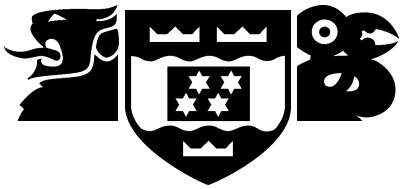
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

Multiclass Object Classification Using Genetic Programming

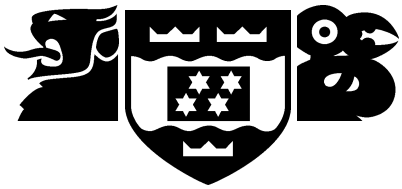
Mengjie Zhang, Will Smart

Technical Report CS-TR-04/2
Feb 2004

School of Mathematical and Computing Sciences
Victoria University
PO Box 600, Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

Multiclass Object Classification Using Genetic Programming

Mengjie Zhang, Will Smart

Technical Report CS-TR-04/2
Feb 2004

Abstract

We describe an approach to the use of genetic programming for multi-class object classification problems. Rather than using fixed static thresholds as boundaries to distinguish between different classes, this approach introduces two methods of classification where the boundaries between different classes can be dynamically determined during the evolutionary process. The two methods are centred dynamic class boundary determination and slotted dynamic class boundary determination. The two methods are tested on four object classification problems of increasing difficulty and are compared with the commonly used static class boundary method. The results suggest that, while the static class boundary method works well on relatively easy object classification problems, the two dynamic class boundary determination methods outperform the static method for more difficult, multiple class object classification problems.

Keywords Genetic programming, genetic algorithms, dynamic class boundary determination, object recognition.

Author Information

Mengjie Zhang is an academic staff member in computer science and Will Smart is a postgraduate student in computer science. Both authors are in the School of Mathematical and Computing Sciences, Victoria University of Wellington, New Zealand.

Multiclass Object Classification Using Genetic Programming

Mengjie Zhang and Will Smart

School of Mathematical and Computing Sciences
Victoria University of Wellington,
P. O. Box 600, Wellington, New Zealand,
Email: {mengjie,willsmart}@mcs.vuw.ac.nz

Abstract. We describe an approach to the use of genetic programming for multi-class object classification problems. Rather than using fixed static thresholds as boundaries to distinguish between different classes, this approach introduces two methods of classification where the boundaries between different classes can be dynamically determined during the evolutionary process. The two methods are centred dynamic class boundary determination and slotted dynamic class boundary determination. The two methods are tested on four object classification problems of increasing difficulty and are compared with the commonly used static class boundary method. The results suggest that, while the static class boundary method works well on relatively easy, linearly separable object classification problems, the two dynamic class boundary determination methods outperform the static method for more difficult, multiple class object classification problems.

1 Introduction

Classification tasks arise in a very wide range of applications, such as detecting faces from video images, recognising words in streams of speech, diagnosing medical conditions from the output of medical tests, and detecting fraudulent credit card fraud transactions [1, 2]. In many cases, people (possibly highly trained experts) are able to perform the classification task well, but there is either a shortage of such experts, or the cost of people is too high. Given the amount of data that needs to be classified, automated classification systems are highly desirable. However, creating automated classification systems that have sufficient accuracy and reliability turns out to be very difficult.

GP research has considered a variety of kinds of classifier programs, using different program representations, including decision tree classifiers and classification rule sets [3]. Recently, a new form of classifier representation - numeric expression classifiers - has been developed using GP [4–7]. In these years, this form has become the "standard form" of GP and has been successfully applied to some real world classification problems such as detecting and recognising particular classes of objects in images [5, 6, 8, 9], demonstrating the potential of GP as a general method for classification problems.

Numeric expression GP classifiers model a solution to a classification problem in the form of a mathematical expression, using a set of arithmetic and mathematical operators, possibly combined with conditional/logic operators such as the "if-then-else" structures commonly used in computer programs.

The output of a numeric expression GP classifier is a numeric value that is typically translated into a class label. For the simple binary classification case, this translation can be based on the sign of the numeric value [5, 10–12]; for multi-class problems, finding the appropriate boundary values to separate the different classes is more difficult. The simplest approach – fixing the boundary values at manually chosen points – often results in unnecessarily complex programs and could lead to poor performance and very long training times [4, 7, 9].

