176 *J. Chem. Inf. Comput. Sci.* **2001,** *41,* 176−180

A Novel Method for Building Regression Tree Models for QSAR Based on Artificial Ant Colony Systems

Sergei Izrailev* and Dimitris Agrafiotis

3-Dimensional Pharmaceuticals, Inc., 665 Stockton Drive, Exton, Pennsylvania 19341

Received July 25, 2000

Among the multitude of learning algorithms that can be employed for deriving quantitative structure−activity relationships, regression trees have the advantage of being able to handle large data sets, dynamically perform the key feature selection, and yield readily interpretable models. A conventional method of building a regression tree model is recursive partitioning, a fast greedy algorithm that works well in many, but not all, cases. This work introduces a novel method of data partitioning based on artificial ants. This method is shown to perform better than recursive partitioning on three well-studied data sets.

I. INTRODUCTION

The use of artificial intelligence algorithms, such as *k* nearest neighbors, classification and regression trees, and neural networks, for structure−activity correlation has vastly increased over the past few years, due to the growing availability of biological data and the rising demand for more accurate and interpretable models for pharmaceutical development. Regression trees[1] offer several advantages over alternative quantitative structure−activity relationship (QSAR) methodologies, including simplicity, interpretability, and the ability to handle large data sets.[2] A regression tree model can be viewed as a special type of decision tree that relates a continuous target variable *y* (activity) with a set of *M* predictor variables x_k, $k = 1, ..., M$ (molecular property descriptors). Creating a regression tree model usually involves two stages: growing and pruning. The growing stage attempts to partition the training data in a way that minimizes the mean squared error $(1/N)\sum_{i=1}^{N}(y_i - \tilde{y}_i)^2$, where N is the number of training cases and y_i and $\tilde{y}_i$ are the measured and predicted activities of the *i*th case, respectively. The resulting tree is then pruned; i.e., some of the lower branches are eliminated to avoid poor partitioning decisions based on small data samples in the lower levels of the tree. A prediction of the target variable from a given set of values for the predictor variables is made by traversing the tree until a leaf is reached. The predicted value is calculated from a model, e.g., the average or some other linear model, that is derived from the training cases associated with this leaf (see Figure 1).

While there are a multitude of methods for pruning a regression tree and a variety of different models that can be used in the tree leaves, the growing stage almost invariably employs a greedy recursive partitioning (RP) algorithm that finds the best possible split for each tree node as growing proceeds. RP often provides a good tree model; however, it rarely results in the optimal tree. In this paper we introduce a novel stochastic partitioning algorithm for growing regression trees, ANTP, that is based on artificial ant colony systems.[3] Because ANTP allows exploration of possible data partitioning decisions, it produces better quality tree models than those obtained by RP.

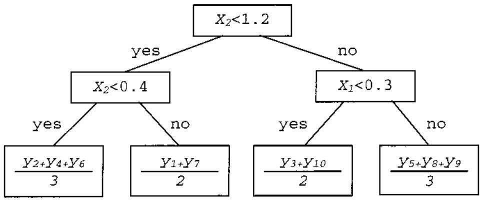

Figure 1. A regression tree trained on a sample data set consisting of 10 training cases with 2 predictor variables x_1 and x_2. During the growing stage, internal tree nodes are assigned test conditions, e.g., $x_1 < 0.3$, by a data partitioning algorithm. If a case satisfies the condition in a node, it proceeds to the left subtree. Otherwise it goes to the right subtree. Any case with a given set of x_1 and x_2 reaches a unique tree leaf. A prediction model is stored in each leaf, such that the predicted activity of a case is the average activity of the training cases that reached the same leaf. Leave-one-out predictions for a training case *i* are made by excluding y_i from the average, e.g., the leave-one-out prediction for y_4 is $(y_2 + y_6)/2$, while the regular prediction is $(y_2 + y_4 + y_6)/3$ given by the leftmost leaf.

II. METHODS

The algorithms based on artificial ant systems are inspired by the fact that real ants, using deposits of pheromone as a communication agent, are able to find the shortest path between a food source and their nest.[3] A moving ant deposits pheromone on the ground, thus marking its path. Although each individual ant moves at random, it can detect pheromone trails and follow one of them with a probability proportional to the amount of pheromone on the trail. By adding its own pheromone deposits, the ant reinforces the trail and makes it more attractive to the other ants. While all paths are initially equally probable, the shorter ones encounter more ants making round trips to the food source per time unit and, therefore, receive more pheromone. Thus, short paths become increasingly more attractive to the ants. Eventually, all ants follow the shortest trail.

