Internet Electronic Journal of Molecular Design **2004**, *3*, 572–585 ISSN 1538–6414
BioChem Press http://www.biochempress.com

Inter*net* Electronic Journal of Molecular Design

September 2004, Volume 3, Number 9, Pages 572–585

Editor: Ovidiu Ivanciuc

Proceedings of the Internet Electronic Conference of Molecular Design, IECMD 2003
November 23 – December 6, 2003
Part 3

An Ant Colony Optimization–based Classifier System for Bacterial Growth

Prakash S. Shelokar, Valadi K. Jayaraman, and Bhaskar D. Kulkarni

Chemical Engineering & Process Development Division, National Chemical Laboratory, Pune 411008, India

Received: November 20, 2003; Revised: March 5, 2004; Accepted: May 15, 2004; Published: September 30, 2004

Citation of the article:
P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, An Ant Colony Optimization–based Classifier System for Bacterial Growth, *Internet Electron. J. Mol. Des.* **2004**, *3*, 572–585, http://www.biochempress.com.

Inter*net* Electronic Journal of Molecular Design

BioChem Press
http://www.biochempress.com

An Ant Colony Optimization–based Classifier System for Bacterial Growth[#]

Prakash S. Shelokar, Valadi K. Jayaraman, and Bhaskar D. Kulkarni *

Chemical Engineering & Process Development Division, National Chemical Laboratory, Pune 411008, India

Received: November 20, 2003; Revised: March 5, 2004; Accepted: May 15, 2004; Published: September 30, 2004

Internet Electron. J. Mol. Des. **2004**, *3* (9), 572–585

Abstract

Motivation. In predictive microbiology, identification of different combination of environmental factors (such as temperature, water activity, pH), which lead to growth/ no–growth of microorganism, is a problem of potential importance. Ant colony optimization (ACO) is one of the most recently developed nature–inspired metaheuristic techniques, based on the foraging behavior of real life ants and has already exhibited superior performance in solving combinatorial optimization problems. This work explores the search capabilities of this metaheuristic for learning classification rules in bacterial growth/no growth data pertaining to pathogenic *Escherichia coli* R31 as affected by temperature and water activity. The discovered rules thus can be used to verify whether any combination of temperature and water activity belong to either growth or no–growth of the microorganism.

Method. The ant algorithm for classification works iteratively as follows: At any iteration level, software ants construct rules using available heuristic information and dynamically evolved pheromone trails. A rule that has highest prediction quality is said to be a discovered rule, which represents information extracted from the database. Examples correctly covered by the discovered rule are removed from the training set, and another iteration is started. Guided by the modified pheromone matrix, the agents build improved rules and the process is repeated for as many iterations as necessary to find rules covering almost all cases in the training set.

Results. The developed ACO classifier system is utilized on several datasets and its performance is compared with the performance of other well known algorithms in terms of the average accuracy attained in 10–fold cross validation. The results obtained by this algorithm compare very favorably with other classifiers. Additionally, for discovery of classification rules in the dataset pertaining to bacterial growth/no–growth, the performance of the ACO classifier is compared with the C4.5 system with respect to the predictive accuracy and the simplicity of discovered rules. In both these performance indices the ACO classifier compares very well with the C4.5.

Conclusions. The results obtained on several data sets indicate that the algorithm is competitive and can be considered a very useful tool for knowledge discovery in a given database.

Keywords. Ant colony optimization; metaheuristic; classification; *Escherichia coli*; C4.5.

Abbreviations and notations

ACO, Ant colony optimization	NN, Nearest neighbors
FEBANN, Feedforward error back propagation artificial neural network	PNN, Probabilistic neural network
F_{ANNC}, Fast Adaptive Neural Networks for Classification	LLR, Linear logistic regression
BP, Back propagation artificial neural network	NLLR, Nonlinear logistic regression
HDT, Hybrid decision tree	

[#] Presented in part at the Internet Electronic Conference of Molecular Design, IECMD 2003.

* Correspondence author; phone: 91–020–589–3095; fax: 91–020–589–3041; E–mail: bdk@ems.ncl.res.in.

1 INTRODUCTION

Classifying new patterns from a domain of observable attributes by associating their membership to a class belonging to a set of classes is a problem having wide applications in diverse fields including chemistry, microbiology, molecular biology etc [1–20]. Methods applied for this task can be broadly categorized as statistical [1–5], clustering [6–9], rule based methods [10,11], artificial neural networks [12–19], and support vector machines [20].

Compared with the rule–based methods, neural networks have some advantages. Neural networks can handle noisy data. They can induce hypotheses that generalize better than those of other competing algorithms and have relatively high classification accuracy rate. Several empirical studies have pointed out that there is some problem domains in which neural networks provide superior predictive accuracy to commonly used symbolic learning algorithms. However, the trained neural networks are not commonly used for data mining tasks because they are usually not comprehensible, and many neural network learning methods are slow, making them impractical for very large datasets. A hypothesis represented by a trained neural network is a black box, which need to be translated into a more comprehensible language; rule extraction can be one of the ways to solve this problem. Rule based classification, which partitions a given dataset into disjoint groups, is one very important class of data mining problems. In this task, the discovered knowledge is often expressed as a set of rules in the form:

IF <*conditions*> THEN <*class*>

The advantage of this knowledge representation is that it is quite insightful to the user hence we have used this kind of knowledge representation to express the information extracted by the proposed classifier system. Approaches proposed so far for mining classification rules from databases are mainly decision tree based like C4.5, ID3 on symbolic learning methods. Recently, metaheuristic algorithms [10,11,21,22] are also used for knowledge discovery in a database. In this paper, we have explored the idea of nature inspired metaheuristic algorithm, known as ant colony optimization as a rule based machine–learning system. The ant colony optimization (ACO) is a population based co–operative search technique that imitates the way how real–life ants, using the pheromone trail communication, find the shortest path from their nest to food source and back [23]. The ant colony behavior is described using the diagram shown as Figure1.

