Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Wise Mining Method through Ant Colony Optimization

Yang Jianxiong and Junzo Watada
Graduate School of Information, Production and Systems
Waseda University
2-7 Hibikino, Wakamatsu, Kitakyushu 808-0135 JAPAN
e-mail: leoworldplus@yahoo.co.jp, junzow@osb.att.ne.jp

Abstract— This paper proposes an algorithm for data mining named Pheromone-Miner (ant-colony-based data miner). The algorithm is inspired by both researches on the behavior of real ant colonies and data mining concepts as well as principles. The goal of Pheromone-Miner is to extract more exact knowledge from a database. Pheromone-based mining breaks through limitations of other mining approaches. We compare the performance of pheromone-miner with a general semantic miner. The accident causes discovered by ant-miner are considerably more accurate than those discovered by a general semantic miner. In a word, this evolutionary algorithm is suitable for improving the accuracy of data miners.

Keywords— **ant colony optimization algorithm, pheromone, data mining, knowledge discovery**

I. Introduction

The goal of data mining is to extract knowledge from data. Data mining is an interdisciplinary field, whose core is at the intersection of machine learning, statistics, and databases.

We emphasize that in data mining—unlike, e.g., classical statistics—the goal is to discover knowledge that is not only accurate, but also comprehensible for the user [4], [5], [14]. Comprehensibility is important whenever discovered knowledge will be used for supporting a human decision. After all, if discovered knowledge is not comprehensible for a user, it will not be possible to interpret and validate the knowledge. In this case, probably the user will not have sufficient trust in the discovered knowledge. In decision making, this can lead to incorrect decisions.

There are several data mining tasks, including classification, regression, clustering, dependence modeling, etc. [4]. Each of these tasks is regarded as a kind of problem which can be solved by a data mining algorithm. Therefore, the first step in designing a data mining algorithm is to define which task the algorithm will address.

In this paper, we propose an ant colony optimization (ACO) algorithm [2], [3] for the classification task of data mining. In this task, the goal is to assign each case (object, record, or instance) to one class, out of a set of predefined classes, based on the values of some attributes (called predictor attributes) for the case.

In the context of the classification task of data mining, discovered knowledge is often expressed in the form of IF–THEN rules, as follows:

IF <conditions> THEN <class>

The rule antecedent (**IF part**) contains a set of conditions, usually connected by a logical conjunction operator (**AND**). We will refer to each rule condition is a **term**(class), so that the rule antecedent is a logical conjunction of terms in the form **IF term 1 AND term 2 AND …** . Each term is denoted by a triple tupplet **<attribute, operator, value>**.

The rule consequent (**THEN part**) specifies the class predicted for cases whose predictor attributes satisfy all the terms specified in the rule antecedent. From a data-mining viewpoint, this kind of knowledge representation has the advantage of being intuitively comprehensible for the user, as long as the number of discovered rules and the number of terms in rule antecedents are not large.

It is a research area still unexplored to use ACO algorithms [1]–[3] for discovering classification rules in discovering the best knowledge in mining. Actually, the ant algorithm is developed for clustering [6] in this paper to mine data , which is a very special data-mining activator.

We believe that the development of ACO algorithms for data mining is a promising research area. ACO algorithms involve simple agents(ants) that cooperate with one another to achieve an emergent unified behavior for the system as a whole, producing a robust system which is capable of finding high-quality solutions for problems with a large search space. In the context of rule discovery, an ACO algorithm has the ability to perform a flexible robust search for a good combination of terms (logical conditions) involving values of the predictor attributes.

II. Social insects and real ant colonies

In a colony of social insects, such as ants, bees, wasps, and termites, each insect usually performs its own tasks independently from other members of the colony. However, the tasks performed by different insects are related to each other in such a way that the colony, as a whole, is capable of solving complex problems through cooperation [7]. Important survival tasks such as selecting and picking up materials, and finding and storing food, which require sophisticated planning, are solved by insect colonies without any kind of supervisor or centralized controller. This collective behavior which emerges from a group of social insects has been called "swarm intelligence" [7].