In the application at hand, each ant represents a regression tree, while pheromone trails are emulated by a separate binary reference tree that represents the topological union

---

* To whom correspondence should be addressed. Phone: (610) 458-5264, ext 6570. Fax: (610) 458-8249. E-mail: sergei@3dp.com.

10.1021/ci000336s CCC: $20.00 © 2001 American Chemical Society
Published on Web 11/01/2000

of all the "ant" trees encountered in the course of the simulation. The growing process consists of recursive binary splits of the training cases contained in the tree leaves. For each leaf node one must choose a descriptor k and a value of this descriptor v_k. The cases are then partitioned into the left and right leaves on the basis of the condition

$$x_{k,i} < v_k \tag{1}$$

where $x_{k,i}$ is the value of descriptor k for case i. RP tests all possible combinations of k and i to find the best possible split $v_k = x_{k,i}$, most commonly one that minimizes the sum of squared deviations in the two resulting leaves $\sum_{i=1}^{n_L}(y_i - \bar{y}_L)^2 + \sum_{i=1}^{n_R}(y_i - \bar{y}_R)^2$, where n_L, $\bar{y}_L$, n_R, and $\bar{y}_R$ are the number of cases and the average of the target variable in the left and right leaves, respectively. In contrast, ANTP selects k and v_k randomly on the basis of a probability distribution that reflects the amount of pheromone deposited in the reference tree. During initialization, unique values $x_{k,i}$ of each predictor variable k are sorted and distributed into a small preset number of bins, so that each bin contains an approximately equal number of close values. Each node of the reference tree contains the weights w_k contributing to the probability

$$p_k = w_k / \sum_k w_k \tag{2}$$

of choosing descriptor k for splitting the data in the corresponding ant tree node, as well as weights $w_{k,n}$ for each of the bins, where $n = 1, ..., N_b$ (N_b is the number of bins). After k is chosen, the algorithm identifies the values $x_{k,i}$ that can be used for splitting, and the bins to which they belong. One of these bins, b, is chosen randomly with a probability

$$p_b = w_{k,b} / \sum_b w_{k,b} \tag{3}$$

and the best possible split value within this bin is selected. With a probability p_0, the latter choice can be replaced by a search for the overall best split value among the identified bins. In addition, the last split in each tree branch is chosen in a greedy fashion, i.e., by finding the best possible split among all possible k and i. Once the ant tree growth is completed, for each of its nodes the weights w_k and $w_{k,b}$ are updated by adding "pheromone" on the corresponding nodes of the reference tree according to the pheromone update rules defined by eqs 4a and 4b

$$w_k = w_k + \Delta w_{\mathrm{var}} g(R^2_{\mathrm{loo}}) \tag{4a}$$

$$w_{k,b} = w_{k,b} + \Delta w_{\mathrm{bin}} g(R^2_{\mathrm{loo}}) \tag{4b}$$

Δw_{var} and Δw_{bin} are the weight increments, $g(x)$ is a scaling function, and R^2_{loo} is the "leave-one-out" correlation coefficient defined by eq 5, where $\tilde{y}_i$ is the prediction for case i made without taking that case into account (see Figure 1)

$$R^2_{\mathrm{loo}} = 1 - \sum_{i=1}^{N}(y_{\mathrm{i}} - \tilde{y}_i)^2 / \sum_{i=1}^{N}(y_{\mathrm{i}} - \bar{y})^2 \tag{5}$$

and $\bar{y}$ is the average activity over all training cases. A number of ant trees are grown consecutively, and the tree that exhibits the largest R^2_{loo} is recorded. Because finding the best possible tree is a multimodal optimization problem, this process should be repeated several times to minimize the likelihood of accidental convergence to a poor local minimum.

To assess the effectiveness of the ANTP algorithm, we compare its results to those of a completely random search in regression tree space for each data set. Random tree generation starts with a single tree node containing all training cases. At each step, the training cases in a randomly selected leaf node are partitioned according to eq 1 into two subsets that are placed into two newly formed leaf nodes by randomly choosing a descriptor k and its value v_k. This process is repeated until the stopping criteria are met.