Initially, ants start their foraging process by moving randomly in different directions. Some of them follow a path that turns out to be shorter and the rest of the ants trace the longer route. During their sojourns ants deposit pheromone on the ground. Ants that follow shorter route need less time to cover the distance (from nest to food source and back) as compared to along the longer path hence they trace this path with strong pheromone deposition and forming a pheromone trail. Ants also possess the ability to sense the pheromone. Thus, ants probabilistically favor moving in the direction in which there is a higher concentration of pheromone. More and more ants follow the

path of stronger pheromone trail and also simultaneously deposit their own pheromone in the trail and the shortest route of the entire colony is quickly established. It is this autocatalytic and pheromone mediated collective intelligent swarm behavior of real life ants that inspired researchers to devise artificial ant algorithms to solve several problems [23–29].

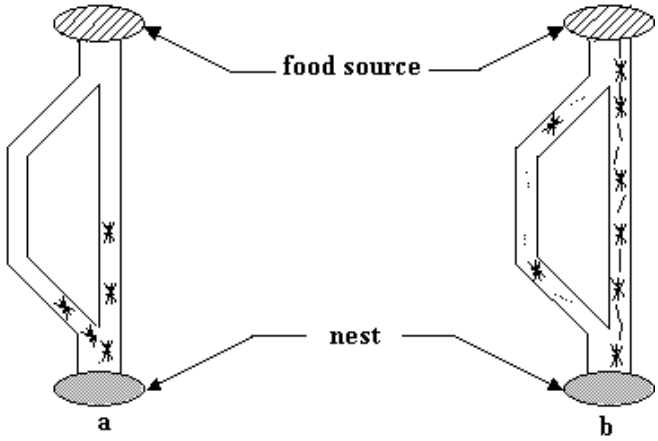

Figure 1. Movement of ants from nest to food source and back. (*a*) Ants randomly start with equal probability in different directions of food source. (*b*) Pheromone is deposited more quickly on the shorter path and eventually most of the ants choose the shortest path.

In predictive microbiology, developing classification models for identification of growth/no–growth of a given microorganism under a set of environmental conditions (e.g. temperature (T), water activity (A_w), pH) is an important class of research activity [1–5,12–19]. Methods based on statistical logistic regression are commonly used for classification and prediction of growth under environmental factors [1–5]. For example, Presser et al [2] using logistic regression derived the equation for estimating growth of *E. coli* strain with respect to T, pH, A_w and lactic acid concentration. Recently, artificial neural networks also been utilized for classification task [12–19]. For example, modeling the time dependent microorganism growth [14,16].

This paper illustrates how *ant colony optimization* can be utilized in predictive microbiology for discovery of classification rules in the data pertaining to bacterial growth as affected by environmental factors. The ant algorithm for classification works as follows. At any iteration level, software ants employ available heuristic information and dynamically evolved pheromone trails to construct rules. A rule that has highest prediction quality is said to be a discovered rule, which represents information extracted from the database. Examples correctly covered by the discovered rule are then removed from the training set, and another iteration is started. This process is repeated for as many iterations as necessary to find rules covering almost all cases in the training set. At this point the procedure has already discovered several rules, which can be readily applied to future problems or can become a part of an expert system. The algorithm is applied on the dataset containing the instances of bacterial growth/no–growth pertaining to pathogenic *Escherichia coli* R31 as affected by T and A_w. The objective is to develop classification rules that are able to identify

any (T, A_w) combination as belonging to either one of the two classes: growth or no–growth. The performance of the ACO classifier is also compared with the performance of tree based C4.5 algorithm with respect to the predictive accuracy and the simplicity of discovered rules.

2 MATERIALS AND METHODS

2.1 Method

The present ACO classifier system handles categorical attributes to learn rules hence continuous attributes need to be discretized. This can be readily done by using any of the several discretization methods available in the literature [30]. The simplest discretization technique is to divide each attribute into domains of equal length. We have used the C4.5–Disc discretization algorithm given in Kohavi and Sahami [31]. The C4.5 software contains this algorithm for discretizing features. To explain the ACO classifier details, a sample training dataset is considered as an illustrative example. It is given both in continuous and discretized form as shown in Table 1.

Table 1. Illustrative training dataset to describe steps in the ACO algorithm for rules discovery

		Dataset (cases, N = 11; attributes, n = 2)										
Case No		1	2	3	4	5	6	7	8	9	10	11
Continuous	$A1$	85	80	70	65	64	70	69	75	75	71	68
data	$A2$	85	90	96	83	65	95	70	79	70	80	79
Discretized	$A1$	D_{12}	D_{12}	D_{11}	D_{11}	D_{11}	D_{11}	D_{11}	D_{12}	D_{12}	D_{12}	D_{11}
data	$A2$	D_{22}	D_{22}	D_{22}	D_{22}	D_{21}	D_{22}	D_{21}	D_{21}	D_{21}	D_{22}	D_{21}
Class		1	1	2	1	2	1	2	2	2	1	2

For discretizing continuous attributes of the training set, the ranges of the domains of each attribute were obtained as: for attribute, A_1: $\{D_{11} = A_1 \leq 70, D_{12} = A_1 > 70\}$ and for attribute A_2: $\{D_{21} = A_2 \leq 79, D_{22} = A_2 > 79\}$.