The goal of this paper is to develop better classification strategies in GP for multi-class object classification problems. The main focus is on the translation of the numeric output of a genetic program classifier into class labels. Rather than using manually pre-defined boundary values, we will consider new methods which allow each genetic program to use a set of dynamically determined class boundaries. We will compare the dynamic methods with the current static (manually defined) method on a number of image classification problems of increasing difficulty.

This paper is organised as follows. Section 2 describes the overall GP approach for object classification problems. Section 3 describes the class translation rules. Section 4 presents the four image classification problems used in this approach. Section 5 presents the experimental results and section 6 gives the concluding remarks.

2 The GP Approach for Object Classification

In this approach, we used the numeric expression based tree-structure to represent genetic programs. The ramped half-and-half method was used for generating the programs in the initial population and for the mutation operator. The proportional selection mechanism and the reproduction, crossover and mutation operators were used in the learning and evolutionary process.

In the remainder of this section, we address the other aspects of the GP learning/evolutionary system: (1) Determination of the terminal set; (2) Determination of the function set; (3) Construction of the fitness measure; and (4) Selection of the input parameters and determination of the termination strategy.

2.1 Terminals

For object classification problems, terminals generally correspond to image features. Some conventional approaches to image recognition usually use high level, domain specific features of images as inputs to a learning/classification system, which generally involves a time consuming feature selection and a hand-crafting of feature extraction programs. In this approach, we used pixel level, domain independent statistical features (referred to as pixel statistics) as terminals and

we expect the GP evolutionary process can automatically select features that are relevant to a particular domain to construct good genetic programs.

Four pixel statistics are used in this approach: the average intensity of the whole object cutout image, the variance of intensity of the whole object cutout image, the average intensity of the central local region, and the variance of intensity of the central local region.

Since the range of these four features are quite different, we linearly normalised these feature values into the range [-1, 1] based on all object image examples to be classified.

In addition, we also used some constants as terminals. These constants are randomly generated using a uniform distribution. To be consistent with the feature terminals, we also set the range of the constants as [-1, 1]. Unlike the feature terminals where the same feature usually has different values for different object images, the constant terminals will remain unchanged for all object images through the whole evolutionary process.

2.2 Functions

In the function set, the four standard arithmetic and a conditional operation was used to form the function set:

$$FuncSet = \{+, -, *, /, if\} \tag{1}$$

The $+$, $-$, and $*$ operators have their usual meanings — addition, subtraction and multiplication, while / represents "protected" division which is the usual division operator except that a divide by zero gives a result of zero. Each of these functions takes two arguments. The *if* function takes three arguments. The first argument, which can be any expression, constitutes the condition. If the first argument is negative, the *if* function returns its second argument; otherwise, it returns its third argument.

2.3 Fitness Function

We used classification accuracy on the training set of object cutout images as the fitness function. The classification accuracy of a genetic program classifier refers to the number of object cutout images that are correctly classified by the genetic program classifier as a proportion of the total number of object images in the training set. According to this design, the best fitness is 100%, meaning that all object images have been correctly recognised without any missing objects or any false alarms for any class.

To calculate the classification accuracy of a genetic program, one needs to determine how to translate the program output to a class label. This is described in section 3.

Table 1. Parameters used for GP training for the four datasets.

Parameter Kinds	Parameter Names	Shape1	shape2	coin1	coin2
Search Parameters	population-size	300	300	300	500
	initial-max-depth	3	3	3	3
	max-depth	5	5	6	8
	max-generations	50	50	50	50
	object-size	16×16	16×16	70×70	70×70
Genetic Parameters	reproduction-rate	20%	20%	20%	20%
	cross-rate	50%	50%	50%	50%
	mutation-rate	30%	30%	30%	30%
	cross-term	15%	15%	15%	15%
	cross-func	85%	85%	85%	85%

2.4 Parameters and Termination Criteria

The parameter values used in this approach are shown in table 1.

