Genetic Programming for Feature Discovery and Image Discrimination

Walter Alden Tackett
University of Southern California, Dept. of EE Systems
and Hughes Missile Systems Co.
Mailing Address: 8433 Fallbrook Ave, Canoga Park Ca, 91304
Mail Code CP / 262 / C-64
Phone: (818) 702-1611
e-mail: tackett@ipld01.hac.com

Abstract: We apply Genetic Programming (GP) to the development of a processing tree for the classification of features extracted from images: measurements from a set of input nodes are weighted and combined through linear and nonlinear operations to form an output response. No constraints are placed upon size, shape, or order of processing within the network. This network is used to classify feature vectors extracted from IR imagery into target/nontarget categories using a database of 2000 training samples. Performance is tested against a separate database of 7000 samples. This represents a significant scaling up from the problems to which GP has been applied to date. Two experiments are performed: in the first set, we input classical "statistical" image features and minimize misclassification of target and non-target samples. In the second set of experiments, GP is allowed to form it's own feature set from primitive intensity measurements. For purposes of comparison, the same training and test sets are used to train two other adaptive classifier systems, the binary tree classifier and the Backpropagation neural network. The GP network achieves higher performance with reduced computational requirements. The contributions of GP "schemata," or subtrees, to the performance of generated trees are examined.

1. Introduction

Genetic Programming is a relatively recent technology which has been demonstrated as a versatile tool for Automatic Program Generation in a variety of applications **[1]**. Many of these applications have been under conditions where a known optimal solution is determined in advance. This leaves open the question as to whether GP can "scale up" to real-world situations, where answers are not known, data is noisy, and measurements may be of poor quality. We attempt to address this by constructing such a problem: large volumes of noisy image data are segmented and processed into statistical features. These features are used to assign each image to a "target" or "nontarget" category. Because they are constrained by processing requirements, the segmentation and feature measurement are of a coarse and sometimes unreliable nature. Because of this it is likely that overlap between classes in feature space exists, and hence discovery of a 100% correct solution is unlikely. Two experiments are performed: in the first we insert a GP-generated tree into an existing system in order to test it against other well-established adaptive classifier technologies, specifically Backpropagation and binary tree methods. We then proceed to examine whether GP can be inserted at an earlier stage of processing, obviating the need for costly segmentation and feature extraction stages.

2. Genetic Programming

Koza **[1]** has successfully demonstrated the use of Genetic Programming for pattern recognition, control , planning, and the generation of neural networks. Unlike conventional Genetic Algorithms (GA), which use fixed-length binary strings, GP begins with a population of randomly generated LISP programs represented by their parse trees. The LISP programs and the trees which represent them are constructed from a *function set* and a *terminal set* chosen by the user. A function takes one or more arguments, which may be terminals or may themselves be functions. Functions form the root and the internal nodes of the parse tree. Terminals form the leaves of the parse tree and may represent input variables, constants, or sensor measurements. Fitness-proportionate selection is carried out in the same manner as that of conventional GA. In the GP crossover operation, randomly selected subtrees of each chosen parent pair are swapped to form a pair of (usually) unique, syntactically correct offspring. Mutation consists of randomly replacing subtrees in existing members of the population with new randomly generated subtrees.

3. The Problem

3.1. Target / Non-Target Discrimination

Target / nontarget discrimination is an important first stage in Automatic Target Recognition (ATR), and in general for systems which require attention to a small sub-area (or areas) within a much larger image. Whereas a highly sophisticated pattern recognizer may make fine discriminations between subtly different patterns **[2]**, it generally does so at a very high computational cost. A much more efficient way to process information is to employ a simpler "*detection*" algorithm to the entire image, identifying subareas of interest. These areas are only coarsely classified as "target" or "non-target," and with lower reliability than the recognizer. Only target areas are passed from the detector to the recognizer, reducing image bandwidth and throughput requirements. Thus there is a constraint on this problem that GP must produce a solution which is computationally efficient in order to be useful. There is further advantage to be gained if GP is able to bypass costly processing associated with feature extraction.