Table 2. A set of classification rules discovered by the ACO algorithm for illustrative dataset given in Table 1

Rule no.	Rule: *IF <antecedent> THEN <consequent>*	
	Antecedent	*consequent* (*predicted class*)
α_1	$A_1 = D_{12}$ AND $A_2 = D_{22}$	C_1
α_2	$A_2 = D_{21}$ AND $A_1 = D_{11}$	C_2
α_3	$A_1 = D_{11}$ AND $A_2 = D_{22}$	C_1
α_4	Default Rule	C_2

The ACO algorithm for discovery of an optimal set of classification rules in a given dataset is based on a pheromone meditated cooperative search capabilities of ants. Solutions generated by the software ants or agents are in the form of rules. The structure of the rule is: *IF <antecedent> THEN <consequent>*. The <antecedent> part of the rule contains terms T_{ij} using the logical operator, AND. The term T_{ij} is of the form A_i=D_{ij}, where A_i is the i^{th} attribute and D_{ij} is the j^{th} value of the domain of A_i. The <*consequent*> part of the rule is the predicted class that maximizes the quality of rule. For example, consider one of the rules given in Table 2 as: *IF* A_1=D_{12} AND A_2=D_{22} *THEN* C_1. It can be

interpreted as: if given example has attribute $A_1 \leq 70$ and attribute $A_2 \leq 79$ then it can be assigned to class 1.

The algorithm considers R agents to build rules. An agent starts with an empty rule i.e. rule with no term in its antecedent, and generates an antecedent of the rule by adding one term at a time in its current partial rule. For illustration, consider an antecedent part of the first rule α_1 shown in Table 2 as:

$$antecedent : A_1=D_{12} \text{ AND } A_2=D_{22}$$

This current partial rule covers three cases (1,2 and 10) in the dataset shown in Table 1. To construct a rule, the agent uses a problem dependent information and pheromone trail communication matrix. At the start of the algorithm, the pheromone matrix τ is initialized to some small value, τ_0. The trail value, τ_{ij} at location (i, j) represents the pheromone concentration of term, T_{ij}. The pheromone trail matrix evolves as we iterate. At any iteration level, each one of the agents or software ants develops such trial rules and a rule with highest value of quality measure is denoted as a discovered rule, which represents information extracted from the training set. Cases correctly covered by the discovered rule are removed from the training dataset, and another iteration is started. Guided by the modified pheromone matrix, the agents build improved rules and the process is repeated for as many iterations as necessary to find rules covering almost all cases in the training set.

2.1.1 Algorithm details

As explained earlier, ants start with empty rules and in the first iteration, the elements of the pheromone matrix are initialized to the same values. With the progress of iterations, the pheromone matrix is updated depending upon the quality of rules produced. Let us consider for the purpose of illustration, a training dataset containing N = 11 cases defined by n = 2 attributes as shown in Table 1. To generate a set of classification rules, R = 10 agents are deputed. We now proceed to describe the progress of current iteration, t with a view to providing a clear picture of the algorithm details. The agents build their rules by applying the information provided by the pheromone matrix updated at the end of iteration, t–1 and available heuristic information. To construct a current partial rule, the agent selects term T_{ij} with certain probability given as:

$$P_{ij} = \frac{\tau_{ij}(t).\eta_{ij}}{\sum_{k=1}^{n}\sum_{l=1}^{d_k}\tau_{kl}(t).\eta_{kl}}, \quad \forall\, i,k \in I \tag{1}$$

where, P_{ij} is a normalized probability of choosing term T_{ij}. n is the number of attributes defining an example in a dataset. d_k is the number of domains of attribute A_k. I is the list of attributes not yet used by the agent. η_{ij} is the value of problem dependent heuristic function for term T_{ij}. η_{ij} is a measure of the predictive power of term T_{ij}. Higher value of η_{ij} for the term T_{ij} indicates its high

relevance of being part of the classification rule and hence likely to be selected with greater probability. The heuristic function considered in this study can be given as [32]:

$$\text{info}T_{ij} = -\sum_{c=1}^{C}\left(\left(P\left(c \mid A_i = D_{ij}\right)\right)\bullet\left(\log_2 P\left(c \mid A_i = D_{ij}\right)\right)\right) \tag{2}$$

where, infoT_{ij} is a measure of the quality of term T_{ij} with respect to its ability to improve the predictive accuracy of rule. c is the class attribute. C is the number of classes. $P(c \mid A_i = D_{ij})$ is the empirical probability of observing class c conditional on having observed $A_i = D_{ij}$. The amount of information obtained by equation (2) is normalized in the range $0 \le \text{info}T_{ij} \le \log_2(C)$ to facilitate the use of equation (1). The equation used to normalize infoT_{ij} is given as follows:

$$\eta_{ij} = \frac{\log_2(C) - \text{info}T_{ij}}{\sum_{k=1}^{n}\sum_{j=1}^{d_k}\log_2(C) - \text{info}T_{kj}} \tag{3}$$

