A Genetic Algorithm for Expert System Rule Generation

John C. Determan  
Idaho National Engineering and Environmental Laboratory  
P.O. Box 1625  
Idaho Falls, ID 83415-2211  
jcd@inel.gov  
(208) 526-2888

James A. Foster  
Computer Science Department  
University of Idaho  
P.O. Box 1010  
Moscow, ID 83844-1010  
foster@cs.uidaho.edu  
(208) 885-7062

Abstract

We applied genetic algorithms to fuzzy rule generation to compute expert system rules from data. We have attempted to improve on existing techniques for the automatic generation of fuzzy logic expert system rules with a method we call genetic data clustering (GDC). A genetic algorithm groups training data points by their degree of similarity, and fuzzy logic expert system rules are formed from these groups. We designed mutation and crossover operators, and an objective function specifically for the task of data clustering. We employed variable mutation probabilities and variable chromosome lengths. We tested the algorithm against two real-world data sets. The GDC algorithm achieved average performance compared to a suite of classification algorithms found in the literature. The rule representation format was the primary limitation to performance.

INTRODUCTION

We developed a new genetic algorithm (GA) based data clustering algorithm for generating fuzzy logic (Zadeh, 1965) expert system rules. An advantage of performing data clustering genetically is that GAs are probabilistic, and ambiguous data points have some probability of being assigned to any cluster. With a good objective function, and appropriate genetic operators, natural selection should find good data clusters. However, grouping problems are difficult for GAs, due to tight constraints on how the genome can be mutated and crossed. In grouping problems a GA must neither duplicate nor remove alleles from the genome, and ordinary genetic operators do not guarantee these properties. Therefore, genetic data clustering could be expected to shed light on an area of interest in the Evolutionary Computing community: combinatorial or grouping problems.

PREVIOUS WORK

The subtractive clustering algorithm (Chiu, 1994), based on the mountain clustering method (Yager and Filev, 1994) can generate fuzzy logic expert system rules from training data. The clustering process locates potential data clusters and estimates their means and standard deviations in each dimension of the training data. Backpropagation against the training data refines these means and standard deviations. The technique then forms a fuzzy logic rule from each cluster, and estimates the rule output as the mean of the correct classification for each cluster.

OUR WORK

We introduce several innovations that improve both the speed of the rule generation process and the accuracy of the generated rules (Determan 2000). Our work also contributes to the application of GAs to combinatorial problems in general. Our genetic data clustering algorithm first locates data points in the training data that belong together, it then calculates exact means and standard deviations over these clusters, and finally, it builds fuzzy rules from the cluster means and standard deviations. We employ subtractive clustering to initialize our population of solutions, but do away with backpropagation altogether. We designed crossover and mutation operators specifically for data clustering. We used self-adaptive techniques, including variable length chromosomes and a variable probability of mutation. We begin by discussing some fundamental issues in automatic rule generation. We then present the genetic data clustering approach to automatic rule generation. Afterwards, we describe our evaluation of this new approach and present data. Finally, we present conclusions and suggestions for future research.

GENETIC DATA CLUSTERING

Generating fuzzy rules from training data is a two-step process. The first step is to locate data clusters in the training data. The second step is to optimize the clusters
and use parameters derived from the data clusters to form fuzzy rule membership functions. A cluster of data points represents an approximate relationship from input to output, and this relationship can be represented as a rule using fuzzy logic.

We modified the automatic rule generation process outlined above to improve both speed of the rule generation algorithm and accuracy of the resulting rules. We replaced the combined steps of data clustering and backpropagation with a new genetic data clustering (GDC) algorithm. We used subtractive clustering in the initialization step of GDC, and completely eliminated backpropagation. After the algorithm identifies a set of clusters in the training data, it calculates the means and standard deviations in each dimension of the training data; finally, the algorithm constructs fuzzy rules from these values.

**GENETIC OPERATORS FOR DATA CLUSTERING**

Our initialization routine used subtractive clustering to estimate cluster membership for a single solution. The estimated cluster means were used to calculate the distance of each point from the estimated cluster centroids. Points were assigned to each cluster probabilistically, with the probability of assignment to a cluster inversely proportional to the square of distance between a point and the cluster centroid. Any given point had the greatest probability of being associated with the closest cluster centroid, but had some probability of being associated with any cluster. A population of proposed solutions initialized in this manner will tend to include varied estimations of the membership partition of the data points, each close to the correct solution, and each differing from each other primarily in the membership of questionable points from regions of overlap. Finally, we used the mutation operator, described below, to clone the entire population from this initial solution.