3.2. Image Database

Data was taken from US Army NVEOD Terrain Board imagery, specifically the MTAP Phase I and Phase II test database. These are 512x640 pixel images which simulate IR views of vehicles, including tracked and wheeled vehicles, fixed- and rotary- wing aircraft, air defense units, and a wide variety of cluttered terrain. The range, field of view, and sensor resolution are such that individual targets occupy between 100-300 pixels. There are a total of about 1500 images containing multiple target and clutter objects, providing a total of about 13,000 samples to be classified. Training and test data for the experiments described here were drawn randomly from these samples.

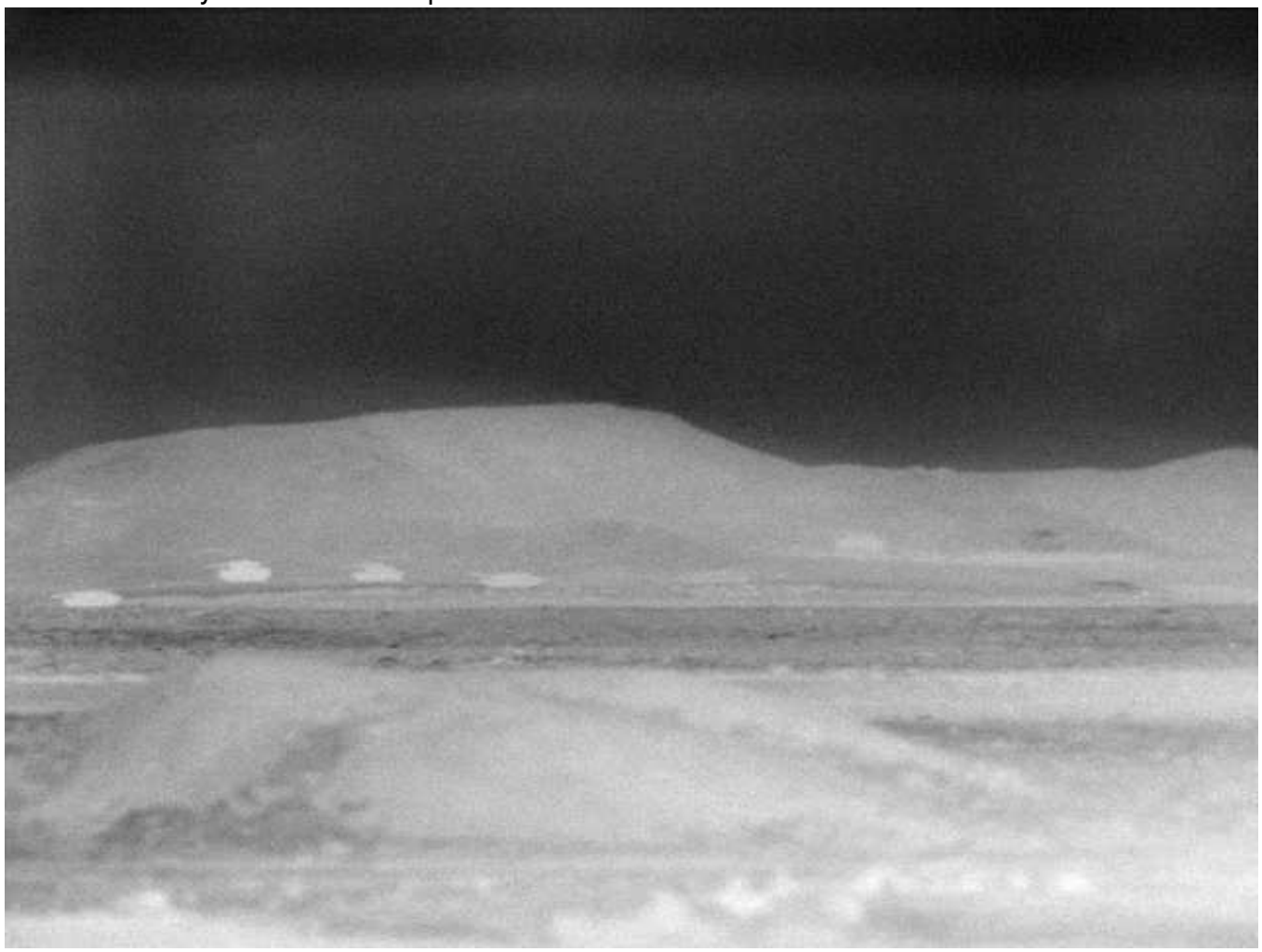

Figure 1:Eight (!) Tank targets, light clutter (from Army NVEOD Terrain Table)

3.3. Feature Extraction

In the experiments described here we have used Genetic Programming to construct classifiers which process the feature vectors produced by an existing algorithm - specifically the Hughes MTAP ATR system. This system performs two sequential steps of image processing.