Consider the calculation of η_{ij} value for term T_{22} i.e. A_2=D_{22}. It covers a total of six cases (five belong to class one and one is from class two) in the dataset shown Table 1. The value of infoT_{ij} measure for term T_{22} using the equation (2) is 0.6500. Similarly, for other values of the domain of attributes, infoT_{ij} can be given as:

	infoT_{ij}	
i/j	1	2
1	0.9183	0.971
2	0.0	0.6500

Using the equation (3) heuristic information η_{ij} for all the terms can be computed as:

	η_{ij}	
i/j	1	2
1	0.0559	0.0199
2	0.6846	0.2396

Let us consider the pheromone trail matrix at the start of current iteration t as:

	τ_{ij}	
i/j	1	2
1	0.0148	0.0153
2	0.0199	0.0120

From the estimates of predictive power η_{ij} and the current values of the pheromone trail τ_{ij} the normalized probability of a term T_{ij} can be computed by using Eq. (1). To illustrate how an agent chooses a term to add in the current partial rule by Eq. (1), consider that an agent has started with an empty rule and is presently developing the antecedent part, α_2, as shown in Table 2. To select a term, first compute normalized probabilities for all the terms T_{ij}. This can be done by using the

above values of η_{ij} and τ_{ij} by Eq. (1) as: 0.0454, 0.0170, 0.7714, and 0.1645. Then draw a random number r in the range (0,1) using uniform distribution. If r is less than 0.0454 then the term T_{11} is chosen. If r lies between 0.0454 and 0.0624 then term T_{12} is preferred. If it is in the range from 0.0624 to 0.7884 then the term T_{21} is selected and if it is greater than 0.7884 then term T_{22} is chosen. Suppose the random number generated is r = 0.4546. It lies in the range 0.0624 to 0.7884 hence term T_{21} *i.e.* $A_2 = D_{21}$ is chosen. The algorithm can add the chosen term (say, T_{21}) in the current partial rule if the following conditions are satisfied: (*i*) it covers the minimum number of cases in the training set defined *a priori*, *min_cover_cases* (for illustrative example, *min_cover_cases* = 3), and (*ii*) the attribute (say, A_2) is not already been used by the agent. Similarly, the agent selects the term T_{11} and checks its feasibility with both the above–mentioned conditions. Both the terms T_{21} and T_{11} individually and in combination (T_{21} AND T_{11}) cover the number of cases in the training dataset greater than or equal to *min_cover_cases*. The developed antecedent part by an agent using the above formalism is depicted in the current partial rule α_2 in Table 2. The algorithm chooses the consequent (i.e. predicted class) for the antecedent of the rule α_2 that maximizes the quality of the rule. This is done by assigning to the rule consequent, the majority class among the cases covered by the antecedent of the rule. Consider the antecedent of the rule α_2, and it is clear from Table 1 that it covers the cases 5,7, and 11. All these covered cases belong to class two hence, the rule consequent of the rule α_2 is class two, C_2. The quality of the constructed rule is calculated as [33]:

$$Q = \frac{TP}{TP + FN} x \frac{TN}{FP + TN} \tag{4}$$

where, *TP* is the number of cases covered by the rule that have the class predicted by the rule. *FP* is the number of cases covered by the rule that have a class different from the class predicted by the rule. *TN* is the number of cases that are not covered by the rule and that do not have the class predicted by the rule. *FN* is the number of cases that are not covered by the rule but that have the class predicted by the rule. As soon as agent develop its solution (rule), pheromone trail is updated locally as [34]:

$$\tau_{ij} = (1 - \rho).\tau_{ij} + \rho.\tau_0 \tag{5}$$

The local trail updating is similar to trail evaporation in real ants. This process intends agents to bias the exploration to those parts of the search space rather than following the strongly concentrated trails. Thus, pheromone trails related to terms T_{21} and T_{11} are updated corresponding to the rule α_2. Similarly, remaining nine agents will develop their rules in current iteration *t*. The process of rule development by the *R* agents can be terminated earlier if a rule constructed by the current agent is same as one of the rules developed by the previous (*no_rules_converged* –1) agents, where *no_rules_converged* is an algorithm parameter used to test the convergence of the software ants. This criterion of terminating the rules construction process in the current iteration in a way reflects the establishment of the shortest path in real–life ant colony.

The rule of highest quality among the rules constructed by all the agents is considered as a (best) discovered rule. It is stored in the special set of discovered rules in the order of its discovery. The pheromone trail matrix is updated globally using the quality measure of the best rule discovered. It is called global pheromone trail updating [34] and is given as:

$$\tau_{ij} = (1-\rho).\tau_{ij} + \rho.Q \tag{6}$$

The global trail updating process is a kind of intensification, which helps agents to favor, terms that are part of higher quality rules. Suppose the best rule among the rules constructed by ten agents at current iteration is rule α_2; the pheromone trails related to terms T_{21} and T_{11} (found in the antecedent part of α_2) would be updated by the global pheromone trail updating process. This completes an iteration of the ACO algorithm. The cases covered by the discovered rule are removed from the training dataset and in the next iteration agents will work on the reduced dataset. The algorithm works for as many iterations as necessary to find rules covering almost all the examples in a dataset or leaving uncovered examples in the training dataset less than a predefined number *max_cases_uncovered*. This terminates the iteration process of the ACO algorithm. At this stage, the ant classifier has developed several rules. A default rule is added at the bottom of the set of discovered rules. The default rule has the empty antecedent and its rule consequent is the majority class that predicts among the cases in the training dataset not covered by any of the discovered rules.