In this approach, the learning/evolutionary process is terminated when one of the following conditions is met:

- The classification problem has been solved on the training set, that is, all objects of interest in the training set have been correctly classified without any missing objects or false alarms for any class.
- The accuracy on the validation set starts falling down.
- The number of generations reaches the pre-defined number, *max-generations*.

3 Translation Rules in Classification

As mentioned earlier, each evolved genetic program has a numeric output value, which needs to be translated into class labels. The methods which perform this translation are referred to as *class translation rules* in this paper.

This section briefly describes the static class boundary determination (SCBD) method for multi-class classification commonly used in many approaches, then details the two new class translation rules: centred dynamic class boundary determination (CDCBD) and slotted dynamic class boundary determination (SDCBD).

3.1 Static Class Boundary Determination

Introduced in [6, 7], the static class boundary determination (SCBD) method has been used in many approaches to classification problems with three or more classes. In this method, two or more pre-defined thresholds/boundaries are applied to the numeric output value of the genetic program and the ranges/regions between these boundaries are linearly translated into different classes. This method is simple because these regions are set by the fixed boundaries at the beginning of evolution and remain constant during evolution.

If there are n classes in a classification task, these classes are sequentially assigned n regions along the numeric output value space from some negative numbers to positive numbers by $n-1$ thresholds/boundaries. Class 1 is allocated to the region with all numbers less than the first boundary, class 2 is allocated to all numbers between the first and the second boundaries and class n to the region with all numbers greater than the last boundary $n-1$, as shown in equation 2.

$$\text{Class} = \begin{cases} \text{class 1}, & v \leq T_1 \\ \text{class 2}, & T_1 < v \leq T_2 \\ \text{class 3}, & T_2 < v \leq T_3 \\ \cdots & \cdots \\ \text{class i}, & T_{i-1} < v \leq T_i \\ \cdots & \cdots \\ \text{class n}, & v > T_{n-1} \end{cases} \tag{2}$$

In this equation, n refers to the number of object classes, v is the output value of the evolved program, and T_1, T_2, T_{n-1} are static, pre-defined class boundaries.

3.2 Centred Dynamic Class Boundary Determination

The first new method is the Centred Dynamic Class Boundary Determination (CDCBD), where the class boundaries are dynamically determined by calculating the centre of the program output value for each class. The algorithm is presented as follow.

Step 1 Initialise the class boundaries as certain pre-defined values as in the SCBD method.

Step 2 Evaluate each genetic program in the population to obtain the program output value for each training example based on the SCBD method.

During the evolutionary process, repeat step 3 and step 4:

Step 3 For each class c, calculate the centre of the class according to equation 3:

$$\text{Center}_c = \frac{\sum_{p=1}^{M} \sum_{\mu_c=1}^{L} \text{ProgOut}_{p\mu_c}}{M \times L} \tag{3}$$

where M is the number of programs in the population and p is the index, L is the number of training examples for class c and μ_c is the index, $\text{ProgOut}_{p\mu_c}$ is the output value of the pth program on training example μ_c for class c.

Step 4 Calculate the boundary between every two classes by taking the middle point of the two adjacent class centres.

Step 5 Perform classification based on the new boundaries and equation 2.

While this method could be applied to every generation of the evolutionary process, we applied this method to the training examples every five generations to keep balance between evolution and class boundary determination.

3.3 Slotted Dynamic Class Boundary Determination

The second new class translation rule is Slotted Dynamic Class Boundary Determination (SDCBD). In this method, the output value of a program is split into certain slots. When a large number of slots are used, a large amount of computation would be required. In our experiment, we used 100 slots derived from the range of [-25, 25] with a step of 0.50. Since the input features (terminals) are scaled into [-1, 1], the range [-25, 25] is usually sufficient to represent the program output. Each slot will be assigned to a value for each class.

In the first step, this method evaluates each genetic program in the population to obtain the program output value (`ProgOut`) for each training example based on the SCBD method.

