Internet Electronic Journal of Molecular Design **2003**, *2*, 000–000
BioChem Press http://www.biochempress.com

An Ant Colony Optimization Classifier System for Bacterial Growth

P.S. Shelokar, V. K. Jayaraman, B.D. Kulkarni*

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

Received xxx; Preprint published xxx; Accepted xxx ; Published xxx

Internet Electron. J. Mol. Des. **2003**, *1*, 000–000

Abstract

Motivation. Ant colony optimization is one of the most recent nature-inspired metaheuristics. The algorithm mimics cooperative foraging behavior of real life ants, has already exhibited superior performance in solving combinatorial optimization problems. In this work, we have explored the searching capabilities of this metaheuristic for learning classification rules in bacterial growth/no growth data pertaining to pathogenic *Escherichia coli* R31.

Method. The algorithm iteratively discovers a set of classification rules for a given dataset. At any iteration level, each one of the software ants develops 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. The 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.

Results. The capability of the ACO algorithm is gauged by considering two real world datasets. The performance of ACO algorithm is compared with the performance of tree based C4.5 algorithm with respect to the predictive accuracy and the simplicity of discovered rules. In both these performance indices ACO algorithm compares very well with C 4.5.

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

Keywords. ACO, Metaheuristic, Optimization, Classification, *Escherichia coli,* C4.5.

Abbreviations and notations

ACO, Ant colony optimization
FEBANN, Feedforward error backpropagation artificial neural network
LLR, Linear logistic regression
NLLR, Nonlinear logistic regression
PNN, Probabilistic neural network

1 INTRODUCTION

Classification tasks form an important set of problems in many fields including chemistry [1, 2], microbiology [3,4], molecular biology [5], process engineering [6] etc. and a number of methods such as decision tree based [7], neural networks [8] and rule based [1, 6, 9] techniques have been applied to tackle these types of problems.

The present paper describes the application of an ant colony optimization metaheuristic as a rule based machine learning technique to discover classification rules in bacterial growth/no growth data pertaining to pathogenic *Escherichia coli* R31. Recently Dorigo and coworkers developed a

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

metaheuristic known as ant colony optimization to solve combinatorial optimization problems [10]. The ant algorithms mimic the techniques employed by real ants to rapidly establish the shortest route from food source to their nest and vice-versa. Ants start searching the area surrounding their nest in a random manner. When an isolated ant comes across some food source in its random sojourn, it deposits certain quantity of pheromone on that location. Other randomly moving ants in the neighborhood can detect this marked pheromone trail. Further, they follow this trail with a very high degree of probability and simultaneously enhance the trail by depositing their own pheromone. It is this auto catalytic process characterized by positive feedback mechanism that helps the ants to establish the shortest route.

The proposed ACO algorithm for generation of classification rules is also based on the indirect communication capabilities of the ants. Software ants are deputed to generate rules by using heuristic information and by using the principle of indirect pheromone communication capabilities for iterative improvement of rules. At every iteration, the algorithm obtains rules constructed by all ants and a rule with highest prediction quality is denoted as a discovered rule, which represents information extracted from the database. The cases (objects) correctly covered by the discovered rule are 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 discovers several rules that can be applied to future problems or can become a part of an expert system.

To evaluate the performance of the ACO algorithm, it is implemented on the two real world datasets, viz, bacterial growth/no growth data pertaining to pathogenic *Escherichia coli* R31, and promoter gene sequences (DNA) of a bacteria with associated imperfect domain theory pertaining to the *Escherichia coli*. The performance of the ACO algorithm is 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 algorithm 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 [11]. The simplest discretization technique is to divide each attribute into equal length. We have used the C4.5-Disc discretization algorithm given in Kohavi and Sahami [12]. This algorithm uses the C4.5 algorithm for discretizing continuous attributes. To explain the algorithm 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 data	$A1$	85	80	70	65	64	70	69	75	75	71	68
	$A2$	85	90	96	83	65	95	70	79	70	80	79
Discretized data	$A1$	D_{12}	D_{12}	D_{11}	D_{11}	D_{11}	D_{11}	D_{11}	D_{12}	D_{12}	D_{12}	D_{11}
	$A2$	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 the 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: $\{D_{21} = A_2 \leq 79, D_{22} = A_2 > 79\}$.

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 the 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.