Our initial mutation operator swapped data points between clusters of a given solution. Specifically, we select two clusters uniformly at random, then randomly choose a data point from each cluster with probability equal to the squared distance from the cluster centroid, and swapped them. A variety of constant and variable probabilities of mutation were tested. The application of genetic data clustering to large data sets revealed weaknesses in the mutation technique. The swap mutator described above was neither sufficiently focused nor “intelligent” enough to deal effectively with large data sets. Tests on the LANDSAT data indicated the classification error rate of the generated rules was about double that reported in the literature. These tests had indicated that the mutation operator was largely responsible for finding new and improved solutions, so we focused our attention here. It was determined that while mutation was still occurring at a high rate as the error rate in the evolved rules leveled off, equal numbers of correctly classified points and incorrectly classified ones were being swapped out of their clusters. A kind of equilibrium was reached and no further progress was possible. The original method used only distance from the cluster center to choose which points to swap. New information was needed to improve the equilibrium error.

<table>
<thead>
<tr>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
<th>Cluster M</th>
<th>Cluster sizes</th>
<th>k fit parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_1 ) points</td>
<td>( N_2 ) points</td>
<td>( N_3 ) points</td>
<td>( N_M ) points</td>
<td>( (N_1, N_2, N_3 \ldots N_M) )</td>
<td>( (b_1, b_2, b_3 \ldots b_k) ), ( k = # ) of components in data</td>
</tr>
</tbody>
</table>

**Figure 1.** Genetic representation for the GDC algorithm. The representation has variable size, and this size depends on both the number of training points, \( N \), and the number of clusters in a given solution, \( M \). Cluster membership is indicated by sequences of data point identifiers, while a separate sequence of integers represents cluster sizes. The \( k \) additional parameters at the end of the chromosome are needed in a modified form of the rule representation, described below.
rate. The fundamental problem, of course, is regions of overlap in the training data. When there are thousands of data points, the algorithm has to focus clearly on these regions. We used nearest neighbor information to determine probability distributions of a point being misclassified.

The simple data classification procedure known as “k nearest neighbors” determines the k “closest” data points in a training set of known classifications to any data point of unknown classification. The method uses a majority vote of these neighbors to classify any unknown data point. However, nearest neighbor information can also generate a probability distribution for cluster membership, which is useful in stochastic algorithms such as the GDC. This distribution can induce a variable mutation probability that allows the algorithm to focus on regions of greatest difficulty. In homogenous regions there is essentially zero probability of a point being misclassified, while in regions of overlap the probability of misclassification is defined by the proportions of the different classes in the region. In our refined scheme, we use the misclassification probability, $P_{\text{miss}}$, as one of two parameters that define a variable mutation rate.

In a reasonably homogenous region, most of a point’s nearest neighbors will have very similar classification values, the sum of exponential terms divided by $k$ will approach unity, and the misclassification probability will go to zero (plus a small random value $r$). At the opposite extreme, an isolated point distinct from its neighbors, the misclassification probability will go to about 0.5 (again, plus $r$). The small random term ($0.1 \leq r \leq 0.3$) is to ensure that all data points have some probability of being swapped to another cluster.

Our refined algorithm also estimated a second probability as part of the variable mutation rate. We call the second probability the retention probability, $P_{\text{ret}}$, and it estimates how well the classification of a given data point agrees with the classification of the cluster to which it belongs. When the classifications of a data point and the cluster to which it belongs are similar the retention probability approaches unity, and in the opposite extreme will go to zero. Both probabilities are normalized by the same “significance factor,” defined to be unity for integral classification values, and 10% of the full scale classification range for continuous values.

To simplify “calibration” of the mutation operator, the adjustable terms were restricted to the misclassification probability. Figure 2 shows both estimated probabilities as functions of “significance units.” The misclassification probability curve was derived for one hypothetical situation, and would vary in different particular situations. The main curve represents the nearest neighbor term and the error bar the maximum possible value attainable by the addition of the small random term $r$. The retention probability curve is an absolute curve for any point that differs by some number of significance units from the cluster associated with it. Both curves may have distinct abscissa values during any particular calculation, such that the intersection does not have any particular significance. It is important that the retention probability may approach a value of unity, but the misclassification probability would have only a very slim chance of doing so.