3.3.1. AntiMean Detection Filter

This system uses an antimean filtering to extract "blobs" in a range of sizes conforming to those of expected targets. The image is divided into a mosaic of 62x62 pixel overlapping regions, or "large windows." These regions are themselves divided into overlapping 5x5 "small

windows." When a blob of appropriate size is detected, it is described in terms of seven primitive features: contrast with the local background, global image intensity mean and standard deviation, and the means and standard deviations of the "large window" and "small window" in which it is centered. Later we will apply GP to the classification of these features. In the conventional MTAP algorithm, however, they are passed to a second processing stage.

F00 Size Filter Value ("Blob Contrast")
F01 Global Image Intensity Mean
F02 Global Image Intensity Standard Deviation
F03 "Large Window" Intensity Mean
F04 "Large Window" Intensity Standard Deviation
F05 "Small Window" Intensity Mean
F06 "Small Window" Intensity Standard Deviation

Table 1: Seven primitive features are based on a hierarchy of image window intensities.

3.3.2. Segmentation and Feature Extraction

In the conventional MTAP system, the seven features from the antimean detection filter are passed through a simple linear classifier in order to determine whether segmentation and feature extraction should be performed upon the blob. If so, the large image window undergoes a 3:1 decimation filtering to reduce bandwidth and statistical measures of local intensity and contrast are used to perform a binary figure / ground segmentation, resulting in a closed-boundary silhouette. Twenty moment- and intensity-based features are extracted from this segmented region. They are summarized in table 2. Because this segmentation scheme depends on fixed thresholds under varying image conditions, silhouettes for a given target type can vary significantly. Likewise, since features do not encode detailed shape properties it is possible for non-target objects such as oblong rocks to be encoded into target-like regions of feature space. Thus these feature vectors may display significant variance as well as overlap between target and non-target classes.

F00 radius of gyration	**F08** perimeter^2/area
F01 rectangularity	**F09** normalized average segmentation greylevel
F02 height^2/width^2	**F10** area
F03 width^2/height^2	**F11** height^2/area
F04 normalized delta greylevel	**F12** height^2/range^2
F05 symmetry	**F13-F17** higher order moments
F06 range to target	**F18** area*range^2
F07 depression angle	**F19** polarity of delta greylevel

Table 2: Twenty Statistical Measurements on Segmented Image

4. Approach

The fitness of an individual tree is computed by using it to classify about 2000 samples. At each generation, the individual which performs best against the training set is additionally run against a 7000 sample "validation test set," and the results reported.

4.1. Function Set

Both experiments share a common function set: **F** = {**+**, **-**, *****, **%**, **IFLTE**} represents four arithmetic operations which form 2nd order nodes (i.e., 2 branches, or arguments) and a conditional operation which forms 4th order nodes (4 branches / arguments). The +, -, and *

operators represent common addition, subtraction, and multiplication, while % indicates "protected" division: division by zero yields a zero result without error. The conditional IFLTE returns the value of the third argument if the first argument is less than the second, otherwise the fourth argument is returned.

4.2. Terminal Set and Fitness Function for Experiment 1

The terminal set **T** = {**F00**...**F19**, **RANFLOAT**} for the first experiment consists of the 20 segmented "statistical" features shown in Table 2 as well a real random variable RANFLOAT, which is resampled to produce a constant value each time it is selected as a terminal node. The resulting tree takes a bag of floating-point feature values and constants as input, and combines them through linear and nonlinear operations to produce a numeric result at the root of the tree. If this result is greater than or equal to 0, the sample is classified as a target, otherwise it is classified as clutter (non-target). Testing of trees against the 2000-sample set results in two measurements: the first is probability of incorrect classification, or the total fraction of samples assigned to the incorrect category. It had previously been observed that performance could be enhanced by including a larger number of target samples than clutter samples in the training database (see Section 6: Results). This gives rise to the problem that a classifier which says *everything* is a target may get a relatively high fitness score while conveying no information in the Shannon sense. To counteract this, a second measure of fitness was added: the a posteriori entropy of class distributions **H**(**class**|**output**) after observing classifier output. These multiple objectives are minimized via mapping to the Pareto plane and subjecting them to a nondominated sort **[3]**. The resulting ranking forms the raw fitness measure for the individual.