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 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. The 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:

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

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_i is the number of values on the domain of attribute A_i. *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}. The 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 [13]:

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

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:

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

Consider the calculation of η_{ij} value for term T_{22} i.e. A_2=D_{22}. It covers total 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 equation (1). To illustrate how an agent chooses a term to add in the current partial rule by equation (1), consider the agent has started with empty rule and is developing the antecedent of rule same as the rule α_2 shown in Table 2. To

select a term, normalized probabilities for all the terms T_{ij} can be computed using the above values of η_{ij} and τ_{ij} by equation (1) as: 0.0454, 0.0170, 0.7714, and 0.1645. Draw a random number r in the range (0,1) using uniform distribution. Thus, if r is less than 0.0454 then the term T_{11} is chosen. If r lies between 0.0454-0.0624 then term T_{12} is preferred. If it is in the range 0.0624-0.7884 then 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 0.0624-0.7884 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 the two above-mentioned conditions. Both the terms T_{21} and T_{11} as individual and when combined (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 ACO 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 [14]:

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

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 the agent develop its rule, pheromone trail is updated locally as [15]:

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

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 the rule 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 [15] and is given as:

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

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 one 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 algorithm 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 algorithm 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 Sets

To analyze the performance of the ACO machine learning algorithm to discover classification rules, it is implemented on two data sets namely: bacterial growth/no growth data pertaining to pathogenic *Escherichia coli* R31, and promoter gene sequences (DNA) of a bacteria with associated imperfect domain theory pertaining to the *Escherichia coli*. The description of these datasets is given as:

2.2.1 *E. coli* 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 experimental testing of a large number of combinations of temperature in the range 7.7-37.0 and water activity in the range 0.943-0.987. All samples were observed daily, and a sample was scored positive (i.e. growth occurred) if it showed an 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. For any temperature and water activity combination, growth was recorded as one if it occurred and zero if it did not. The total 179 samples were observed with 99 as growth cases and 80 no-growth cases.

2.2.2. E. coli promoter gene sequences (DNA)

This dataset is obtained from the UCI data repository [16]. It involves the task of recognizing biological concepts in deoxyribonucleic acid (DNA) sequences of the *E. coli* bacteria. The dataset consists of total 106 patterns of 57 bases of DNA out of which 53 as promoters while remaining as non-promoters [17].

3 RESULTS AND DISCUSSION

The proposed ACO machine learning algorithm for classification rule discovery is implemented on two data sets namely: bacterial growth/no growth data pertaining to pathogenic *Escherichia coli* R31, and promoter gene sequences (DNA) of a bacteria with associated imperfect domain theory pertaining to the *Escherichia coli*. The ACO classifier system is executed in C++ compiler on Pentium 533MHz 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 [18]. 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 [18]. The comparison is based on the predictive power of discovered rules on testing examples and the simplicity of rules discovered. A ten-fold cross validation procedure is used to measure the predictive accuracy of the algorithm. The data set is divided arbitrarily into ten partitions. During each run, a different partition is kept aside as test set while the remaining nine partitions are used as training set. The average of rate of correct classification of the ten runs is reported as the predictive accuracy rate of the discovered rule set. The results comparing the average predictive accuracy of rule set discovered by ant classifier and C4.5 are given in Table 3 while the results comparing the simplicity of discovered rule set are given in Table 4.

Table 3. Average predictive accuracy rate of discovered rules learned by the algorithms.

Sr. No.	Data set	Average predictive accuracy on test set (%)	
		ACO	C4.5
1	Ecoli_growth	98.89	93.33
2	Ecoli_promoter	94.00	75.00

Table 4. Measure of simplicity of rule sets discovered by the algorithms

Sr. No.	Data set	Average number of rules		Average number of terms	
		ACO	C4.5	ACO	C4.5
1	Ecoli_growth	12.1	10.4	22.1	47.3
2	Ecoli_promoter	8.0	16.9	11.8	39.2

The average predictive accuracy of discovered rules obtained by the ACO algorithm 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 3. The accuracy rate of the learned rules by the ACO algorithm is 98.89 while that of C4.5 is 93.33. The average number of rules obtained by the ACO algorithm on the ten training sets is 12.1 as compared to 10.4 obtained by C4.5 (Table 4). The average number of rules discovered by the ACO algorithm 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 is quite less than the number of terms used by the rule set obtained by the C4.5. As can be seen in Table 4, the average number of terms considered by the rule set learned by the ACO is 22.1 while that of 47.3 in the rule set obtained by the C4.5. Hajmeer and Basheer [4] 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 [4] are given as:

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

To check the performance of the rules learned by the ACO algorithm with other classifiers studied by the Hameer and Basheer [4], 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 Ecoli_growth dataset, a set of rules discovered by the ACO algorithm is 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 [4] is given in the following Table 5.