![Figure 2](image-url) Comparison of the probabilities of misclassification and retention as a function of normalized classification differences (significance units). The probability of misclassification curve is based on average significance units for a particular hypothetical situation, while the probability of retention curve is absolute and valid for any point and cluster.

The swap mutator steps through the chromosome testing each point in turn by comparing the probabilities described above and swapping the point in question to a randomly selected cluster if the point’s retention probability is less than its misclassification probability. The retention probability is dynamic and must be calculated at every step. While the misclassification probability involves some expensive calculations, it is static (for a given value of $k$), calculated once only, and stored with each data point.

Data clustering requires a crossover method that, to the extent possible, picks whole clusters from two parents, resolves membership conflicts between clusters from different parents that possess a common data point, and ensures that the resulting child genome represents a valid and complete set of data points. The algorithm randomly selects clusters from two parents and copies them into a child. With each cluster selection, the algorithm removes a cluster from the donor parent and the points of this cluster from the other parent. This algorithm ensures that only unused data points are available for inclusion in the child at each step, and that all data points are included in the child. We limited the least number of points allowed in a cluster because a cluster with only one data point has standard deviation of zero. This condition causes an error, since the standard deviations occur as divisors in the fuzzy membership functions. While there is little chance that problems with large data sets will produce single-point clusters, it is safest to establish a limit, and smaller data sets require this bound to prevent errors. We chose three points per cluster as a reasonable minimum. For small data sets, a minimum value of two points per
cluster sometimes resulted in two-point clusters where at least one dimension in the data points possessed the same value. This situation resulted in a standard deviation of zero in that dimension, resulting in a divide-by-zero error. It should be noted that cluster crossover is similar in effect to multiple applications of order crossover (Goldberg, 1989), except that the present implementation acts more strongly to preserve existing clusters, and locate those points that are only loosely bound to these clusters.

Two factors are primary in designing an objective function for data clustering: training set performance and the cluster validity index. Training set performance is defined as the ratio of correctly classified training data points to the total number of training points. While training set performance is a useful component of the objective function, the correct number of clusters is generally unknown, so optimizing only this parameter will tend to subdivide proper clusters and overfit of the training data. Therefore, the objective function should also include a cluster validity index. The Xie-Beni index (Xie and Beni, 1991) evaluates the validity of fuzzy clusters, with a small index indicating better clustering than a large value. Table 1 shows the results of tests performed to compare the performance of the objective function either with or without the Xie-Beni component. We repeated each of these tests 20 times. The case using the Xie-Beni index yielded considerably lower error rates than the case without the Xie-Beni index. The Iris data (described below) were used to test the objective function because the data set was small enough to perform numerous repetitions in a reasonable amount of time.

Our objective function combined these two metrics:

\[
\text{Objective}(\text{training set performance}, Xie - Beni index) = \text{training set performance}^2 + \exp \left( - \frac{Xie - Beni index}{c} \right)
\]

where \(c\) is a scaling constant that gave best performance at a value of 3. Figure 3 shows the results of tests on the Iris data with values for the scaling factor \(c\) set to 1, 2, 3, 4, 5, and 10. We repeated each of these tests 20 times. While the dependence is not strong, the lowest minimum, average, and maximum error rates were all obtained for the test with \(c\) set to three. We square the training performance score to intensify the difference between high and low scores.

To improve the accuracy of the generated rules, we modified the fuzzy rule membership function to contain a factor \(b\) in the denominator, referred to as the “fit parameter.” This factor improved the fit of the membership function to the data. The best setting for this parameter varied from one data set to the next, with the most common values being close to unity or 0.5. While this modification typically (about 75% of the time) improved performance by about 10%, tests performed on the LANDSAT satellite image data (described below) showed considerable degradation in performance as the number of components in each data point increased. Therefore, we gave each component in the data a distinct fit parameter, and the algorithm determined these genetically. We added extra genes to the genome to hold these parameters (see Figure 1), initialized randomly to the integer values 100 or 50. We also modified the swap mutator to randomly increment or decrement these parameters by 5, at a probability of mutation of 0.1, and we modified the cluster crossover operator to perform uniform crossover on this segment. The algorithm divided the parameter values by 100 to scale them back to the proper range before using them in the fuzzy matching strength calculation.