Here, we are interested in a particular behavior of real ants, namely, the fact that they are capable of finding the shortest path between a food source and the nest (adapting to changes in the environment) without the use of visual information [1]. This interesting ability of almost-blind ants has been studied extensively by ethologists. They discovered that, in order to exchange information about which path should be followed, ants communicate with one another by means of pheromone (a chemical substance) trails. As ants move, a certain amount of pheromone is dropped on the ground, marking the path with a trail of this substance. The more ants follow a given trail, the more attractive the frequent following makes this trail. This process can be described as a loop of positive feedback, in which the probability that an ant chooses a path is proportional to the number of ants that have already passed by that path [1], [2], [8].

When an established path between a food source and the ants' nest is disturbed by the presence of an object, ants soon try to go around the obstacle. First, each ant can choose to go around to the left or to the right of the object with a 50%–50% probability distribution. All ants move roughly at the same speed and deposit pheromone on the trail at roughly the same rate. Therefore, the ants that (by chance) go around the obstacle by the shortest path will reach the original track faster than the others that have followed longer paths to the obstacle. As a result, pheromone accumulates faster in the shorter path around the obstacle. Since ants prefer to follow trails with larger amounts of pheromone, eventually all the ants converge to the shorter path.

III. Ant Colony Optimization

An ACO algorithm is essentially a system based on agents that simulate the natural behavior of ants, including mechanisms of cooperation and adaptation. In [2], the use of this kind of system as a new meta-heuristic was proposed in order to solve combinatorial optimization problems. This new meta-heuristic has shown both robust and versatile—in the sense that it has been applied successfully to a range of different combinatorial optimization problems [3].

ACO algorithms are based on the following ideas.

1) Each path followed by an ant is associated with a candidate solution for a given problem.

2) When an ant follows a path, the amount of pheromone deposited on that path is proportional to the quality of the corresponding candidate solution for the target problem.

3) When an ant has to choose between two or more paths, the ant chooses the path(s) with a larger amount of pheromone with a greater probability.

As a result, the ants eventually converge to a short path, which is hopefully an optimum or near-optimum solution for the target problem, as explained before for the case of natural ants.

In essence, the design of an ACO algorithm involves the specification of [7]:

1) An appropriate representation of the problem, which allows the ants to incrementally construct/modify solutions through the use of a probabilistic transition rule, based on the amount of pheromone in the trail and on a local problem-dependent heuristic;

2) A method to enforce the construction of valid solutions, i.e., solutions that are legal in the real-world situation corresponding to the problem definition;

3) A problem-dependent heuristic evaluation **function( )** that measures the quality of items that can be added to the current partial solution;

4) A rule for pheromone updating, which specifies how to modify the pheromone **trail( )**;

5) A probabilistic transition rule based on the value of the heuristic **function( )** and on the contents of the pheromone **trail( )** that is used to iteratively construct a solution.

Artificial ants have several characteristics similar to real ants, namely:

1) Artificial ants have a probabilistic preference for paths with a larger amount of pheromone;

2) Shorter paths tend to have larger rates of growth in their amount of pheromone;

3) The ants use an indirect communication system based on the amount of pheromone deposited on each path.

IV. Ant-Miner: A new ACO Algorithm for data mining

In this section, we discuss in detail our proposed ACO algorithm for the discovery of classification rules, called Ant-Miner. The section is divided into five subsections, namely, a general description of Ant-Miner, heuristic function, rule pruning, pheromone updating, and the use of the discovered rules for classifying new cases.

A. General Description of Ant-Miner

In an ACO algorithm, each ant incrementally constructs/modifies a solution for the target problem. In our case, the target problem is the discovery of classification rules. As discussed in the introduction, each classification rule has the form

IF <term 1 AND term 2 AND …> THEN <class>

Each term is a triple tupplet **<attribute, operator, value>**, where **value** is a value belonging to the domain of **attribute**. The operator element in the triple tupplet is a relational operator. The current version of Ant-Miner copes only with categorical attributes, so that the operator element in the triple tupplet is always "=" Continuous (real-valued) attributes are discretized in a preprocessing step.