In the second step, the method calculates the slot values for each class (`Array[slot][class]`) based on the program output value. The algorithm for this step is as follows:

```
FOR each slot and each class
   Array[slot][class] = 0
FOR each training example X {
   FOR each program p {
      ProgOut = execute program p with X as input
      Round ProgOut to nearest slot
      IF ProgOut > 25 THEN ProgOut = 25
      IF ProgOut < -25 THEN ProgOut = -25
      Array[slot][class] += 1
   }
}
```

In the third step, this method dynamically determines to which class each slot belongs by simply taking the class with the largest value at the slot. However, in case a slot does not hold any positive value, that is, no programs produce any output at that slot for any training examples, then this slot will be assigned to the class of the nearest neighbouring slot, as shown in the following algorithm:

```
FOR slot = 1 to 100 {
   FOR all class c {
      IF  values of all class c in Array[slot][c] are zero {
         Class[slot] = ‘‘?’’ }
      ELSE {
         Search c for which  Array[slot][c] is largest
         Class[slot] = c }
   }
}

FOR slot = 1 to 100 {
   IF Class[slot] = ‘‘?’’ {
      Class[slot] = nearest value to slot in the Class vector
         whose value is not ‘‘?’’ }
}
```

Similarly to the CDCBD method, this method is also applied to the evolutionary process every five generations so that at other generations the classification performance will be only improved based on the evolutionary process.

3.4 Characteristics of the Dynamic Methods

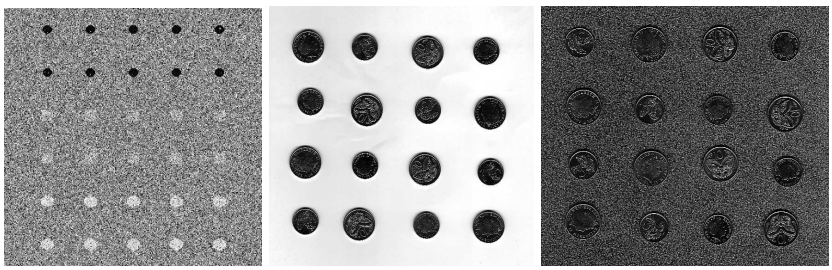
Compared with the SCBD method, these two new methods have the following characteristics:

- The optimal boundaries for every two different classes or the optimal slot values for each class can be dynamically determined during the evolutionary process.
- Class labels do not have to fit into the predefined sequential regions. The optimal regions for each class in the program output space can be automatically determined in the evolutionary process. For example, class 3 can be set in between class 1 and class 2 if necessary.

With these new properties, we expect that these two methods would perform better on multi-class object classification problems, particularly for relatively difficult problems.

4 Image Data Sets

We used four image databases in two groups with object classification problems of increasing difficulty in the experiments. Example images are shown in figure 1.



(a) (b) (c)

Fig. 1. Example images from Shape (a), Coin1 (b) and Coin2 (c)

4.1 Computer Generated Shape Datasets

The first group of images (figure 1 (a)) was generated to give well defined objects against a noisy background. The pixels of the objects were produced using a Gaussian generator with different means and variances for each class. Four classes of 713 small objects were cut out from those images to form the classification

data. The four classes are: black circles, light grey squares, white circles, and the grey noisy background.

Two different data sets, *shape1* and *shape2* were constructed from this group of images. While set shape1 arranges the four classes in an ordinary order based on the intensities, set shape2 out of this order. Table 2 gives the class order and the initial setting of the boundaries between these classes for the SCBD, CDCBD, and SDCBD methods. In the Shape1 data set, for example, classes 1, 2, 3, and 4 are arranged based on the ascending order of the intensities of the four classes. The class boundaries were set to -1.0, 0, and 1.0, meaning that if a program output value is less than -1.0 for a particular object example, then this object example will be classified as class1 (black circles); if the program output value is in (-1.0, 0], it will be classified as class2 (background). Clearly, the classification problem is linearly separable in Shape1, but non-linear separable in Shape2. The goal here is to investigate whether these classification methods perform well for the same data with the two different orders of class setting.

Table 2. The class orders in the two shape data sets.