4.3. Terminal Set and Fitness Function for Experiment 2

In the second experiment, only the seven primitive intensity features from the antimean discriminant filter are used. The terminal set **T** = {**F00**...**F06**, **RANINT**} consists of these seven integer features and an integer random variable which is sampled to produce constant terminal nodes.

The fitness function is reformulated from experiment 1 for "historical reasons," namely the constraints that were placed on the original MTAP system. These state that the detection filter should be able to find five (5) targets in an image with 80% probability of detecting all of them. This means that individual Probability of Detection (**p(D)**) must be $0.8^{0.2}$, or about 96%. With this figure fixed, we seek to minimize the Probability of False Alarms (**p(FA)**), the chance that non-targets are classed as targets. This is done in the following manner: a tree is tested against each sample in the data base, and the values it produces are stored into two separate arrays, one for targets and the other for clutter. These arrays are then sorted and the threshold value is found for which exactly 96% of the target samples produced a greater output. We compare this value with those in the clutter array in order to determine what percentage fall above this threshold value. This percentage is the False Alarm Rate which we seek to minimize.

5. Backpropagation and Binary Tree Classifier

As an experimental control the same test and training database are used to build and evaluate two other classifiers. The first is a Backpropagation network with adaptive stepsize η **[4]** and 2 output nodes (one for each class). For both experiments, the "desired output" activation for the back propagation was {1.0, 0.0} when presented with a target sample, and {0.0, 1.0} when presented with a clutter sample. Differences between these desired outputs and those actually obtained form error terms to be fed back. Weight updates take place after each sample presentation. Training inputs to the Backprop network are normalized so that values do not exceed an absolute value of 1.0. A threshold difference between network outputs is used to determine if a sample is target or clutter. This threshold is 0.0 for

experiment 1 and variable for experiment 2. The second classifier tested is a binary tree classifier **[5]** which partitions feature space into hyperrectangular regions, choosing features and their thresholds at decision nodes based upon the maximization of Information Rate (also known as mutual information) **[6]**. This particular binary tree classifier was originally developed and optimized for use with the feature set of experiment 1. It was not used in experiment 2.

6. Results

For all experiments performance was plotted as a function of Probability of False Alarm **p(FA)** vs. Probability of Detection **p(D)**. Probability of False Alarm is the fraction of non-targets classified as targets divided by the total number of non-target samples. Probability of Detection is the fraction of targets correctly classified divided by the total number of target samples. Ideally a system should achieve a p(FA) of 0.0 and a p(D) of 1.0.

6.1. Experiment 1

Four runs were performed using GP with different random seeds. Each used a population of 500 and run length of 60 generations, resulting in the analysis of a total of 120,000 individuals. Three different clutter-to-target ratios (C/T ratio) were used in the training database: 0.5:1, 0.71:1, and 1:1, with 0.5:1 consistently producing the best results[1]. Backpropagation was similarly tested using three C/T ratios, again with C/T ratio of 0.5:1 producing best results. The network used 20 input nodes (the 20 features) and 10 hidden nodes (empirically determined to be the best number). Four random initial weight sets were used for each C/T ratio, resulting in a total of 12 learned weight sets. After each training epoch each network was tested against the separate validation test set. The best result achieved against this test set was reported as the figure-of-merit.