A high-level description of Ant-Miner is shown in **Algorithm I**. Ant-Miner follows a sequential covering approach to discover a list of classification rules covering all, or almost all, the training cases. At first, the list of discovered rules is empty and the training set consists of all the training cases. Each iteration of the **WHILE loop** of Algorithm I, corresponding to a number of executions of the **REPEAT-UNTIL** loop, discovers one classification rule. This rule is added to the list of discovered rules and the training cases that are covered correctly by this rule (i.e., cases satisfying the rule antecedent and having the class predicted by the rule consequent) are removed from the training set. This process is performed iteratively while the number of uncovered training cases is greater than a user-specified threshold, called Max_uncovered_cases.

Each iteration of the REPEAT-UNTIL loop of Algorithm I consists of three steps, comprising rule construction, rule pruning, and pheromone updating, detailed as follows.

First, starts with an empty rule, i.e., a rule with no term in its antecedent, and adds one term at a time to its current partial rule. The current partial rule constructed by an ant corresponds to the current partial path followed by that ant. Similarly, the choice of a term to be added to the current partial rule corresponds to the choice of the direction in which the current path will be extended. The choice of the term to be added to the current partial rule depends on the problem-dependent heuristic evaluation **function( )** and the amount of **pheromone( )** associated with each term. Let us discuss this in detail in the next sections. **Ant$_t$** keeps adding one term at a time to its current partial rule until one of the following two stopping criteria is satisfied.

1) Any term to be added to the rule makes the rule cover a number of cases that is smaller than a user-specified threshold, called Min_cases_per_rule(minimum number of cases covered per rule).

2) All attributes have already been used by the ant, so that there are no more attributes to be added to the rule antecedent. Note that each attribute can occur only once in each rule, to avoid invalid rules such as "IF (sex = male) AND (Sex = female)."

ALGORITHM I: A High-Level Description of Ant-Miner

```
TrainingSet = {all training cases};
DiscoveredRuleList = []; /* rule list is initialized with an emptylist */
WHILE (TrainingSet > Max uncovered cases)
t = 1; /* ant index */
j = 1; /* convergence test index */
Initialize all trails with the same amount of pheromone;
REPEAT

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;
Prune rule R_t;

Update the pheromone of all trails by increasing pheromone in the trail followed by Ant_t (proportional to the quality of R_t ) and decreasing pheromone in the other trails (simulating pheromone evaporation);

IF (R_t is equal to R_t-1) /* update convergence test */
THEN j = j + 1;
ELSE j = 1;
END IF
t = t + 1;
UNTIL (t>=No_of_ants) OR (j>=No_rules_converge)
Choose the best rule R_best among all rules R_t constructed by all the ants;
Add rule R_best to DiscoveredRuleList;
TrainingSet = TrainingSet - {set of cases correctly covered by R_best};
END WHILE
```

Second, rule **R$_t$** constructed by **Ant$_t$** is pruned in order to remove irrelevant terms, as discussed later. For the moment, we only mention that these irrelevant terms may also have been included in the rule due to stochastic variations in the term selection procedure and/or due to the use of a shortsighted, local heuristic function, which considers only one attribute at a time, ignoring attribute interactions.

Third, the amount of pheromone in each trail is updated, increasing the pheromone in the trail followed by **Ant$_t$** (according to the quality of rule **R$_t$**) and decreasing the pheromone in the other trails (simulating the pheromone volatilization). Then another ant starts to construct its rule, using the new amounts of pheromone to guide its search. This process is repeated until one of the following two conditions is satisfied.

1) The number of constructed rules is equal to or greater than the user-specified threshold **No_of_ants** (the number of ants).

2) The current **Ant$_t$** has constructed a rule that is exactly the same as the rule constructed by the previous **No_rules_converge – 1** ants, where **No_rules_converge (**the number of convergence ants**)** stands for the number of rules used to test convergence of the ants.

Once the **REPEAT-UNTIL** loop is completed, the best rule among the rules constructed by all ants is added to the list of discovered rules, as mentioned earlier, and the system starts a new iteration of the **WHILE loop** by reinitializing all trails with the same amount of pheromone.