The set of discovered rules thus developed by the ACO system can be applied on the new test cases, unseen during the training process. The rules are tried in the ordered list on a new test case. If the first rule is applied on the new case and the attributes of the test case satisfy antecedent part of this rule, then it assigns its consequent to the test case. The system will apply default rule on the test case if any of the rules from the list of discovered rules is not able to classify the test case.

2.2 Data Set

To exemplify the application of the ACO algorithm as rule based method for classification problems in predictive microbiology, the dataset of bacterial growth/no growth pertaining to pathogenic *Escherichia coli* R31 as affected by a set of operating conditions is utilized. The description of the dataset is given as:

2.2.1 *Escherichia coli* R31 growth/no–growth data

This dataset is given by Salter et al [3]. The dataset pertaining to growth/no–growth of an *E. coli* strain R31 as affected by temperature and water activity is used for learning the classification rules. The data consist of experiments performed with different combinations of temperature in the range 7.7 to 37.0 and water activity in the range 0.943 to 0.987, using NaCl as humectant. All samples of *E. coli* R31 were cultured in plates and L–tube observed daily. If growth in a sample occurred, it

was scored positive. The growth in the sample was related to a visible increase in turbidity or deposit in the base of the tube. If after 50 days there was neither turbidity nor deposit, a loopful of culture was streaked onto plate count agar to determine if any growth is present. A total of 179 samples were observed using different values of temperature and water activity, with 99 as growth cases and 80 no–growth cases.

3 RESULTS AND DISCUSSION

To demonstrate the competitiveness of the proposed classification system, we applied it on several datasets. All the datasets are from the UCI machine learning repository [35] and are designed for classification tasks. Examples that have missing values are removed. Information of the datasets is given in Table 3.

Table 3. Summary of datasets utilized for the performance study of the ACO classifier system

Dataset		size	class	Attributes		
abbreviation	original name			total	continuous	discrete
iris	iris plant	150	3	4	4	0
pima	pima indians diabetes	768	2	8	8	0
win	wine recognition	179	3	13	13	0
wis	wisconsin breast cancer	683	2	9	9	0

The performance of the ACO classifier is compared with the performance of other classifier systems such as C4.5, BP, F_{ANNC}, HDT, and NN. Our experiments were conducted using the 10–fold cross–validation technique, which is a commonly used technique to evaluate the performance of a pattern classification algorithm [36]. We conducted the 10–fold cross–validation in the following way: we randomly partitioned each data set into 10 groups, each group having approximately the same number of examples. Then we ran the ACO system 10 times. Each time, one different group of data was held out for use as the test set; the other nine groups were used as the training set. The results are reported in Table 4 as average values of rate of correct classification for these ten runs.

Table 4. Comparison of ACO classifier versus other classifiers

Dataset	Accuracy rate (%)					
	ACO	C4.5 [37]	HDT [37]	BP [37]	F_{ANNC} [37]	NN [38]
Iris	96.00	94.68	93.98	93.62	93.98	96.00
Pima	74.20	74.26	70.76	71.06	70.76	75.30
win	97.88	92.54	96.56	96.90	96.56	97.70
wis	96.02	93.00	95.60	96.06	95.60	97.30

The results show that the ACO classifier performed as good as NN on two datasets namely, iris and win and only ranked second to NN in terms of accuracy on datasets, pima and wis. The experimental results show the effectiveness of the proposed ACO classifier system.

Now, we applied the ACO classifier to extract classification rules in the dataset of bacterial

growth/no–growth pertaining to pathogenic *E. coli* as affected by T and A_w. The ACO classifier system is executed in C++ compiler on Pentium 533 MHz PC. We evaluated the performance of the proposed classifier system with the decision tree based C4.5 algorithm. The C4.5 is a well–known induction algorithm developed by Quinlan [39]. It converts input data into a decision tree, which can be used to classify a test case by starting from the root of the tree and continuing down the branches until a terminating leaf is encountered. C4.5 applies information theory to the training samples with an information maximization process to generate a decision tree. Detailed description of C4.5 algorithm can be found elsewhere [39]. The comparison is based on the predictive power of discovered rules on testing examples and the simplicity of rules discovered. A 10–fold cross validation procedure is used to measure the predictive accuracy of the algorithm. The results comparing the average predictive accuracy of rule set discovered by the ant classifier and the C4.5 are given in Table 4 while the results comparing the simplicity of discovered rule set are given in Table 5.

Table 4. Average predictive accuracy rate of rules discovered by the algorithms on *E. coli* dataset

Average predictive accuracy on test set (%)	
ACO	C4.5
98.89	93.33

Table 5. Measure of simplicity of rule sets discovered by the algorithms on *E. coli* dataset

Average number of rules		Average number of terms	
ACO	C4.5	ACO	C4.5
12.1	10.4	22.1	47.3