Dataset	class label	description	intensity	boundary
Shape1	1	black circle	10±5	-1.0
	2	background	140±50	0
	3	grey square	180±50	1.0
	4	white circle	220±50	
Shape2	1	background	140±50	-1.0
	2	black circle	10±5	0
	3	white square	220±50	1.0
	4	grey square	180±50	

4.2 NZ Coin Datasets

The second group of images has two NZ coin data sets. The first data set (*coin1*, figure 1 (b)) consists of scanned 5 cent and 10 cent New Zealand coins. There are five classes of 576 object cutouts: 5 cent heads, 5 cent tails, 10 cent heads and 10 cent tails, and a relatively uniform background. Compared with the *shape* data set, this set of objects (heads versus tails for either 5 cent or 10 cent coins) are more difficult to distinguish and it has more classes.

The second data set (*coin2*, figure 1 (c)) also consists of five classes of object cutouts, but the background is highly clustered, which makes the classification problems much harder. Even human eyes cannot perfectly distinguish these objects.

The objects in each of the these data sets were equally split into three separate data sets: one third for the training set used directly for learning the genetic program classifiers, one third for the validation set for controlling overfitting, and one third for the test set for measuring the performance of the learned program classifiers.

5 Results and Discussion

This section presents a series of results of the two new dynamic class boundary determination methods on the four data sets in the shape and coin image groups. These results are compared with those for the static class boundary method. For all experiments, we run 10 times and the average results were presented.

5.1 Shape Data Sets

Table 3 shows the results of the three methods on the two shape data sets. The first line shows that for the Shape1 data set with 4 classes, the SCBD method achieved an average accuracy of 99.90% of 10 runs on the test set and the average number of generations of the 10 runs spent on the training process was 9.2.

Table 3. Results on the shape data sets.

Data set	Classes	Method	Gens	Accuracy
Shape1	4	SCBD	9.2	99.90%
		CDCBD	17.0	99.69%
		SDCBD	35.0	98.59%
Shape2	4	SCBD	50.0	96.87%
		CDCBD	26.5	99.67%
		SDCBD	43.5	98.46%

For the Shape1 data set, all the three classification methods obtained nearly ideal results, reflecting the fact that this classification problem is relatively easy. In particular, the SCBD method achieved the best performance.

For the Shape2 data set, the two new dynamic methods gave very good results. However, the static SCBD method produced a much worse performance in both classification accuracy and training time[1] than the two new dynamic methods because the classes in this data set were arranged arbitrary rather than in an ordinary order. This suggests that while the SCBD method could perform well on relatively easy, linearly separable classification problems with the classes arranged in an ordinary order, this method is not very appropriate for multi-class,non-linearly separable object classification problems with a randomly arranged order of classes. The two new dynamic methods, however, should be applied in this case. In addition, for these relatively easy classification problems, the CDCBD method seemed to be more effective and more efficient than the SDCBD method.

[1] Note that the dynamic boundary determination process takes a bit time time. However, since we apply the dynamic methods once every five generations and the computation cost of the two dynamic methods is quite low, the average time spent on each generation can be still considered similar.

5.2 Coin Data Sets

Table 4 shows the results of the three methods on the two coin data sets. For these more difficult datasets, both the two dynamic methods achieved better results than the static method. In particular, the SDCBD was clearly superior to the CDCBD method, suggesting that the SDCBD method is more effective than the CDCBD for these difficult classification problems.

Table 4. Results on the coin data sets.

Data set	Classes	Method	Gens	Accuracy
Coin1	5	SCBD	50	82.94%
		CDCBD	48.4	85.46%
		SDCBD	48.5	89.44%
Coin2	5	SCBD	50	72.78%
		CDCBD	50	76.48%
		SDCBD	49.3	84.50%

5.3 Summary and Discussion

In summary, the results suggest that the SCBD method could perform well on relatively easy object classification problems if the classes were arranged in their ordinary order (such as Shape1), but would perform badly when the classes were out of this order (as in Shape2) or when the classification problems became more difficult (such as Coin1 and Coin2). This is mainly because a high degree of non-linearity is required to map the class regions on the program output to the object features in these situations.

The performances of all the three methods on the Coin1 and Coin2 data sets were worse than the two shape data sets, reflecting the fact that the classification problems in these two data sets are more difficult than in the two shape data sets. Because these problems were harder, more features might need to be selected, extracted and added to the terminal set. Also more powerful functions might also need to be applied in order to obtain good performance. However, the investigation of these developments is beyond the goal and the scope of this paper. We leave this for the future work.