It should be noted that, in a standard definition of ACO[2], a population is defined as the set of ants that build solutions between two pheromone updates. According to this definition, in each iteration of the **WHILE loop**, Ant-Miner works with a population of a single ant, since pheromone is updated after a rule is constructed by an ant. Therefore, strictly speaking, each iteration of the **WHILE loop** of Ant-Miner has a single ant that performs many iterations. Note that different iterations of the **WHILE loop** correspond to different populations, since each population's ant tackles a different problem, i.e., a different training set. However, in the text we refer to the $\mathbf{t_{th}}$ iteration of the ant as a separate ant, called the $\mathbf{t_{th}}$ ant ($\mathbf{Ant_t}$), in order to simplify the description of the algorithm.

B. *Rule Pruning*

Rule pruning is a common technique in data mining [9]. As mentioned earlier, the main goal of rule pruning is to remove irrelevant terms that might have been unduly included in the rule. Rule pruning potentially increases the predictive power of the rule; helping to avoid it's over fitting to the training data. Another motivation for rule pruning is that it improves the simplicity of the rule, since a shorter rule is usually easier to be understood by the user than a longer one.

As soon as the current ant completes the construction of its rule, the rule pruning procedure is executed. The strategy for the rule pruning procedure is similar to that suggested by [11], It is possible to re-express complex decision trees as small sets of production rules that outperform the original trees when asked to classify unseen cases. The methods outlined here also provide a way to merge different decision trees for the same task, thereby obtaining another increase in accuracy. But the rule quality criteria used in the two procedures are very different.

The basic idea is to iteratively remove one term at a time from the rule while this process improves the quality of the rule. More precisely, in the first iteration, one starts with the full rule. Then it is tentatively tried to remove each of the terms of the rule—each one in turn—and the quality of the obtained rule is computed using a given rule-quality function [defined by formula (1)]. It should be noted that this step might involve replacing the class in the rule consequent, since the majority class in the cases covered by the pruned rule can be different from the majority class in the cases covered by the original rule. The term whose removing most improves the quality of the rule is effectively removed from it, completing the first iteration. In the next iteration, the term whose removal most improves the quality of the rule is again removed and so on. This process is repeated until the rule has just one term or until there is no term whose removing will improve the quality of the rule.

C. *Pheromone Updating*

Recall that each $\mathbf{term_{ij}}$ corresponds to a segment in some path that an ant can follow. At each iteration of the **WHILE loop** of **Algorithm I** all $\mathbf{term_{ij}}$ are initialized with the same amount of pheromone, so that when the first ant starts its search, all paths have the same amount of pheromone. The initial amount of pheromone deposited at each path position is inversely proportional to the number of values of all attributes and is defined by formula (2) The "**a**" is the total number of attributes and "$\mathbf{b_i}$" is the number of possible values that can be taken on by attribute $\mathbf{A_i}$.

Whenever an ant constructs its rule and it is modified (see **Algorithm I**), the amount of pheromone must be updated in all segments of all paths. The two following basic ideas can support this pheromone updating:

1) The amount of pheromone associated with each $\mathbf{term_{ij}}$ is increased in proportion to the quality of that rule when the ant finds the rule (after pruning);

2) The amount of pheromone associated with each $\mathbf{term_{ij}}$ t is decreased, when it does not occur in simulating pheromone evaporation in real ant colonies.

1) *Increasing the Pheromone of Used Terms:* Increasing the amount of pheromone associated with each $\mathbf{term_{ij}}$ corresponds to increasing the amount of pheromone along the path completed by an ant when the ant finds the rule. In a rule discovery context, this corresponds to increasing the probability of $\mathbf{term_{ij}}$ being chosen by other ants in the future in proportion to the quality of the rule. The quality of a rule, denoted by **Q**, is computed by the formula **Q** = sensitivity × specificity[11], defined as

$$Q = \frac{TP}{TP+FN} \cdot \frac{TN}{FP+TN} \qquad (1)$$

$$\tau_{ij}(t=0) = \frac{1}{\sum_{i=1}^{a} b_i} \qquad (2)$$

where
TP: True **P**ositives, the number of cases covered by the rule that has the class predicted by the rule.
FP: False **P**ositives, the number of cases covered by the rule that has a class different from the class predicted by the rule.
FN: False **N**egatives, the number of cases that are not covered by the rule but that has the class predicted by the rule.
TN: True **N**egatives, the number of cases that are not covered by the rule and that do not have the class predicted by the rule.