We demonstrate the use of the ANTP algorithm on three well-studied data sets: antifilarial activity of antimycin analogues (AMA),[4] binding affinities of ligands to benzodiazepine/GABA$_A$ receptors (BZ),[5] and inhibition of dihydrofolate reductase by pyrimidines (PYR).[6] For each data set we report R^2_{loo} and R^2 of the best model obtained in two series of 10 and 1000 runs of random search and ANTP algorithms. In each run of the ANTP algorithm, 2000 ant trees were grown, with the weights w_k and $w_{k,n}$ initialized to 0.01, $N_b = 10$, $p_0 = 0.5$, and the weight increments set to 0.1. The scaling function $g(x)$ in eqs 4a and 4b was set to

$$g(x) = 2^d(5^{5x} - 1)/(5^5 - 1) \tag{6}$$

where d is the depth of the tree at the given node. The choice of this function emerged from two considerations. First, the scaling function should allow distinguishing better models. The function given by eq 6 increases 5-fold as $x \equiv R^2_{\mathrm{loo}}$ increases by 0.2. Second, the deeper the node is located in the reference tree, the fewer ant trees will sample the possible splits at that node. The factor 2^d was introduced to partially compensate for that. Each random search run generated 2000 random trees and saved the one with the largest R^2_{loo}. In all calculations, the stopping criterion of a minimum of five cases per tree leaf was employed. The reported time is the total CPU time required to complete one run on a 733 MHz Pentium III processor.

All programs were implemented in the C++ programming language and are part of the DirectedDiversity[7] software suite. They are based on 3-Dimensional Pharmaceuticals' Mt++ class library[8] and are designed to run on all Posix-compliant Unix and Windows platforms. Parallel execution on systems with multiple CPUs is supported through the multithreading classes of Mt++. All calculations were carried out on a Dell workstation equipped with two 733 MHz Pentium III Intel processors running Windows NT 4.0.

III. RESULTS AND DISCUSSION

The values of R^2_{loo} and R^2 obtained after the growing stage by RP and by 10 runs of ANTP and random search are presented in Table 1. For all three data sets ANTP significantly improved the model produced by RP, thereby providing better starting points for the pruning stage. For the BZ and PYR data sets, the random search also produced better models than RP. In addition, for the PYR data set the random search performed almost as well as ANTP. As one would expect, increasing the number of runs 10-fold

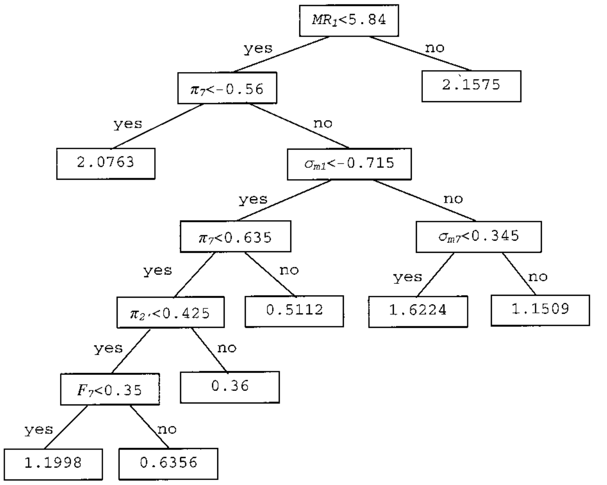

Figure 2. Regression tree model generated by the RP algorithm for the BZ data set (not pruned). The notation for the descriptors can be found in ref 5.

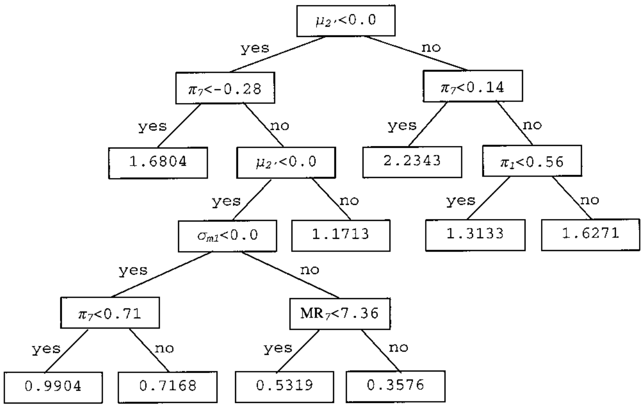

Figure 3. Regression tree model generated by the ANTP algorithm for the BZ data set (not pruned). The notation for the descriptors can be found in ref 5.