**Table 1.** This table shows the results of tests performed to compare the performance of the objective function either with or without the Xie-Beni component. We repeated each of these tests 20 times. The case using the Xie-Beni index yielded considerably lower error rates than the case without the Xie-Beni index.

<table>
<thead>
<tr>
<th>Test</th>
<th>Min. Error Rate %</th>
<th>Ave. Error Rate %</th>
<th>Max. Error Rate %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xie-Beni component in objective function</td>
<td>2.7</td>
<td>5.1</td>
<td>11</td>
</tr>
<tr>
<td>No Xie-Beni component in objective function</td>
<td>4.0</td>
<td>7.0</td>
<td>13</td>
</tr>
</tbody>
</table>

**Figure 3.** The results of calculations on the Iris data to determine an appropriate value for the scaling factor \(c\) (where \(c = 1, 2, 3, 4, 5, \) and 10). We repeated each test 20 times. While the dependence is not strong, we obtained the lowest minimum, average, and maximum error rates for the test with \(c = 3\).

The results of these modifications are illustrated in Figure 4. The label “none” indicates that the fit parameter was not present, “fixed” means that the parameter was a constant (0.55, in this calculation) and the same for all components of the data, and “varied” means that each component of the data had distinct, genetically determined, fit parameters. Another way of stating these results is that the parameter \(b\) makes the fitness function self-adaptive, and this considerably improves the algorithm.
TEST PROCEDURES AND RESULTS

We tested the automatic rule generation algorithm on two data sets and discuss the results of those tests in this section. The data were preprocessed in one of two ways, by either centering and reducing the data to unit variance, or performing a principal components analysis on the data. To perform valid comparison between our method and others, we obtained and used two data sets available in the literature in preprocessed form. Table 2 summarizes the properties of the data sets that we used. We used two data sets from a comparative study of neural network derived classification algorithms, the Enhanced Learning for Evolutive Neural Architecture (ELENA) project (Aviles-Cruz, et al. 1995). The ELENA data contained several large data sets with numeric feature values, making them appropriate for use in our algorithm. The testing methodology used in the ELENA project (Blayo et al., 1995) was to partition the full data sets in half, and to use one half of the data for training and the other half for testing. In order to reduce dependence of the results on any particular choice of training data, we partitioned the data five separate times, and trained and tested the methods being compared against each partition. The reported results, reproduced in the plots at the end of this chapter, are in the form of classification errors averaged over the five different partitions of the data set. We constructed confidence intervals (95%) for the minimum and maximum error rates, as in the ELENA project, to provide a statistical bound on the expected performance range of the GDC algorithm.

![Comparison of % error rates between three calculations with slight variations in the rule representation form.](image)

**Figure 4.** Comparison of % error rates between three calculations with slight variations in the rule representation form. The variations refer to the fit parameter in the denominator of the fuzzy membership functions. “None” indicates that the parameter was not present, “fixed” that the parameter was a constant (0.55, in this calculation) and the same for all components of the data, and “varied” that each component of the data had distinct fit parameters, with values that were genetically determined.

Data sets used to test classification algorithms generally consist of data points labeled by some integral set of values. It is a useful benchmark to treat these labeled data as already clustered (by classification) and form fuzzy rules by taking the average and standard deviations of the data points in these clusters. These rules represent the results that the GDC (Genetic Data Clustering) algorithm could obtain with “perfect” clustering on a given data set. In other words, these results represent the order of magnitude performance obtainable by any reasonable algorithm, given the form of the rule representation. To improve performance, one would have to address the form of the rule representation. The rule representation – Gaussian membership functions – imposes a particular inductive bias on the solution. Hence the “ideal” solution is only ideal with respect to this inductive bias.

Table 3 presents all of the genetic parameter settings used in the GDC calculations. As discussed above, we used the cluster crossover and swap mutation operators for the GDC. A variety of standard scaling and selection functions (Goldberg, 1989) were tested, and the combination of linear scaling and tournament selection yielded the best performance. Tournament selection involved two individuals at a time, and the linear scaling technique used in GALib (Wall, 1996), the GA code used in this project, is the one due to Goldberg (Goldberg, 1989) with default parameter value of 1.2. We performed the tests on a Pentium II PC (450 MHz, 128 MB RAM).