Q has a value within the range 0<=Q<=1 and the larger the value of **Q's**, the higher the quality of the rule will be. Pheromone updating for a $\mathbf{term_{ij}}$ is performed according to formula (3), for all terms $\mathbf{term_{ij}}$ that occur in the rule:

$$\tau_{ij}(t+1) = (1+Q) \cdot \tau_{ij}(t), \quad \forall i, j \in R \qquad (3)$$

where **R** is the set of terms occurring in the rule constructed by the ant at iteration **t**.

Therefore, for all **term**$_{\mathbf{ij}}$ occurring in the rule found by the current ant, the amount of pheromone is increased by a fraction of the current amount of pheromone and this fraction is given by **Q**.

2) *Decreasing the Pheromone of Unused Terms:* As mentioned above, the amount of pheromone associated with each **term**$_{\mathbf{ij}}$ that does not occur in the rule found by the current ant has to be decreased in order to simulate pheromone evaporation in real ant colonies. In Ant-Miner, pheromone evaporation is implemented in a somewhat indirect way. More precisely, the effect of pheromone evaporation for unused terms is achieved by normalizing the value of each pheromone τ_{ij}. This normalization is performed by dividing the value of each τ_{ij} by the summation of all τ_{ij}, $\forall i, j$.

Data Set	Cases	Reason attributes			
		Short	Tinder	Poisoning	Other
Electric appliance	350	335	0	0	15
Oil appliance	150	0	145	0	5
Gas appliance	150	0	95	40	15