Table 1. Comparison of Models Created by 10 Runs of Random Search and ANTP Algorithms and by RP

			R^2_{loo}			R^2		
data set	*N*	M	random	RP	ANTP	random	RP	ANTP
AMA	31	53	0.681	0.736	0.821	0.776	0.829	0.877
BZ	57	42	0.667	0.499	0.728	0.755	0.652	0.806
PYR	74	27	0.626	0.423	0.634	0.712	0.618	0.721

considerably improved the outcome of the random search due to better sampling (see Table 2). In particular, the best R^2_{loo}= 0.742 found in the series of 1000 random search runs for the AMA data set exceeded the RP value of 0.736. However, the ANTP results changed only marginally. This reflects the advantages of the ANTP algorithm, since only 10 runs were sufficient to find a good tree model for each data set. To illustrate the resulting regression tree models, the best models generated by the ANTP and RP algorithms for the BZ data set are presented in Figures 2 and 3.

The execution time of a single ANTP run was 1.2, 1.6, and 2.0 s for the AMA, BZ, and PYR data sets, respectively. The overhead of maintaining the reference tree was minimal, so the random search running time was practically the same as that of ANTP. The RP algorithm is very fast; its execution time was 2−4 ms for all three data sets.

Table 2. Comparison of Models Created by 1000 Runs of Random Search and ANTP Algorithms and by RP

			R^2_{loo}			R^2		
data set	N	M	random	RP	ANTP	random	RP	ANTP
AMA	31	53	0.742	0.736	0.821	0.819	0.829	0.877
BZ	57	42	0.709	0.499	0.766	0.793	0.652	0.833
PYR	74	27	0.624	0.423	0.643	0.710	0.618	0.728

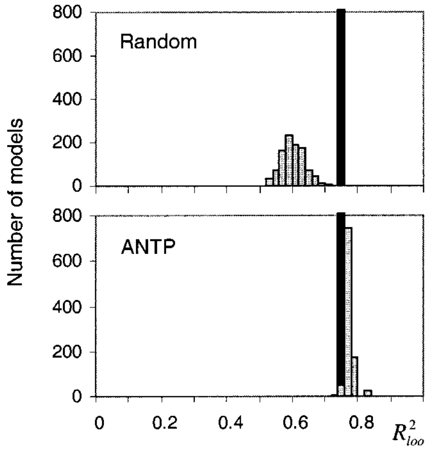

Figure 4. Histograms of the number of tree models that were produced for the AMA data set by the ANTP and random search algorithms in a series of 1000 runs and that exhibited R^2_{loo} within each of the 50 consecutive ranges of values between 0 and 1. R^2_{loo} of the model generated with the RP algorithm is marked with a black bar on both plots.

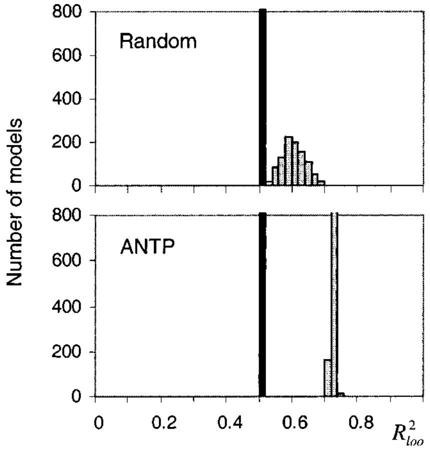

Figure 5. Histograms of the number of tree models that were produced for the BZ data set by the ANTP and random search algorithms in a series of 1000 runs and that exhibited R^2_{loo} within each of the 50 consecutive ranges of values between 0 and 1. R^2_{loo} of the model generated with the RP algorithm is marked with a black bar on both plots.

