$$

where the parameters are the same as mentioned in the second section of this article.

Our approach also imitate the same pheromone updating rule as in the original Ant-Miner:

$$\tau_{ij}(t) = \tau_{ij}(t-1) + \tau_{ij}(t-1)\cdot Q = (1+Q)\tau_{ij}(t-1)$$

Meanwhile, the effect of evaporation for unused terms is achieved by dividing the value of each $\tau_{ij}(t)$ by the summation of all τ_{ij} (a specific normalization process).

In the case, when pheromone values were generally too low, and only few good rules were being generated and reinforced by new pheromone values, we propose another method of changing the values. This process causes that the pheromone is increased, but the rest of the terms have decreasing pheromone values. Because the results indicate that the pheromone values are not in the correct boundaries, we propose to employed the another method of normalization in changeable the scheme of pheromone values. This idea was presented

by V. Maniezzo as Approximate Nondeterministic Tree Search (ANTS) [33], where the pheromone rule is presented as follows:

$$\Delta\tau_{ij}(t) = \tau_{ij}(t-1) \cdot \left(1 - \frac{Q - LB}{avgQ - LB}\right), \forall i, j \in R,$$

where LB is the value of the lower bound on the optimal solution value computed at the start of the algorithm and we have $LB = 0.05$ and $avgQ$ is the moving average of the last $l = 10$ evaluated solutions according to the Q function; l is a parameter of the algorithm.

Another point that needs to be clarified is whether the normalization is correctly performed. We examine this procedure by changing the upper bound of the normalization (in these circumstances we establish the boundary equal to 3). We also reinforced the addition value of the Q function by multiplying this value by $(1-\alpha)$, which leads to $(1+Q)\cdot 0.9$ updating the previous values of pheromone:

$$\tau_{ij}(t) = \tau_{ij}(t-1) \cdot 0.9 + \tau_{ij}(t-1) \cdot 0.9 \cdot Q,$$

where α is the evaporation factor in the global updating rule (equal to 0.1).

The best achieved predictive accuracies are presented in the tab. 7 in bold. Unfortunately, the interesting ANTS approach is not as promising as we expected. To sum up all the algorithm changes and developments we can

Table 7 The influence of the pheromone updates on the predictive accuracy (%). The numbers below, next tor the „±" symbol are the standard deviations of the corresponding average predictive accuracies

Dataset	Standard update	Normalize 0-3	Increase c. 0.9	Minus c. 0.01	ANTS
Breast cancer	74.69 (± 1.76)	72.99 (± 2.06)	73.93 (± 2.46)	**74.14** (± 1.86)	71.97 (± 2.88)
Wisconsin breast cancer	92.13 (± 1.06)	**91.98** (± 1.10)	91.58 (± 1.39)	91.93 (± 1.59)	91.57 (± 1.35)
Dermatology	94.02 (± 1.77)	93.27 (± 1.64)	**93.96** (± 1.41)	92.71 (± 2.18)	88.32 (± 2.42)
Hepatitis	79.17 (± 2.64)	79.58 (± 2.80)	79.22 (± 3.11)	**79.82** (± 1.92)	76.98 (± 3.26)
Tic-tac-toe (1)	73.80 (± 1.43)	**74.26** (± 1.87)	73.31 (± 1.02)	74.23 (± 2.24)	71.80 (± 1.97)
Tic-tac-toe	73.80 (± 1.61)	72.46 (± 1.96)	72.71 (± 1.11)	**73.17** (± 1.41)	71.29 (± 1.69)

observed that the pheromone values for some term in some of the data sets were getting extraordinarily low. Several solutions were possible to include in our approach:

- setting a minimum value for the pheromone values of terms, similarly to the Min-Max approach proposed by Stützle and Hoos [76],
- ignore the terms with small, uninteresting values, i.e. do not consider them for inclusion in the currently created rule.

6.4 TEST 3 — Changing the Main Transition Rule

In this experimental study we want to see whether altering the main transition rule (selection) from the random proportional selection to the pseudo random proportional selection has an effect on the predictive accuracy. A pseudo random proportional selection requires a setting of the q parameter value that enables the exploitation or the exploration of the problem search space.