Table 1. Data sets used in the experiment

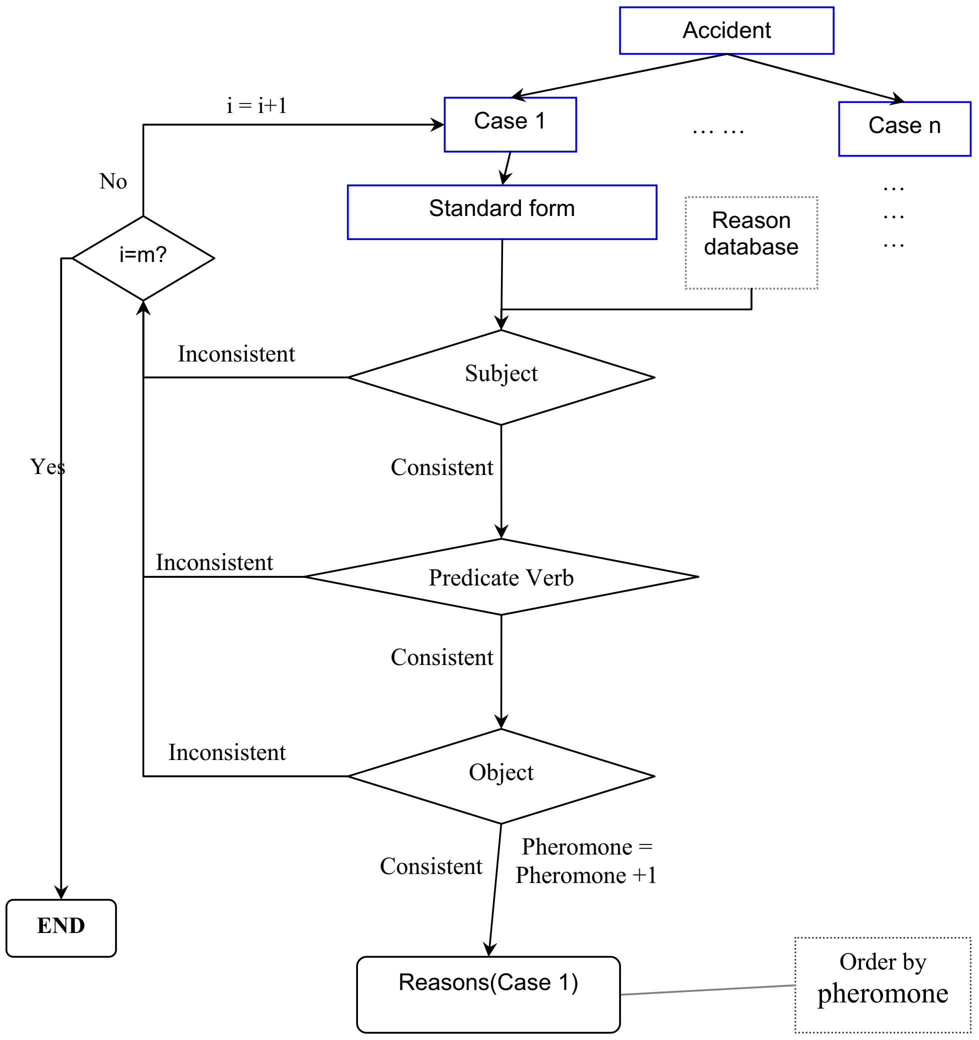

Figure 1. The process of Ant-Miner

Data Set	Cases	Reason attributes			
		Short	Tinder	Poisoning	Other
Electric appliance	350	320(95.52%)	0	0	30
Oil appliance	150	0	135(93.10%)	0	15
Gas appliance	150	0	85(89.47%)	33(82.50%)	32

Table 2. The result of experiment

To see how this implements pheromone evaporation, it should be noted that only the terms used by a rule have their amount of pheromone increased by (3). Therefore, at normalization time, the amount of pheromone of an unused term will be computed by dividing its current value [not modified by (3)] by the total summation of pheromone for all terms [which was increased as a result of applying (3) to all used terms]. The final effect will be the reduction of the normalized amount of pheromone for each unused term. Used terms will, of course, have their normalized amount of pheromone increased due to the application of (3).

V. ACO BASED DATA MINING

The performance of Ant-Miner was evaluated using 3 public-domain data sets from the NIIE(National Institute of Technology Evaluation).

The main characteristics of the data sets used in our experiment are summarized in Table 1. The first column of this table gives the data set name, while the other columns indicate: the number of cases, the number of classes of the data set (Short, Tinder, Poisoning, Other), respectively.

A. Extracting the accident causes

At first, we build an accident cause database. It should be noted that we can extract the accident causes by key words which are about accident products. But we may get a lot of results and so we can not get the exacter causes. Then we must add other conditions to limit the results. The sentences about accident content must be classified into two types: 1. Subject + Be + Adjective/Noun/Verb past tense, *e.g. "...about 10 square meters area was burnt in the second floor..."*; 2. Subject + Verb + Object, *e.g. "...In fire fighting, the cause was investigated that the fire started from the surround of the television in the first floor....".*

And then we delete the adjective, adverb, quantifier and change them to standard form: 1. "area be burn"; 2. "fire start from television". At last, we can search the cause database by standard form and judge whether it is accident cause or not. If its standard form is the same as cause database, then extract the cause. Of course, The "Subject + Verb + Object" series sentences include the "Subject + Be + Adjective/Noun/Verb past tense" series sentences. [12]

B. Embedding Ant-Miner in accident system

Nouns are singular noun and plural noun (2 types). Verbs are present simple, present progressive, present perfect, present perfect progressive, past simple, past progressive, past perfect, past perfect progressive, future simple, future progressive, future perfect, future perfect progressive, passive voice (13 types). So we need change them to standard form to compare. In the Cause database, we save a lot of samples by standard form.

For improving the system accuracy, we embedded the pheromones which like ants leave pheromone to each other for getting the shortest route. Ant-miner leaves the pheromone to each other too. When we check the accident causes again, we can find the more accurate results very easily.