The average predictive accuracy of discovered rules obtained by the ACO classifier in the classification of the bacterial growth/no growth data pertaining to pathogenic *Escherichia coli* R31 is higher than that obtained by the C4.5 algorithm as can be seen from Table 4. The accuracy rate of the learned rules by the ACO classifier is 98.89 while that of C4.5 is 93.33. The average number of rules obtained by the ACO system on the ten training sets is 12.1 as compared to 10.4 obtained by C4.5 (Table 5). The average number of rules discovered by the ACO system is slightly on higher side as compared to that obtained by the C4.5, but the average number of terms considered in the rule set discovered by the ACO classifier is quite less than the number of terms used by the rule set obtained with the C4.5. As can be seen in Table 5, the average number of terms considered by the rule set learned by the ACO classifier is 22.1 while that of 47.3 in the rule set obtained by the C4.5. Hajmeer and Basheer [19] also used logistic regression and artificial neural networks as classifiers for this bacterial growth data and compared them using analysis of the receiver operating characteristic curves as well as a number of scalar performance measures pertaining to the classification contingency matrices. The scalar performance measures for binary classification considered by Hajmeer and Basheer [19] are given as:

$$\text{FC} = \frac{TP+TN}{N},\ \text{FAR} = \frac{FP}{TN+FP},\ \text{POD} = \frac{TP}{FN+TP} \tag{7}$$

To check the performance of the rules learned by the ACO classifier with other classifiers studied by the Hameer and Basheer [19], we have used the measure fraction correct, FC. FC is nothing but a fraction of correct classification. Considering all the cases (combined training and testing set) in the *E. coli* growth dataset, a set of rules discovered by the ACO algorithm was applied to calculate the performance measure, FC. The value of FC obtained by the discovered rules is 0.949. The best value of the performance measure FC reported for various classifiers in [19] is given in the following Table 6.

Table 6. Comparison of the ACO classifier versus other classifiers using training and validation combined (179 cases) *E .coli* growth dataset

classifier type	performance measure (FC)
ACO	0.949
LLR	0.782
NLLR	0.905
FEBANN	0.939
PNN	1.0

As can be seen from Table 6, the developed ACO classifier for bacteria growth has higher FC value than other three algorithms namely, LLR, NLLR and FEBANN. The discovered rule set corresponding to FC = 0.949 misclassified 9 cases out of 179. The number cases misclassified in the *E. coli* growth dataset by LLR, NLLR, FEBANN are 39, 17, and 11 respectively [19]. The discovered rule set for the bacteria growth dataset obtained by the ACO system is given in Table 7.

Table 7. A set of classification rules discovered by the ACO classifier for *E. coli* growth dataset

Sr. No.	Rules
1	IF TEMP ≤ 10.6 AND WAT > 0.977 THEN GROW
2	IF TEMP > 22.2 AND 0.961< WAT ≤ 0.977 THEN GROW
3	IF 15.0 < TEMP ≤ 22.2 AND 0.961< WAT ≤ 0.977 THEN GROW
4	IF 10.6 < TEMP ≤ 15.0 AND 0.961< WAT ≤ 0.977 THEN GROW
5	IF TEMP ≤ 10.6 AND 0.961< WAT ≤ 0.977 THEN NOGROW
6	IF TEMP > 10.6 AND 0.951< WAT ≤ 0.961 THEN GROW
7	IF 15.0 < TEMP ≤ 22.2 AND 0.951< WAT ≤ 0.961 THEN GROW
8	IF 10.6 < TEMP ≤ 15.0 AND 0.951< WAT ≤ 0.961 THEN NOGROW
9	IF TEMP ≤ 10.6 AND 0.951< WAT ≤ 0.961 THEN NOGROW
10	IF TEMP > 22.2 AND 0.949< WAT ≤ 0.951 THEN GROW
11	IF 0.949< WAT ≤ 0.951 THEN NOGROW
12	IF TEMP > 22.2 AND 0.948< WAT ≤ 0.949 THEN GROW
13	IF WAT ≤ 0.948 THEN NOGROW
14	DEFAULT RULE IS GROW

TEMP: temperature, WAT: water activity, GROW: *E. coli* bacteria growth, NOGROW: *E. coli* bacteria no–growth.

We can summarize the results of our experiments taking into account both the accuracy rate and the rule set simplicity criteria. The discovered rule set by the ACO classifier is simpler and more accurate than the rule set discovered by the C4.5 system. Also, a total number of terms in the rules

generated by the ACO system were smaller than that of the rules obtained by the C4.5 method. The advantages of metaheuristic (evolutionary) algorithms as rule based techniques over decision tree based approaches are: first they work with a population of candidate solutions. Second, they evaluate a candidate solution as a whole by the fitness function. These characteristics are in contrast with most greedy rule induction algorithms, which work with a single candidate solution at a time and typically evaluate a partial candidate solution based on local information only. Third, they use probabilistic procedures that make them less prone to get trapped into local minima in the search space. Empirical evidence that in general evolutionary algorithms cope with attribute interaction better than greedy rule induction algorithms, can be found in literature on data mining [40,41]. However, they also have some disadvantages for rule discovery. First, in general, they are considerably slower than greedy rule induction algorithms. Second, the rules developed by the algorithm are stored in the order of their discovery, in order to be applied to a test case; the previous rules in the list must not cover that case. Hence, the rules discovered by the algorithm are not as modular and independent as the rules discovered by the C4.5 system. This has the effect of reducing a little simplicity of the rules discovered by the algorithm in comparison with the rules discovered by the C4.5 system. This effect seems to be compensated by, overall, the size of the rule list discovered by the ant classifier, which is much smaller than the size of the rule set, discovered by the C4.5. So, it can be said that, the rules discovered by the ant classifier are simpler than the rules discovered by C4.5.