Different values of q were tested, ranging from 0.1 to 1.0. A setting of $q = 0.0$ means no exploitation of owned knowledge and is the same as using a random proportional rule. A setting $q = 1.0$ turns the algorithm into a deterministic approach and a term with a higher probability is always chosen.

The following tab. 8 shows the results for the different q_0-values settings.

Table 8 Comparing predictive accuracy for different q_0 values

Dataset	Expl./Explor. Rate (q_0)	Predictive Accuracy (%)	Exploration Accuracy (%)
Breast cancer	0.9	**74.71** (± 1.79)	74.69 (± 1.76)
Wisconsin breast cancer	0.6	**92.69** (± 1.16)	92.13 (± 1.06)
Dermatology	0.0 (s.)	94.02 (± 1.77)	94.02 (± 1.77)
Hepatitis	1.0 (d.)	**80.31** (± 2.79)	79.17 (± 2.64)
Tic-tac-toe (1 ant)	0.9	**74.87** (± 1.82)	73.80 (± 1.43)
Tic-tac-toe	0.9	**74.33** (± 1.94)	73.80 (± 1.61)

It is especially interesting to note that for the Dermatology data set the stochastic algorithm achieved the best predictive accuracy. On contrary, in the Hepatis data set we observe the best performance for the fully deterministic approach.

It can be an intriguing aspect of future research to adjust specific features of data sets to the nature of this stochastic/deterministic approach. In order to achieve better predictive accuracy than in the standard C4.5 and CN2

we can combine this study with the population size examination. Another question is the dynamic or adaptive scheme of changing the q_0 value. This possibility enable the employment of coarse-grained or fine-grained methodology during the searching process.

6.5 TEST 4 — Changing the Q Values (Modifications of Pruning)

The rule pruning procedure may be slightly different from Ant-Miner's implementation. We have introduced three extensions to the classical approach.

Accuracy Convergence. It is a compromise between accuracy and convergence:

$$Q = \frac{TP}{TP+FP} \cdot \frac{TP+FP}{TP+FP+TN+FN}$$

$$Q = \frac{TP}{TP+FP+TN+FN}$$

Laplace Accuracy. This evaluation function is based on the Laplace error estimate (we explain it using the standard notation, first used in Q function):

$$Q = \frac{TP+1}{TP+FP+k}$$

where k — the number of decision cases.

Elite 0.2. This formula is derived from the elitist strategy commonly applied in many other combinatorial optimization problems. We have analyzed 20% of the population size (when this size is greater or equal to 10).

In the case of Ant-Miner and rule induction we search a fitness function that assesses how well a rule constructed by an ant fulfills our expectations.

From the study summarized in the tab. 9, we see that the Laplace accuracy is used successfully in almost every type of data sets (with an exception of the Breast cancer). It should be emphasized that in this type of modification the number of rules increases regularly in all the analyzed data sets (see tab. 10). We also observe a smaller value of accuracy in the first column for the Dermatology data set. As in classification, the question arises as to whether the loss of accuracy is due to the wrong method of pruning and consequently the rule constructed or to the increased difficulty in classifying cases in this data set. We achieved the best results for the Tic-Tac-Toe data set with a pruning procedure utilizing the Laplace error estimate.

6.6 TEST 5 — Changing the Heuristic Function

According to the proposition concerning the heuristic function [52], we also analyze the simplicity of this part of the main transition rule in Ant-Miner.

Table 9 Modifications of pruning

Dataset	Accuracy Convergence (%)	Laplace Accuracy (%)	Elite 0.2 (%)	Standard Q (%)
Breast cancer	70.73 (± 2.14)	65.15 (± 2.60)	—	**74.69** (± 1.76)
Wisconsin breast cancer	79.27 (± 1.26)	**93.95** (± 1.02)	—	92.13 (± 1.06)
Dermatology	47.49 (± 2.50)	91.97 (± 2.02)	—	**94.02** (± 1.77)
Hepatitis	**79.82** (± 2.93)	76.46 (± 2.17)	79.03 (± 3.70)	79.17 (± 2.64)
Tic-tac-toe (1)	69.71 (± 1.29)	**97.77** (± 1.31)	—	73.80 (± 1.43)
Tic-tac-toe	70.10 (± 0.95)	**99.80** (± 0.19)	73.42 (± 1.84)	73.80 (± 1.61)