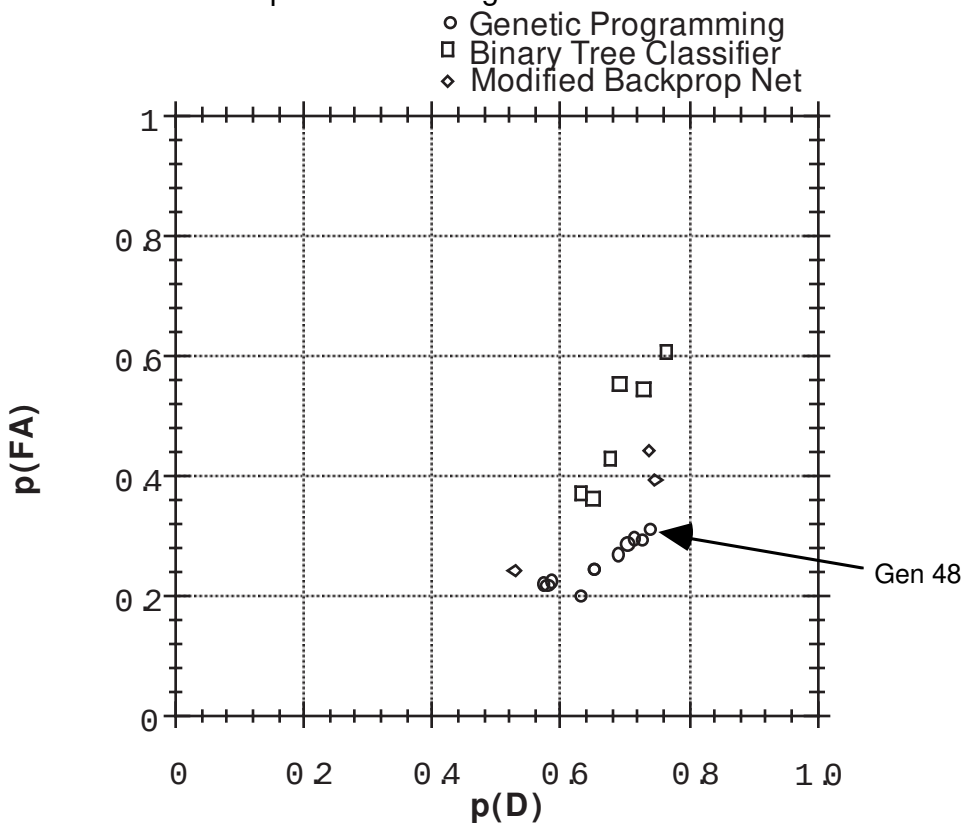

Figure 2: Probability of False Alarm vs. Detection

[1]C/T ratios less than 0.5:1 were found to produce particularly lousy results.

The binary tree classifier was tested using the three C/T ratios, and additionally tested with an input set that eliminates gray-level features. There is no random initial element. Various system parameters of the tree builder were refined during its development cycle.

Figure 2 summarizes the performance of the three classifiers. The three points corresponding to Backpropagation depict the best performance achieved against the 3 C/T ratios. The six points shown for the binary tree correspond to three C/T ratios for the two input feature sets described above (the cluster of three points with higher p(FA) were obtained with gray level features omitted). The 10 points shown for GP are the best-of-generation individuals for selected generations in a single evolution run with C/T ratio of 0.5. All three methods were able to achieve p(D) of 74%-75%. Genetic programming, however, achieves a 31% false alarm rate, some 8% lower than the 39% p(FA) achieved by Backprop for the same p(D), and 30% lower than the 61% figure achieved by the Binary Tree.

```
(+ (* F07
      (IFLTE (* F07 F04)
         (-
          (-
            (*
               (IFLTE (* F13 F06) (- F16 F03)
                     (IFLTE F17 F19 F14 F07)
                     (+ F09 F09))
               (+ F06 F01))
            (% F02 F00))
           F11)
          (- F06 F04)
          (+ -3.3356472329868225 F04)))
  (+ (- (+ (IFLTE (* F13 F06)
                (+ -3.3356472329868225 F04)
                (+ (+ -3.3356472329868225 F04)
                   (% F02 F00))
                (+ F09 F09))
           F04)
         F11)
    (- (- (IFLTE
          (IFLTE F16 F16 F19 (- F04 F11))
          (* (* F13 F06) (+ F06 F01))
          (IFLTE F16 F15 F04 F01)
          (* F07
             (IFLTE (* F07 F04)
                (+ -3.3356472329868225 F04)
                (% F11 F11) (+ F04 F02))))
       (+
        (*
         (-
          (IFLTE F16 F16 F19 (+ F09 F09))
          (* F07 F04))
         (+ F09 F09))
        (-
         (* (- F01 F07)
            (IFLTE F14 F08 F06 F08))
         (+ (- F16 F03)
            (*
             (IFLTE
              (+ -3.3356472329868225 F04)
              (IFLTE F14 -3.3356472329868225
                     F12 F03)
              (IFLTE (* F13 F06)
                     (IFLTE F16 F16 F19
                            (- F04 F11))
                     F02 (- F06 F04))
              (+ F04 F02))
             (IFLTE (* F13 F06) (- F16 F03)
                    (+ -3.3356472329868225 F04)
                    (+ F09 F09)))))))
    (* F10 F08))))
```