Figure 1 shows the process of this system("i" is index of this process; "m" is the number of sentences which are in case.)

We attempt to optimize the mining method. And we obtained results as shown in Table 2. We obtained more than one cause for every case. But the first cause which has the strongest pheromone is the best one. So we choose it as the accident cause.

C. Mining Effect of Ant-Miner

From Table 2, we can obtain an average predictive accuracy of 88.64%. Therefore, in the three data sets used in the experiments, rule pruning seems to be beneficial. The fact that rule pruning cannot extract any cause by 100% accuracy is not surprising. But, rule pruning is essential to improve accuracy of data mining.

VI. CONCLUSIONS AND FUTURE WORKS

We have presented here a novel application of a new Ant Colony Optimization model, called Pheromone-Miner, to the data mining problem in accident cause system. We have compared its theoretical and actual results. It shows that Pheromone-Miner is able to enhance the data mining in intelligence.

As future work, a better understanding of the theoretical properties of ACO algorithm is certainly another research direction that will be pursued in the future. Seventeen years ago, when the first ACO algorithm was introduced, taking inspiration from ants for designing optimization algorithms seemed a crazy idea. The many successful applications presented in this article have changed our perspective: what seemed a far out idea is now considered one of the most promising approaches to the approximate solution of difficult optimization problems.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, Volume 247, Issue 935, pp. 529-551, April 1955. *(references)*

[2] M. Dorigo, A. Colorni, and V. Maniezzo, “The ant system: Optimization by a colony of cooperating agents,” *IEEE Trans. Syst. Man Cybern. B*, vol. 26, pp. 29–41, Feb. 1996.

[3] M. Dorigo and G. Di Caro, “The ant colony optimization meta-heuristic,” in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. New York: McGraw-Hill, 1999, pp. 11–32.

[4] M. Dorigo, G. Di Caro, and L. M. Gambardella, “Ant algorithms for discrete optimization,” *Artif. Life*, vol. 5, no. 2, pp. 137–172, 1999

[5] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery: An overview,” in *Advances in Knowledge Discovery & Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Cambridge, MA: MIT Press, 1996, pp. 1–34.

[6] A. A. Freitas and S. H. Lavington, *Mining Very Large Databases with Parallel Processing*. Norwell, MA: Kluwer, 1998.

[7] N. Monmarché, “On data clustering with artificial ants,” in *Data Mining with Evolutionary Algorithms, Research Directions – Papers from the AAAI Workshop*, A. Freitas, Ed. Menlo Park, CA: AAAI Press, 1999, pp. 23–26.

[8] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford Univ. Press, 1999.

[9] T. Stützle and M. Dorigo, “ACO algorithms for the traveling salesman problem,” in *Evolutionary Algorithms in Engineering and Computer Science*, K. Miettinen, M. Makela, P. Neittaanmaki, and J. Periaux, Eds. New York: Wiley, 1999, pp. 163–183.

[10] L. A. Brewlow and D. W. Aha, “Simplifying decision trees: A survey,” *Knowl. Eng. Rev.*, vol. 12, no. 1, pp. 1–40, 1997.

[11] J. R. Quinlan, “Generating production rules from decision trees,” in *Proc. Int. Joint Conf. Artificial Intelligence*, San Francisco, CA, 1987, pp. 304–307.

[12] H. S. Lopes, M. S. Coutinho, and W. C. Lima, “An evolutionary approach to simulate cognitive feedback learning in medical domain,” in *Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives*, E. Sanchez, T. Shibata, and L. Zadeh, Eds, Singapore: World Scientific, 1998, pp. 193–207.

[13] Jianxiong Yang, Junzo Watada; " Accident Prevention System based on Semantic Network," International Conference on Machine Learning and Cybernetics 2008，P.3738-3743，2008．7

[14] Junzo Watada, Keisuke Aoki, Masahiro Kawano, Muhammad Suzuri Hitam, Dual Scaling Approach to Data Mining from Text Data base, Journal of Advanced Computational Intelligence Intelligent Informatics (JACIII), Vol. 10, No. 4, pp. 441-447, 2006.12