Table 10 Simplicity of rule sets generated in the context of Q

Dataset	Average No. of Rules			
	Accuracy Convergence	Laplace Accuracy	Elite 0.2	Standard Q
Breast cancer	3.24 (± 0.24)	18.64 (± 1.00)	—	5.40 (± 0.46)
Wisconsin breast cancer	5.98 (± 0.38)	14.06 (± 1.06)	—	9.82 (± 0.50)
Dermatology	2.96 (± 0.08)	9.32 (± 0.88)	—	7.02 (± 0.38)
Hepatitis	2.54 (± 0.34)	6.66 (± 0.74)	4.44 (± 0.28)	4.42 (± 0.34)
Tic-tac-toe (1)	4.10 (± 0.10)	19.64 (± 2.64)	—	6.94 (± 0.78)
Tic-tac-toe	4.08 (± 0.08)	14.24 (± 0.56)	7.94 (± 0.98)	8.18 (± 0.70)

The motivation is as follows: in ACO approaches we do not need sophisticated information in the heuristic function, because of the pheromone value, which compensate some mistakes in term selections. Our intention is to explore the effect of using a simpler heuristic function instead of a complex one, originally proposed by Parpinelli [66].

Only pheromone. We propose to investigate a simple ant algorithm without the heuristic function. The pheromone information should govern the behavior of agents.

$$P_{ij}(t) = \frac{\tau_{ij}(t)}{\sum_{i}^{a} x_i \sum_{j}^{b_i} \tau_{ij}(t)}.$$

Density η_{ij} I. This proposition is derived from the density measure:

$$\eta_{ij} = \frac{max(d_{ij})}{\sum_{k}^{d_{ij}} d_{ijk}},$$

where $max(d_{ij})$ is the biggest number of objects belonging to the decision class for a specific term T_{ij}, and $\sum_{k}^{d_{ij}} d_{ijk}$, is the sum of all objects from all decision classes concerning this term.

Density η_{ij} II. This version of the density measure is another interpretation of the proposition of using a simpler version of the heuristic function:

$$\eta_{ij} = \frac{max(d_{ij})}{\sum_{l}^{a} x_l \sum_{m}^{b_m} \sum_{k}^{d_{lm}} d_{lmk}},$$

where $max(d_{ij})$ is similar to the majority class of T_{ij}, and a denominator $\sum_{l}^{a} x_l \sum_{m}^{b_m} \sum_{k}^{d_{lm}} d_{lmk}$ describes all the objects belonging to every decision class for all terms used in the examined rule. We consider it the same rule as proposed in [52], i.e.:

$$\eta_{ij} = \frac{max(d_{ij})}{|T_{ij}|}.$$

This new notation should help to better understand the differences in our two density modifications.

Table 11 shows the accuracy rates for rule sets produced in different approaches. It can be seen that in general these modifications are similar to the original Ant-Miner in the context of effectiveness. There are two reasons of disparity in the comparison between the presented results and the proposal

Table 11 Modifications of the heuristic function — predictive accuracy

Dataset	Only pheromone (%)	Density η_{ij} I (%)	Density η_{ij} II (%)	Standard function (%)
Breast cancer	71.23 (± 2.95)	72.20 (± 2.97)	72.85 (± 2.28)	74.69 (± 1.76)
Wisconsin breast cancer	91.39 (± 0.96)	91.56 (± 0.96)	92.05 (± 1.08)	92.13 (± 1.06)
Dermatology	92.72 (± 1.90)	93.63 (± 1.83)	93.98 (± 1.48)	94.02 (± 1.77)
Hepatitis	75.89 (± 2.81)	77.16 (± 2.75)	75.95 (± 3.07)	79.17 (± 2.64)
Tic-tac-toe (1)	71.21 (± 1.67)	72.23 (± 1.68)	71.93 (± 1.54)	73.80 (± 1.43)
Tic-tac-toe	72.36 (± 2.00)	72.22 (± 1.57)	71.81 (± 1.99)	73.80 (± 1.61)

in [52]. Firstly, more ants should be employed in this version of the density oriented heuristic function. Secondly, this version requires more running time for a reliable comparison.