**Experimental Results**

We first present comparisons to results obtained in the ELENA project, using the Iris and LANDSAT data sets to rules produced from 4 variations of our method:

- “Initial GDC” calculations use the initial genetic calculations with uniform fit parameter values
- “Initial Ideal GDC” calculations are the “ideal” rule sets with uniform fit parameter values
- “Improved GDC” calculations use the genetic calculations with variable fit parameter values
- “Improved Ideal GDC” calculations are the “ideal” rule sets with non-uniform fit parameter values (from the “Improved GDC” calculation results)

**Table 2. Properties of the Data Sets Used in our Tests.**

<table>
<thead>
<tr>
<th>Name</th>
<th># Features Used</th>
<th># Examples</th>
<th>Type of Preprocessing</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Data</td>
<td>4</td>
<td>150</td>
<td>Centered &amp; Reduced</td>
<td>Small, but frequently used, classification data set</td>
</tr>
<tr>
<td>LANDSAT Data</td>
<td>4, 8</td>
<td>6000</td>
<td>Principal Components</td>
<td>Large classification data set</td>
</tr>
</tbody>
</table>
Iris Data

We compared our GDC algorithm to three benchmark learning algorithms (in order of increasing error rate): the Gaussian Quadratic Classifier (GQC), the Iterative Radial Vector Quantization (IRVQ), and the Restricted Coulomb Energy (RCE) algorithms. The results from these algorithms span the range of results in the ELENA project for the Iris data, showing the best, worst and average performance. The fact that the GQC performs the best on the Iris data is an indicator that the data have a Gaussian distribution, and should also indicate that good performance is possible with the GDC (Gaussian membership functions are used).

Table 3. GDC Genetic Parameter Settings.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Genetic algorithm</td>
<td>Steady-state GA</td>
</tr>
<tr>
<td>Replacement parameter</td>
<td>Population size – 1</td>
</tr>
<tr>
<td>Probability of crossover</td>
<td>0.8</td>
</tr>
<tr>
<td>Population size</td>
<td>100</td>
</tr>
<tr>
<td>Crossover operator</td>
<td>Cluster crossover</td>
</tr>
<tr>
<td>Mutation operator</td>
<td>Swap mutation</td>
</tr>
<tr>
<td>Selection operator</td>
<td>Tournament, 2 selections</td>
</tr>
<tr>
<td>Scaling operator</td>
<td>Linear, Goldberg, 1.2</td>
</tr>
<tr>
<td>Maximum generations</td>
<td>100 (Iris) / 200 (LANDSAT)</td>
</tr>
</tbody>
</table>

Figure 5 presents results obtained with the Iris data. The average error rate of the “Initial GDC” calculations agrees exactly with the “Initial Ideal GDC” value, and the “Improved GDC” calculation average value slightly exceeds them both. Hence, the genetic clustering procedure worked very well on the Iris data. Due to the close agreement in all variants of the GDC calculation on the Iris data, we did not perform the “Improved Ideal GDC” calculation. The GDC algorithm performed above average among the classification algorithms tested in the ELENA project on the Iris data.

LANDSAT Satellite Image Data

As before, we selected for comparison the results with the lowest, average and highest error rates. These methods are, in order of increasing error rate, the k-Nearest Neighbors (kNN), the Gaussian Quadratic Classifier (GQC), and the Restricted Coulomb Energy (RCE) algorithms. We present the comparisons for two variants of the calculations: 1) calculations using the first four principal components of the data (referred to as the four component calculation), and 2) calculations using the first eight principal components of the data (referred to as the eight component calculation).

Figure 6 presents the error rate comparisons for the four component calculations. We took the fit parameter values for the “Improved Ideal GDC” calculation from the rules found during the actual calculations. Figure 6 shows that for both the initial and improved representations the actual calculations significantly outperform the rule sets based on ideal clustering. The “Improved GDC” calculations achieved considerably lower error rates than the “Initial GDC”. Together these calculations indicate that the genetic procedure is working as well as the rule representation will allow, and even a little better. The “Improved GDC” performed just below the average value in the ELENA project on the four component LANDSAT data.