4 CONCLUSIONS

In this work, ant colony metaheuristic originally developed for solving combinatorial optimization problems is employed as a rule based classifier. To evaluate the performance of this classifier system is tested on several datasets commonly used for comparison study. The results show that the classifier performs very well on the tested data sets and this system can be considered as an alternative to symbolic techniques for discovery of rules in data mining tasks. In predictive microbiology, the application of the ant colony based classifier system is illustrated by generating classification rules in the dataset pertaining to pathogenic *E. coli* R31 as affected by temperature and water activity. The discovered rules thus can be used to identify any combination of temperature and water activity belonging to either one of the two classes: growth or no–growth, or can become a part of expert systems.

Acknowledgment

The financial assistance received from the Department of Science and Technology, the Government of India, New Delhi is gratefully acknowledged. The author P. S. thanks the Council of Scientific and Industrial Research (CSIR), the Government of India, New Delhi, for a Senior Research Fellowship.

5 REFERENCES

[1] D. A. Ratkowsky and T. Ross, Modeling the bacterial growth/no growth interface, *Lett. Appl. Microbiol.* **1995**, *20*, 29–33.
[2] K. A. Presser, T. Ross, and D. A. Ratkowsky, Modeling the growth limits growth/no growth interface of *E. coli* as a function of temperature, pH, lactic acid concentration, and water activity, *Appl. Environ. Microbiol.* **1998**, *64*, 1773–1779.
[3] M. A. Salter, D. A. Ratkowsky, T. Ross, and T. A. McMeekin, Modeling the combined temperature and salt (NaCl) limits for growth of a pathogenic *E. coli* strain using nonlinear logistic regression, *Int. J. Food Microbiol.* **2000**, *61*, 159–167.
[4] A. Lopez–Malo, S. Guerrero, and S. M. Alzamora, Probabilistic modeling of saccharomyces cerevisiae inhibition under the effects of water activity, pH, and potassium sorbate concentration, *J. Food Prot.* **2000**, *63*, 91–95.
[5] S. Tienungoon, D. A. Ratkowsky, T. A. McMeekin, and T. Ross, Growth limits of Listeria monocytogenes as a function of temperature, pH, NaCl, and lactic acid, *Appl. Environ. Microbiol.* **2000**, *66*, 4979–4987.
[6] O. Beckonert, M. E. Bollard, T. M. D. Ebbels, H. C. Keun, H. Antti, E. Holmes, J. C. Lindon, and J. K. Nicholson, NMR–based metabonomic toxicity classification: hierarchical cluster analysis and *k*–nearest–neighbour approaches, *Analytica Chimica Acta* **2003**, *490*, 3–15.
[7] A. V. Tendulkar, P. P. Wangikar, M. A. Sohoni, V. V. Samant, and C. Y. Mone, Parameterization and classification of the protein universe via geometric techniques, *J. Molecular Biology* **2003**, *334*, 157–172.
[8] F. Chibon, O. Mariani, A. Mairal, J. Derré, J.–M. Coindre, P. T. Réal Lagacé, X. Sastre, and A. Aurias, The use of clustering software for the classification of comparative genomic hybridization data: an analysis of 109 malignant fibrous histiocytomas, *Cancer Genetics Cytogenetics* **2003**, *141*, 75–78.
[9] M. Ankerst, G. Kastenmüller, H.P. Kriegel, and T. Seidl, Nearest neighbour classification in 3D protein databases; in: *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology* (*ISMB'99*), AAAI Press, 1999.
[10] A. H. C. van Kampen, Z. Ramadan, M. Mulholland, D. B. Hibbert, and L. M. C. Buydens, Learning classification rules from an ion chromatography database using a genetic based classifier system, *Analytica Chimica Acta* **1997**, *344*, 1–15.
[11] K. Deb and A. R. Reddy, Reliable classification of two–class cancer data using evolutionary algorithms, *Biosystems* **2003**, *72*, 111–129.
[12] I. A. Basheer, M. N. Hajmeer, and Y. M. Najjar, Computational neural networks for predictive microbiology: II. Application, *Int. J. Food Microbiol.* **1997**, *34*, 51–66.
[13] A. H. Geeraerd, C. H. Herremans, C. Cenens, and J. F. Van Impe, Application of artificial neural networks as a non–linear modular modeling technique to describe bacterial growth in chilled food products, *Int. J. Food Microbiol.* **1998**, *44*, 49–68.
[14] I. A. Basheer and M. N. Hajmeer, Artificial neural networks: fundamentals, computation, design and application. *J. Microbiol. Methods* **2000**, *43*, 3–31.
[15] I. A. Basheer, M. N. Hajmeer, J. L. Marsden, and D. Y. C. Fung, New approach for modeling generalized microbial growth curves using artificial neural networks, *J. Rapid Methods Automat. Microbiol.* **2000**, *8*, 265–284.
[16] A. W. Schepers, J. Thibault, and C. Lacroix, Comparison of simple neural network and nonlinear regression models for descriptive modeling of lactobacillus helveticus growth in pH–controlled batch cultures, *Enzyme Microbiol Technol.* **2000**, *26*, 431–445.
[17] W. Lou and S. Nakai, Application of artificial neural networks for predicting the thermal inactivation of bacteria: a combined effect of temperature, pH, and water activity, *Food Res. Int.* **2001**, *34*, 573–579.
[18] S. Jeyamkondan, D. S. Jayas, and R. A. Holley, Microbial growth modeling with artificial neural networks, *Int. J. Food Microbiol.* **2001**, *64*, 343–354.
[19] M. Hajmeer and I. Basheer, Comparison of logistic regression and neural network classifiers for bacterial growth, *Food Microbiol.* **2003**, *20*, 43–55.
[20] R. Burbidge, M. Trotter, B. Buxton, and S. Holden, Drug design by machine learning: support vector machines for pharmaceutical data analysis, *Comput. Chem.* **2001**, *26*, 5–14.
[21] A. A. Freitas, *Data mining and knowledge discovery with evolutionary algorithms*, Springer–Verlag, Berlin, 2002.
[22] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, An ant colony algorithm for classification rule discovery. in: *Datamining: a heuristic approach*, Eds. H. Abbas, R. Sarker, C. Newton, Idea Group Publishing, London, 2002, pp. 191–208.
[23] M. Dorigo, G. Di Caro, and L. M. Gambardella, Ant algorithms for discrete optimization, *Artificial Life*, **1999**, *5*, 137–172.
[24] M. Mathur, S. Karale, S. Priye, V. K. Jayaraman, and B. D. Kulkarni, Ant colony approach for continuous function optimization, *Ind. Eng. Chem.Res.* **2000**, *39*, 3814–3822.