Figure 3: Best-of Generation Tree for Generation 48

Figure 3 shows the best-of-run individual for generation 48 (indicated by the arrow in Figure 2), with the dendritic tree represented in LISP program format. This program achieved 74.2% correct with a 30.8% false alarm rate. Counting function nodes we see that there are 55 mathematical operations required and 15 logical comparisons / branches. By comparison, the Backpropagation network requires 440 math operations and 12 nonlinearities: about 8x more computation is required in order to achieve 8% *lower* performance.

6.2. Experiment 2

As with experiment 1, Genetic Programming was run four times for 60 generations with a population of 500 and three C/T ratios with the fitness function described in Section 4.3. Training and testing for the Backpropagation network were performed exactly as described in sections 5 and 6.1, except that this network contained seven input nodes and four hidden nodes. For both GP trees and Neural Networks a C/T ratio of 0.5:1 resulted in best performance.

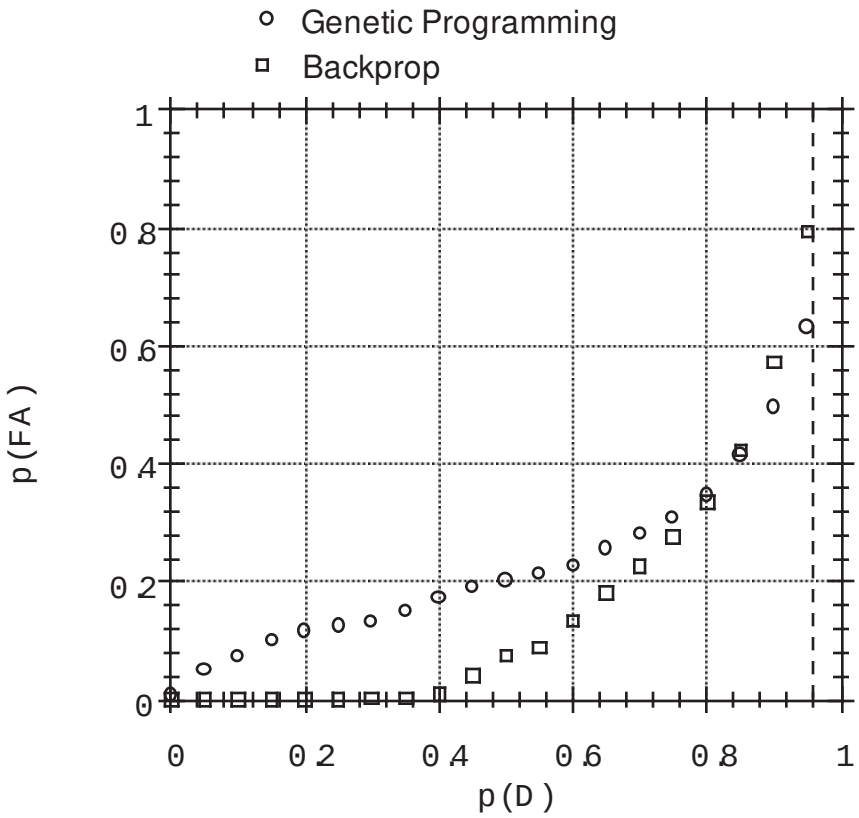

Figure 4: p(FA) as a function of p(D) for primitive feature set. Genetic programming fitness was based on minimization of p(FA) with p(D) fixed at 96% (dashed line).