The quality of the ANTP algorithm can be assessed by comparing the distribution of the R^2_{loo} values of 1000 independently produced ANTP models to that of 1000 tree models generated by random search and to the R^2_{loo} value of the RP model. For all three data sets, the ANTP distribution exhibited a considerable shift to higher R^2_{loo} values with respect to the random search distribution, as shown in Figures 4−6. In fact, 99.3%, 99.6%, and 68.0% of the ANTP models were better than the best randomly generated model for the AMA, BZ, and PYR data sets, respectively. Therefore, a few runs of the ANTP algorithm are likely to produce a significantly better starting point for pruning than RP or random search.

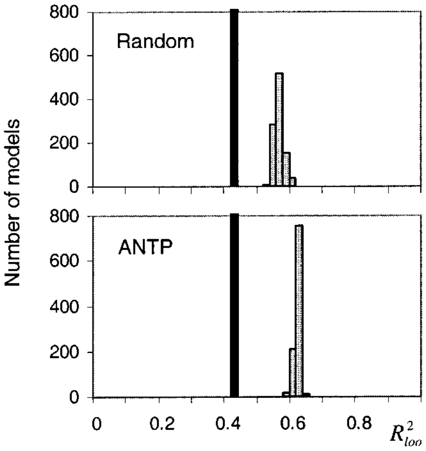

Figure 6. Histograms of the number of tree models that were produced for the PYR data set by the ANTP and random search algorithms in a series of 1000 runs and that exhibited $R^2_{\rm loo}$ within each of the 50 consecutive ranges of values between 0 and 1. $R^2_{\rm loo}$ of the model generated with the RP algorithm is marked with a black bar on both plots.

Because RP performs some optimization of the splitting criteria, it is natural to assume that RP would produce a better model than a random search. Indeed, for the AMA data set only 1 out of the 1000 random search runs resulted in a model that was slightly better than the RP model (see Figure 4). However, for the BZ and PYR data sets, the RP models ($R^2_{\rm loo} = 0.423$ and 0.499, respectively) were actually worse than virtually all 1000 models produced by random search, with the largest $R^2_{\rm loo} = 0.709$ and 0.624, respectively (see Figures 5 and 6). This implies that for some data sets recursive partitioning produces results significantly worse than those of even a short random search.

The ANTP algorithm is effective if (a) the ant trees sufficiently sample possible splitting conditions given by eq 1 at each node and (b) accumulation of "pheromone deposits" in the reference tree clearly distinguishes between "good" and "bad" splits. The first condition is met by using a sufficient number of ant trees in each ANTP run. However, since the number of nodes in the reference tree can grow exponentially with the number of nodes in the target tree model, the required number of ant trees may become prohibitively large. This may limit the application of the ANTP algorithm to reasonably small tree models (but not necessarily small data sets). The second condition can be satisfied by selecting an appropriate scaling function $g(x)$, initial weights w_k and $w_{k,n}$, and their increments. Another possible approach that improves convergence by gradually increasing the probability of choosing a descriptor or its value with relatively large corresponding weights is to use probabilities $p_k = w_k^\alpha/\sum_k w_k^\alpha$ and $p_b = w^\alpha_{k,b}/\sum_b w^\alpha_{k,b}$ instead of those given by eqs 2 and 3. The parameter α is increasing linearly (or according to some other law) with each constructed ant tree between 1 and some maximum value (e.g., 2 or 3).

One can also use other stochastic techniques, such as simulated annealing or genetic algorithms, to grow regression trees. Simulated annealing is a global, multivariate optimization technique based on the Metropolis Monte Carlo search algorithm. The method starts from an initial state, and walks through the state space associated with the problem of interest by generating a series of small, stochastic steps. An objective function maps each state into a numeric value that measures its fitness. With respect to growing regression trees, a *state* is a unique tree that satisfies the given constraints on the minimum number of training cases per leaf node, its *fitness* is the leave-one-out correlation coefficient $R^2_{\rm loo}$, and the *step* is either creation of two new leaf nodes by splitting an existing leaf node or a reverse operation that merges two sibling leaf nodes back into a single node. While downhill transitions are always accepted, uphill transitions are accepted with a probability that is proportional to $p = e^{-\Delta E/K_B T}$, where ΔE is the energy (fitness) difference between the two states. Boltzmann's constant, K_B, is used for scaling purposes, and T is an artificial temperature factor used to control the ability of the system to overcome energy barriers. Simulated annealing is effective only when the energy surface is relatively smooth and the steps are local in nature. However, when applied to regression trees, each step can have a dramatic impact on the fitness function. This leads to very poor convergence of the algorithm. In fact, the distribution of the resulting $R^2_{\rm loo}$ values is very close to that of a random search, as shown in Figure 7 for the AMA data set. Thus, simulated annealing does not provide a significant benefit with respect to the quality of the resulting models, and in addition it is computationally expensive.