We can conclude that looking for the appropriate scheme of changing the analyzed functions and rules independently was the wrong way. A more satisfactory procedure was found that consisted of two key elements:

- tunning the procedure of effective pruning with simultaneous changing the pheromone matrix,
- simplifying the main transition rule by incorporating a new heuristic function based on density measures with a learning procedure via pheromone values precisely performed.

7 Conclusions

Simple ants, following very simple rules, interact with each other and represent an example of swarm intelligence behavior. As we know: ,, the whole is more than the sum of the parts", so this kind of intelligence is an example of an emergent phenomenon. The employment of the ant colony metaphor for discovering rules, classification is a very underestimated research area. In other research areas the ACO algorithms have been presented as an effective approach, which produce good solutions for different combinatorial optimization problems.

The important advantage in the context of data mining and rule induction is that Ant-Miner produces rule sets much smaller than the rule sets created

by the classical approaches: C4.5 and CN2. In this chapter new modifications based on the ant colony metaphor were incorporated in the original Ant-Miner rule producer. Compared to the previous implementations and settings of Ant-Miner, these different experimental studies show how these extensions improve, or sometimes deteriorate, the performance of Ant-Miner. We presented the efficiency of our Ant-Miner was presented via this comparative study.

Examining the previous modifications and extensions (first applied in the combinatorial optimization field) to find appropriate and satisfying decision rules in analyzed data sets was an interesting directions of research.

This approach is the result of a more detailed and deepened review of the previous approaches. We observed that these modifications analyzed separately tend not to be sufficiently motivated. There is still room for improvement in two directions. Firstly, it is still not clear wheather this approach consistently improves its efficiency in maintaining multiple colonies of ants. Secondly, this algorithm is not fully examined in the version without pruning procedure. We also plan to examine this method when used for rule induction in the bigger data sets.

References

1. Corne, D., et al.: New Ideas in Optimization. Mc Graw-Hill, Cambridge (1999)
2. Bauer, A., Bullnheimer, B., Hartl, R.F., Strauss, C.: An Ant Colony Optimization approach for the single machine total tardiness problem. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 1445–1450. IEEE Press, Piscataway (1999)
3. Boffey, B.: Multiobjective routing problems. Top 3(2), 167–220 (1995)
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence. In: From Natural to Artificial Systems. Oxford University Press, Oxford (1999)
5. Bonabeau, E., Henaux, F., Guérin, S., Snyers, D., Kuntz, P., Théraulaz, G.: Routing in telecommunication networks with "Smart" ant–like agents telecommunication applications. Springer, Heidelberg (1998)
6. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees, Belmont C.A., Wadsworth (1984)
7. Bullnheimer, B., Hartl, R.F., Strauss, C.: An improved Ant System algorithm for the Vehicle Routing Problem. Technical Report POM–10/97, Institute of Management Science, University of Vienna (1997)
8. Bullnheimer, B., Hartl, R.F., Strauss, C., Bullnheimer, B., Hartl, R.F., Strauss, C.: A new rankbased version of the Ant System: A computational study. Technical Report POM–03/97, Institute of Management Science, University of Vienna (1997)
9. Bullnheimer, B., Hartl, R.F., Strauss, C.: Applying the Ant System to the Vehicle Routing Problem. In: Martello, S., Osman, I.H., Voß, S., Martello, S., Roucairoll, C. (eds.) MetaHeuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 109–120. Kluwer Academics, Dordrecht (1998)
10. Bullnheimer, B., Strauss, C., Bullnheimer, B., Hartl, R.F., Strauss, C.: Instituts für Betriebwirtschaftslehre, Universität Wien (1996)