Table 5. Performance measure using complete (179 cases) Ecoli_growth dataset for the developed

classifiers.

Classifier type	Performance measure (FC)
ACO (present study)	0.949
LLR	0.782
NLLR	0.905
FEBANN	0.939
PNN	1.0

As can be seen from Table 5, 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 Ecoli_growth dataset by LLR, NLLR, FEBANN are 39, 17, and 11 respectively [4]. The discovered rule set for the bacteria growth dataset obtained by the ACO algorithm is given in Table 6.

Table 6. A set of classification rules discovered by the ACO algorithm for Ecoli_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.

The second dataset is the promoter gene sequences (DNA) of bacteria with associated imperfect

domain theory pertaining to the *Escherichia coli*. From Table 3, the predictive accuracy rate obtained by the ACO algorithm is 94.0, which is higher than that of 75.0 obtained by the C4.5. Similarly, the number of rules discovered by the ACO algorithm is 8.0 and that discovered by the C4.5 is 16.9 (Table 4). In this case, the greater simplicity of the rule set discovered by the ACO algorithm is achieved without unduly sacrificing accuracy rate.

We can summarize the results of our experiments taking into account both the accuracy rate and the rule set simplicity criteria. For both datasets, namely Ecoli_growth and Ecoli_promoter, ACO discovered a rule set that is simpler and more accurate than the rule set discovered by C4.5. Also, the total number of terms of the rules discovered by ACO algorithm was smaller than that of the rules discovered by C4.5.

4 CONCLUSIONS

Ant colony metaheuristic originally developed for solving combinatorial optimization problems is formulated as a rule based machine learning method. The results with two real life data sets indicate that the algorithm is competitive and can be a very useful tool for knowledge discovery in a database.

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] 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.

[2] R. Burbidge, M. Trotter, B. Buxton, S. Holden, Drug design by machine learning: support vector machines for pharmaceutical data analysis, *Computers & Chemistry,* **2001**, *26* 5–14.

[3] M.A. Salter, D.A. Ratkowsky, T. Ross, T.A. McMeekin, Modelling 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] M. Hajmeer and I. Basheer, Comparison of logistic regression and neural network classifiers for bacterial growth, *Food Microbiol.* **2003**, *20*, 43-55.

[5] 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.

[6] B. Özyurt, A.K. Sunol, M.C. Çamrudan, P. Mogili, and L.O. Hall, Chemical plant fault diagnosis through a hybrid symbolic-connectionist machine learning approach. *Computers & Chemical Engineering*, **1998**, *22*, 299-321.

[7] S.-Y. Chang, C.-R. Lin, and C.T. Chang, A fuzzy diagnosis approach using dynamic fault trees, *Chemical Engineering Science*, **2002**, *57*, 2971-2985.

[8] V. Venkatasubramanian, R. Vaidyanathan, and Y. Yamamoto, Process fault detection and diagnosis using neural networks-I. Steady-state processes, *Computers & Chemical Engineering,* **1990,** *14*, 699-712.

[9] 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.
[10] M. Dorigo, G. Di Caro, and L.M. Gambardella, Ant algorithms for discrete optimization, *Artificial Life*, **1999**, *5*, 137-172.
[11] J. Dougherty, R. Kohavi, 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.
[12] 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.
[13] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, 1991.
[14] 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.
[15] M. Dorigo and L.M. Gambardella, Ant colonies for the traveling salesman problem, *BioSystems*, **1997**, *43*, 73-81.
[16] UCI Machine Learning Data Repository: *http://www.ics.uci.edu/~mlearn/MLrepository.html*
[17] C. Harley and R. Reynolds, Analysis of E. Coli promoter sequences, *Nucleic Acids Research*, **1987**, *15*, 2343-2361.
[18] J.R. Quinlan, *C4.5: Programs forMachine Learning,* Morgan Kaufmann, San Francisco, CA, 1993.

Biographies

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

V. K. Jayaraman is a senior Scientist in the Chemical Engineering Division of 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, Pune. His research 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 international publications.

B. D. Kulkarni is a senior Scientist and leads the Chemical Engineering Division at the National Chemical Laboratory, Pune, India. He obtained his Bachelor's, Master's and Ph.D. degrees from Nagpur University, Nagpur, India. His research interests are mathematical modeling chemical reactions and reactors. Dr. Kulkarni has received numerous awards for his work, including the Bhatnagar Prize for Science and Technology. He has published 4 books and over 200 technical papers in prestigious international journals.