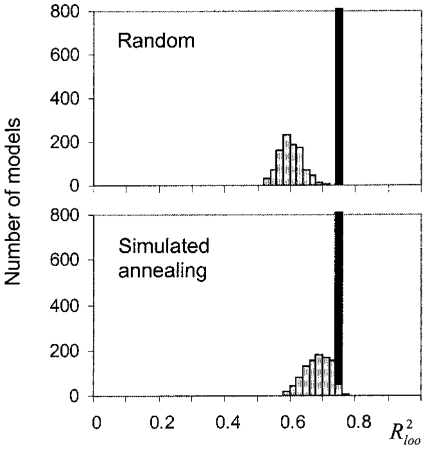

Figure 7. Histograms of the number of tree models that were produced for the AMA data set by the simulated annealing and random search algorithms in a series of 1000 runs and that exhibited $R^2_{\rm loo}$ within each of the 50 consecutive ranges of values between 0 and 1. $R^2_{\rm loo}$ of the model generated with the RP algorithm is marked with a black bar on both plots. In each run of the simulated annealing algorithm the temperature is reduced according to a Gaussian cooling schedule, comprising 30 consecutive temperature cycles with 2000 annealing steps performed at a constant temperature during each cycle (other cooling schedules, such as linear, exponential, and Lorentzian, can also be used). Each cycle starts with the tree with the largest $R^2_{\rm loo}$ encountered during the previous cycle. This strategy proved to work better than starting the cycle with the last tree of the previous cycle. To circumvent the difficulty of selecting an appropriate value for K_B, in our implementation this is not a true constant but is adjusted on the basis of an estimate of the mean transition energy. In particular, at the end of each transition, the mean transition energy is updated, and the value of K_B is adjusted so that the acceptance probability for a mean uphill transition at the final temperature is 0.1%.

In conclusion, we developed a novel stochastic algorithm, ANTP, for binary data partitioning. We also demonstrated that for some data sets the conventional recursive partitioning algorithm is inferior to random search. While the ANTP algorithm may not necessarily converge to the optimal tree, it offers clear benefits in the quality of initial data partitioning for regression tree-based QSAR models, as compared to RP.

ACKNOWLEDGMENT

We thank Dr. Raymond F. Salemme of 3-Dimensional Pharmaceuticals, Inc. for his insightful comments and support of this work.

REFERENCES AND NOTES

(1) Breiman, L; Friedman, J. H.; Olshen, R. A.; Stone, C. J. *Classification and Regression Trees*; Wadsworth Int. Group: Belmont, CA, 1984.

(2) King, R. D.; Hirst, J. D.; Sternberg, M. J. E. *Appl. Artif. Intell.* **1995**, *9*, 213−233. Chen, X.; Rusinko, A., III; Young, S. S. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 1054−1062.

(3) Dorigo, M.; Gambardella, L. M. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53−66 and references therein.

(4) Selwood, D. L.; Livingstone, D. J.; Comley, J. C. W.; O'Dowd, A. B.; Hudson, A. T.; Jackson, P.; Jandu, K. S.; Rose, V. S.; Stables, J. N. *J. Med. Chem.* **1990**, *33*, 136−142.

(5) Maddalena, D. J.; Johnson, G. A. R. *J. Med. Chem.* **1995**, *38*, 715−724.

(6) Hirst, J. D.; King, R. D.; Sternberg, M. J. E. *J. Comput.-Aided Mol. Des.* **1994**, *8*, 405−420.

(7) Agrafiotis, D. K.; Bone, R. F.; Salemme, F. R.; Soll, R. M. U.S. Patents 5,463,564, 1995; 5,574,656, 1996; 5,684,711, 1997; 5,901,069, 1999.

(8) Copyright 1994−2000 3-Dimensional Pharmaceuticals, Inc.

CI000336S