[25] V. K. Jayaraman, B. D. Kulkarni, K. Gupta, J. Rajesh, and H. S. Kusumaker, Dynamic optimization of fed–batch bioreactors using the ant algorithm, *Biotech*. *Prog*. **2001**, *17*, 81–88.
[26] S. Izrailev and D. K. Agrafiotis, A new method for building regression tree models for QSAR based on artificial ant colony systems, *J*. *Chem*. *Info*. *Comput*. *Sci*. **2001**, *41*, 176–180.
[27] S. Izrailev and D. K. Agrafiotis, Variable selection for QSAR by artificial ant colony systems, *SAR & QSAR in Environ*. *Res*. **2002**, *13*, 417–423.
[28] W. Cedeño and D. K. Agrafiotis, Combining particle swarms and k–nearest neighbors for the development of quantitative structure–activity relationships, *Int*. *J*. *Comput*. *Res*. **2002**, *11*, 443–452.
[29] W. Cedeño and D. K. Agrafiotis, Using particle swarms for the development of QSAR models based on *k*–nearest neighbor and kernel regression, *J*. *Comput*. *Aid*. *Mol*. *Des*. **2003**, *17*, 255–263.
[30] J. Dougherty, R. Kohavi, and M. Sahami, Supervised and unsupervised discretization of continuous features; in: *Proceedings of the 12th international conference Machine Learning*, Eds. A. Prieditis and S. Russell, Morgan Kaufmann, San Francisco, 1995, pp.194–202.
[31] R. Kohavi and M. Sahami, Error–based and entropy–based discretization of continuous features; in: *Proceedings of 2nd international conference on knowledge discovery in databases*, Eds. E. Simondis, J. Han, and U. Fayyad, AAAI Press, 1996, vol. 36, pp. 114–119.
[32] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, 1991.
[33] 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*, Word Scientific, Singapore, 1998, pp. 193–207.
[34] M. Dorigo and L. M. Gambardella, Ant colonies for the traveling salesman problem, *BioSystems* **1997**, *43*, 73–81.
[35] UCI Machine Learning Data Repository: *http://www.ics.uci.edu/~mlearn/MLrepository.html*
[36] R. Duda and P. Hart, D. Stork, *Pattern Classification*, 2nd ed. John Wiley & Sons, New York, 2000.
[37] Z.–H. Zhou and Z.–Q. Chen, Hybrid decision trees, *Knowledge based systems* **2002**, *15*, 515–528.
[38] H. Inoue and H. Narhisa, Optimizing a multiple classifier system, in: *Lecture Notes in Computer Science*, Eds. M. Ishizuka and A. Sattar, Springer–Verlag, Berlin, 2002, vol. 2417, pp. 285–294.
[39] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, CA, 1993.
[40] D. P. Greene and S. F. Smith, Competition–based induction of decision models from examples, *Machine Learning* **1993**, *13*, 229–257.
[41] V. Dhar, D. Chou, and F. Provost, Discovering interesting patterns for investment decision making with GLOWER–A genetic learner overlaid with entropy reduction, *Data Mining and Knowledge Discovery* **2000**, *4*, 251–280.

Biographies

Prakash S. Shelokar is a petro–chemical engineer from University of Pune, India and working as a senior research fellow in Chemical Engineering and Process Development Division of the National Chemical Laboratory, Pune, India. His interests are in conventional and non–traditional optimization algorithms and their applications in the field of Chemical Engineering.

Valadi K. Jayaraman is a senior scientist in the Chemical Engineering and Process Development Division of the National Chemical Laboratory, Pune, India. He obtained his Bachelor's and Master's degrees in Chemical Engineering from the University of Madras and his Ph.D. while working at NCL. His interests include chemical and bio–reaction engineering, non–linear dynamics, control and process optimization. Dr. Jayaraman has been a visiting faculty to many Indian Universities and has given many core chemical engineering courses to graduate students. He has more than 80 technical papers in peer reviewed international journals.

Bhaskar D. Kulkarni is a senior scientist and leads the Chemical Engineering and Process Development Division at the National Chemical Laboratory, Pune, India. He obtained his Bachelor's, Master's and Ph.D. degrees from Nagpur University, India, and has been with NCL for over 30 years. His interests are mathematical modeling of chemical reactions and reactors. A Fellow of the Indian National Science Academy, Dr. Kulkarni has received numerous awards for his work, including the prestigious Bhatnagar Prize for Science and Technology. He has published 4 books and over 200 technical papers in peer reviewed international journals.