In terms of the two dynamic methods, the CDCBD method performed better for the relatively easy datasets, while the SDCBD method performed better for the relatively difficult datasets.

6 Conclusions

The goal of this paper was to investigate and explore dynamic class boundary determination methods as class translation rules in genetic programming for multi-class object classification problems, and to determine whether the new

dynamic methods could outperform the static method for relatively difficult problems. Two new classification methods, CDCBD and SDCBD, were developed and implemented where the class boundaries were dynamically determined during the evolutionary process.

The results on the four object classification problems in two groups of images showed that the static method, SCBD, performed very well on the relatively easy, linearly separable object classification problems where the classes were arranged in their ordinary order, but performed less well when the classes were arranged in an arbitrary order. While the two dynamic methods, CDCBD and SDCBD, also performed very well on the relatively easy object classification problems, they generally took longer training times. However, the two new dynamic methods performed much better than the static SCBD method for the relatively difficult, non-linearly separable object classification problems.

As expected, the classification performance on the four image datasets deteriorated as the degree of difficulty of the object classification problems increased.

These results suggest that, for relatively easy object classification problems with classes in a linear order, both the static method (SCBD) and the two dynamic methods (CDCBD and SDCBD) can be applied, but the static method SCBD is recommended if training time is a critical factor. For other situations, the two dynamic methods are recommended.

Although developed for image/object classification problems, these two dynamic methods are also expected to be applied to general classification problems.

For future work, we will investigate whether the performance on the relatively difficult coin data sets can be improved if more features are added to the terminal set. We will also investigate the power and reliability of the two new methods on even more difficult, real-world image classification problems such as face recognition problems and satellite image detection problems, and compare the performance with other long-term established methods such as decision trees and neural networks.

References

1. J. Eggermont, A. E. Eiben, and J. I. van Hemert. A comparison of genetic programming variants for data classification. In *Proceedings of the Third Symposium on Intelligent Data Analysis (IDA-99), LNCS 1642*. Dpringer-Verlag, 1999.
2. Helen Gray. Genetic programming for classification of medical data. In John R. Koza, editor, *Late Breaking Papers at the 1997 Genetic Programming Conference*, pages 291–297. Standford University, 1997.
3. John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, Mass. : MIT Press, London, England, 1994.
4. Thomas Loveard and Victor Ciesielski. Representing classification problems in genetic programming. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1070–1077, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.
5. Andy Song, Vic Ciesielski, and Hugh Williams. Texture classifiers generated by genetic programming. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry

Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 243–248. IEEE Press, 2002.

6. Walter Alden Tackett. Genetic programming for feature discovery and image discrimination. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 303–309, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann.
7. Mengjie Zhang and Victor Ciesielski. Genetic programming for multiple class object detection. In Norman Foo, editor, *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence (AI'99)*, pages 180–192, Sydney, Australia, December 1999. Springer-Verlag Berlin Heidelberg. Lecture Notes in Artificial Intelligence (LNAI Volume 1747).
8. Mengjie Zhang, Peter Andreae, and Mark Pritchard. Pixel statistics and false alarm area in genetic programming for object detection. In Stefano Cagnoni, editor, *Applications of Evolutionary Computing, Lecture Notes in Computer Science, LNCS Vol. 2611*, pages 455–466. Springer-Verlag, 2003.
9. Mengjie Zhang, Victor Ciesielski, and Peter Andreae. A domain independent window-approach to multiclass object detection using genetic programming. *EURASIP Journal on Signal Processing, Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis*, 2003(8):841–859, 2003.
10. Daniel Howard, Simon C. Roberts, and Richard Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311, 1999.
11. Jamie R. Sherrah, Robert E. Bogner, and Abdesselam Bouzerdoum. The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 304–312, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.
12. Walter Alden Tackett. *Recombination, Selection, and the Genetic Construction of Computer Programs*. PhD thesis, Faculty of the Graduate School, University of Southern California, Canoga Park, California, USA, April 1994.