11. Chan, A., Freitas, A.A.: A new ant colony algorithm for multi-label alssification with applications in bioinformatics. In: Proceedings of Genetic and Evolutionary Computation Conf (GECCO 2006), San Francisco, pp. 27–34 (2006)
12. Chen, C., Chen, Y., He, J.: Neural network ensemble based ant colony classification rule mining. In: Proceedings of First Int. Conf. Innovative Computing, Information and Control (ICICIC 2006), pp. 427–430 (2006)
13. Chen, Z.: Data Mining and uncertain reasoning. An integrated approach. John Wiley and Sons, Chichester (2001)
14. Clark, P., Boswell, R.: Rule induction with CN2: some recent improvements. In: Kodratoff, Y. (ed.) EWSL 1991. LNCS (LNAI), vol. 482, pp. 151–163. Springer, Heidelberg (1991)
15. Clark, P., Niblett, T.: The CN2 rule Induction algorithm. Machine Learning 3(4), 261–283 (1989)
16. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: Vavala, F., Bourgine, P. (eds.) Proceedings First Europ. Conference on Artificial Life, pp. 134–142. MIT Press, Cambridge (1991)
17. Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M.: Ant system for job–shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL) 34, 39–53 (1994)
18. Costa, D., Hertz, A.: Ants can colour graphs. Journal of the Operational Research Society 48, 295–305 (1997)
19. Den Besten, M., Stützle, T., Dorigo, M.: Scheduling single machines by ants. Technical Report 99–16, IRIDIA, Université Libre de Bruxelles, Belgium (1999)
20. Deneubourg, J.–. L., Goss, S., Franks, N.R., Pasteels, J.M.: The Blind Leading the Blind: Modelling Chemically Mediated Army Ant Raid Patterns. Insect Behaviour 2, 719–725 (1989)
21. DiCaro, G., Dorigo, M.: AntNet: A mobile agents approach to adaptive routing. Technical report, IRIDIA, Université Libre de Bruxelles (1998)
22. DiCaro, G., Dorigo, M.: AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research (JAIR) 9, 317–365 (1998)
23. DiCaro, G., Dorigo, M.: Extending AntNet for best–effort Quality–of–Service routing. In: ANTS 1998 – From Ant Colonies to Artificial Ants: First International Workshop on Ant Colony Optimization, October 15–16 (1998) (Unpublished presentation)
24. DiCaro, G., Dorigo, M.: Two ant colony algorithms for best–effort routing in datagram networks. In: Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 1998), pp. 541–546. IASTED/ACTA Press (1998)
25. Dorigo, M.: Optimization, Learning and Natural Algorithms (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, IT (1992)
26. Dorigo, M., DiCaro, G.: The ant colony optimization meta–heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization. McGraw–Hill, London (1999)
27. Dorigo, M., DiCaro, G., Gambardella, L.: Ant algorithms for distributed discrete optimization. Artif. Life 5(2), 137–172 (1999)
28. Dorigo, M., Gambardella, L.: A Study of Some Properties of Ant–Q. In: Proceedings of Fourth International Conference on Parallel Problem Solving from Nature, PPSNIV, pp. 656–665. Springer, Berlin (1996)
29. Dorigo, M., Gambardella, L.: Ant Colonies for the Traveling Salesman Problem. Biosystems 43, 73–81 (1997)