Figure 7 presents the error rate comparisons for the eight component calculations. Figure 7 also shows that the actual rule sets performed only slightly better than the ideal rule sets, whereas the “Improved GDC” calculations performed much better than the “Initial GDC calculations.” The fact that eight components are being used makes it more important to establish correct fit parameter values on each component since no single fit

![Figure 5](image-url)  
**Figure 5.** Comparison of % error rates for various classifiers studied in the ELENA project vs. the GDC algorithm on the Iris data. Holdout testing was used with 5 separate partitions into equally sized test and training sets. Sets of 3 data points represent the minimum, average and maximum error rates, respectively, over the 5 partitions. The error bars represent 95% confidence intervals on the minimum and maximum error rates. The GQC algorithm had the lowest error rate, the RCE the highest, and the IRVQ represents average results. Our GDC algorithm performs somewhat above average, and agrees well with the “Initial ideal GDC” performance.
Figure 6. Comparison of % error rates for various classifiers studied in the ELENA project vs. the GDC algorithm (including two variations) on the satellite image data using the first 4 principal components. Holdout testing was used with 5 separate partitions into equally sized test and training sets. Sets of 3 data points represent the minimum, average and maximum error rates, respectively, over the 5 partitions. The error bars represent 95% confidence intervals on the minimum and maximum error rates. For both the initial and improved representations, the actual calculations significantly outperform the rule sets based on ideal clustering. The “Improved GDC” calculations achieved considerably lower error rates than the “Initial GDC” calculations. With respect to the ELENA data, the kNN algorithm had the lowest error rate, the RCE the highest, and the GQC represents average results. Our GDC algorithm achieved slightly below average performance on the satellite image data.

CONCLUSIONS AND FUTURE DIRECTIONS
We have presented a new method for automatically generating expert system rules from data. Innovations include a genetic data clustering algorithm that involves crossover and mutation operators specifically designed for the task of data clustering. In addition, we used self-adaptive techniques, including variable length chromosomes and a variable probability of mutation.

The primary component of the genetic data clustering algorithm is the swap mutation operator that uses a pair of heuristically calculated probabilities to select data points to swap from one cluster to another. These probabilities are based on nearest neighbor calculations and cluster
average classification values, and help to focus the attention of the mutation operator on data points most likely to be found in regions of overlap in the training data.

The cluster crossover operator, to the degree possible, acts to contribute whole clusters from each parent to the child solution under construction. Thus, the GDC algorithm maintains appropriate building blocks under cluster crossover.

Tests have shown that performance of the GDC algorithm is mixed. Specifically:

- We used the Fisher Iris data, and LANDSAT satellite image data to compute ideal classification rules by a simple manual procedure. We used these rules to show that the GDC algorithm computes rule sets that are as good as the rule representation format will allow.

- An improvement to the rule representation format considerably reduced the error rate of the GDC calculations on the satellite image data, and to a much lesser extent on the Iris data. This observation confirms that a primary limitation to performance of the GDC algorithm is the rule representation format.

- Comparison of our results on the above mentioned data sets with results from the ELENA project indicate that classification rules generated by our method achieve about average results, but also indicate that the rule representation, not the performance of the GDC algorithm, is the primary limitation to performance.

It is notable that both the GQC and GDC algorithms work well on the Iris data, as alluded to previously, and likewise both do less well on the satellite image data. Both methods use Gaussian functions, indicating a similar inductive bias in these methods. The satellite image data is likely less amenable to classification by Gaussian functions. Future researchers should therefore investigate different rule representations to see what different results they obtain with different inductive biases.

Interested researchers could experiment with self-adaptive behavior by making the scaling factor in the probability of misclassification a genetic value of the representation. After experimenting with a few constant values, we set this parameter to a constant value of two. However, what may be more valuable than determining an appropriate constant value for this parameter is seeing if varying this value as the problem evolves is of benefit. Evolving this parameter will create a dynamic relationship between the two probabilities that trigger the occurrence of a mutation, possibly increasing the beneficial effects of swap mutation.

ACKNOWLEDGMENTS
This document was prepared for the U.S. Department of Energy Assistant Secretary for Environmental Management under DOE Idaho Operations Office Contract DE-AC07-99ID13727. The authors would like to thank Dr. Tim Roney, and Dr. John Slater for their support of this work, and also thank Mr. Greg Becker for lending his expertise in the field of nondestructive assay.

REFERENCES