30. Dorigo, M., Gambardella, L.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Trans. Evol. Comp. 1, 53–66 (1997)
31. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Technical Report 91–016, Politechnico di Milano, Italy (1991)
32. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Trans. Syst. Man. Cybern. B26, 29–41 (1996)
33. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
34. Freitas, A.A., Johnson, C.G.: Research cluster in swarm intelligence. Technical Report EPSRC Research Proposal GR/S63274/01 — Case for Support, Computing Laboratory, Computing Laboratory, Laboratory of Kent, Kent (2003)
35. Galea, M.: Applying swarm intelligence to rule induction. MS thesis, University of Edingbourgh (2002)
36. Galea, M., Shen, Q.: Simultaneous ant colony optimization algorithms for learning linguistic fuzzy rules. In: Agraham, A., Grosan, C., Ramos, V. (eds.) Swarm Intelligence in Data Mining. Springer, Berlin (2006)
37. Gambardella, L.M., Dorigo, M.: AntQ.Ant–Q. A Reinforcement Learning Approach to the Traveling Salesman Problem. In: Proceedings of Twelfth International Conference on Machine Learning, pp. 252–260. Morgan Kaufman, Palo Alto (1995)
38. Gambardella, L.M., Dorigo, M.: Solving symmetric and asymmetric TSPs by ant colonies. In: Proceedings of the IEEE Conference on Evolutionary Computation, ICEC 1996, pp. 622–627. IEEE Press, Los Alamitos (1996)
39. Gambardella, L.M., Dorigo, M.: HAS–SOP: Hybrid Ant System for the Sequential Ordering Problem. Technical Report 11, IDSIA Lugano (1997)
40. Gambardella, L.M., Taillard, E., Agazzi, G.: MACS–VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. Technical Report 06–99, IDSIA, Lugano, Switzerland (1999)
41. Gambardella, L.M., Taillard, E.D., Dorigo, M.: Ant colonies for the QAP. Technical Report 4–97, IDSIA, Lugano, Switzerland (1997)
42. Gambardella, L.M., Taillard, E.D., Dorigo, M.: Ant colonies for the QAP. Journal of the Operational Research Society (JORS) 50(2), 167–176 (1999)
43. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1997)
44. Goss, S., Beckers, R., Denebourg, J.L., Aron, S., et al.: How Trail Laying and Trail Following Can Solve Foraging Problems for Ant Colonies. In: Hughes, R.N. (ed.) Behavioural Mechanisms for Food Selection, vol. G20. Springer, Berlin (1990)
45. Grasse, P.-P.: La Reconstruction du Nid et les Coordinations Inter–Individuelles chez Bellicositermes Natalensis et Cubitermes sp. La Theorie de La Stigmerie. Insects Soc. 6, 41–80 (1959)
46. Grasse, P.-P.: Termitologia, vol. II, Paris, Masson (1984)
47. Heusse, M., Guérin, S., Snyers, D., Kuntz, P.: Adaptive agent–driven routing and load balancing in communication networks. Technical Report RR–98001–IASC, Départment Intelligence Artificielle et Sciences Cognitives, ENST Bretagne, ENST Bretagne (1998)
48. Smaldon, J., Freitas, A.A.: A new version of the Ant-Miner algorithm discovering unordered rule sets. In: Proceedings of Genetic and Evolutionary Computation Conf (GECCO 2006), San Francisco, pp. 43–50 (2006)

49. Kohavi, R., Sahami, M.: Error-based and entropy-based discretization of continuous features. In: Proc. 2nd Intern. Conference Knowledge Discovery and Data Mining, pp. 114–119 (1996)
50. Leguizamón, G., Michalewicz, Z.: A new version of Ant System for subset problems. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 1459–1464. IEEE Press, Piscataway (1999)
51. Liang, Y.–C., Smith, A.E.: An Ant System approach to redundancy allocation. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 1478–1484. IEEE Press, Piscataway (1999)
52. Liu, B., Abbas, H.A., Mc Kay, B.: Classification rule discovery with ant colony optimization. IEEE Computational Intelligence Bulletin 1(3), 31–35 (2004)
53. Ramalhinho Lourenço, H., Serra, D.: Adaptive approach heuristics for the generalized assignment problem. Technical Report EWP Series No. 304, Department of Economics and Management, Universitat Pompeu Fabra, Barcelona (1998)
54. Maniezzo, V.: Exact and approximate nondeterministic tree–search procedures for the quadratic assignment problem. Technical Report CSR 98–1, C. L. In: Scienze dellInformazione, Universita di Bologna, sede di Cesena, Italy (1998)
55. Maniezzo, V., Carbonaro, A.: An ANTS heuristic for the frequency assignment problem. Technical Report CSR 98–4, Scienze dell Informazione, Universita di Bologna, Sede di Cesena, Italy (1998)
56. Maniezzo, V., Colorni, A.: The Ant System applied to the Quadratic Assignment Problem. IEEE Trans. Knowledge and Data Engineering (1999)
57. Maniezzo, V., Colorni, A.: An ANTS heuristic for the frequency assignment problem. Future Generation Computer Systems 16, 927–935 (2000)
58. Maniezzo, V., Colorni, A., Dorigo, M.: The Ant System applied to the Quadratic Assignment Problem. Technical Report 94–28, IRIDIA, Université Libre de Bruxelles, Belgium (1994)
59. Martens, D., De Backer, M., Haesen, R., Baesens, B., Holvoet, T.: Ants constructing rule-based classifiers. In: Agraham, A., Grosan, C., Ramos, V. (eds.) Swarm Intelligence in Data Mining. Springer, Berlin
60. Michalski, R., Mozetic, J., Hong, J., Lavrac, N.: The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In: AAAI 1986, vol. 2, pp. 1041–1045 (1987)
61. Michel, R., Middendorf, M.: An island model based Ant System with lookahead for the Shortest Supersequence Problem. In: Eiben, A.E., Back, T., Schoenauer, M., Schwefel, H.–P. (eds.) Proceedings of PPSN–V, Fifth International Conference on Parallel Problem Solving from Nature, pp. 692–701. Springer, Heidelberg (1998)
62. Michel, R., Middendorf, M.: An ACO algorithm for the Shortest Common Supersequence Problem. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Methods in Optimisation. McGraw-Hill, New York (1999)
63. Oakes, M.P.: Ant colony optimization for stylometry: the federalist papers. In: Proceedings of Recent Advances in Soft Computing (RASC 2004), pp. 86–91 (2004)
64. Osman, I., Laporte, G.: Metaheuristics: A bibliography. Annals of Operations Research 63, 513–623
65. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: An ant colony algorithm for classification rule discovery. In: Abbas, H., Sarker, R., Newton, C. (eds.) Data Mining: a Heuristic Approach. Idea Group Publishing, London (2002)

66. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation, Special issue on Ant Colony Algorithms 6(4), 321–332 (2004)
67. Quinlan, J.R.: Introduction of decision trees. Machine Learning 1, 81–106 (1986)
68. Quinlan, J.R.: Generating production rules from decision trees. In: Proc. of the Tenth International Joint Conference on Artificial Intelligence, pp. 304–307. Morgan Kaufmann, San Francisco (1987)
69. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
70. Reeves, C.: Modern Heuristic Techniques for Combinatorial Problems. In: Advanced Topics in Computer Science. McGrawHill, London (1995)
71. Schoonderwoerd, R., Holland, O., Bruten, J.: Ant–like agents for load balancing in telecommunications networks. In: Proceedings of the First International Conference on Autonomous Agents, pp. 209–216. ACM Press, New York (1997)
72. Schoonderwoerd, R., Holland, O., Bruten, J., Rothkrantz, L.: Ant–based load balancing in telecommunications networks. Adaptive Behavior 5(2), 169–207 (1996)
73. Stützle, T.: An ant approach to the Flow Shop Problem. Technical Report AIDA–97–07, FG Intellektik, FB Informatik, TH Darmstadt (September 1997)
74. Stützle, T., Hoos: Improvements on the Ant System: Introducing MAX–MIN Ant System. In: Improvements on the Ant System: Introducing MAX–MIN Ant System Algorithms, pp. 245–249. Springer, Heidelberg (1997)
75. Stützle, T., Hoos: The MAX–MIN Ant System and Local Search for the Traveling Salesman Problem. In: Baeck, T., Michalewicz, Z., Yao, X. (eds.) Proceedings of IEEE–ICEC–EPS 1997, IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference, pp. 309–314. IEEE Press, Los Alamitos (1997)
76. Stützle, T., Hoos: MAX–MIN Ant System and Local Search for Combinatorial Optimisation Problems. In: Proceedings of the Second International conference on Metaheuristics MIC 1997, Kluwer Academic, Dordrecht (1998)
77. Subramanian, D., Druschel, P., Chen, J.: Ants and Reinforcement Learning: A case study in routing in dynamic networks. In: Proceedings of IJCAI 1997, International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco (1997)
78. van der Put, R.: Routing in the faxfactory using mobile agents. Technical Report R&D–SV–98–276, KPN Research (1998)
79. Navarro Varela, G., Sinclair, M.C.: Ant Colony Optimisation for virtual–wavelength–path routing and wavelength allocation. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 1809–1816. IEEE Press, Piscataway (1999)
80. Wang, Z., Feng, B.: Classification rule mining with an improved ant colony algorithm. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 357–367. Springer, Heidelberg (2004)
81. White, T., Pagurek, B., Oppacher, F.: Connection management using adaptive mobile agents. In: Arabnia, H.R. (ed.) Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 1998), pp. 802–809. CSREA Press,