Figure 4 depicts the performance of the Backpropagation and GP systems. The dashed vertical line indicates the desired p(D) of 96%. For this value of p(D) the GP tree produced a 65.8% false alarm rate, while Backpropagation produced 82.6%. By varying the p(D) threshold, we generate curves showing p(FA) for these other values. This provides an insight about the two algorithms: GP used a fitness function which was specifically tuned to the desired p(FA) performance, whereas the Backprop training was independent of the threshold (used only *after* training) with no straightforward way to incorporate the 96% constraint. Thus GP chooses a function which specifically trades performance at high p(D) values against that at lower values.

```
(% (- F02 F00)
   (+ (+ (% (- (* F02 F03) F06)
            (+ (- F02 F00) (- F03 F00)))
         (- F03 F00))
      (- F03 F00)))
```

Figure 5: Genetic Programming best-of-generation for generation 5, using primitive feature set. This, the most successful discriminant used neither the full terminal set nor IFLTE from the function set.

The individual whose performance is depicted in Figure 4 is shown in LISP expression form in Figure 5. This function requires 12 mathematical operations, compared to the 72 math operations and 6 nonlinear function evaluations required by the Backpropagation network. In

and of itself it displays some remarkable properties: it does not make use of the IFLTE conditional branch, and ignores the input features F01, F04, and F05. Likewise, the system makes repeated use of synthesized "metafeatures," (- F02 F00) and (- F03 F00). Clearly this gives insight into which features are and are not useful for the low level discrimination task. Moreover, the atypically small size of this expression[2] allows us to make a unique analysis: Figure 6 shows performance obtained by decomposing this expression into it's three unique subtree components, which we postulate comprise "GP schemata**[7]**."

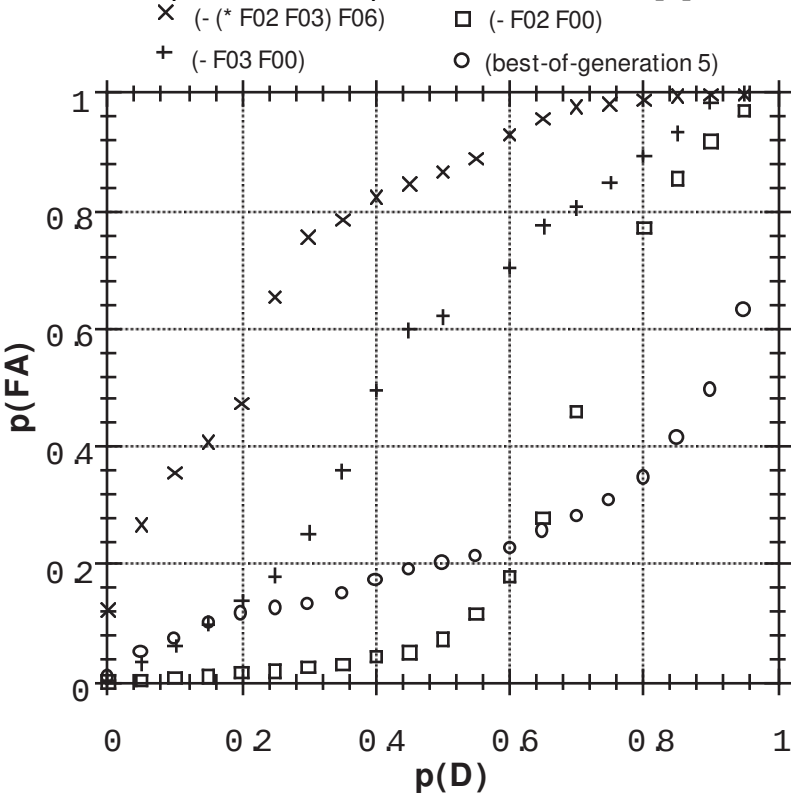

Figure 6: Performance of individual schemata (subexpressions) and their parent expression.

The schema (- F02 F00) appears, at p(D) values below 60%, to display performance in the same ballpark as that of the Backpropagation network. Significantly, all three schemata approach 100% p(FA) at 96% p(D). Thus they are in no way contributing to the fitness of the parent expression by a principle of superposition, but rather by their nonlinear interactions. Although GP may have a different notion of schemata and a nonbinary alphabet, we suggest that this observation underscores principles previously set forth by Goldberg **[8]**.

7. Discussion

7.1. Primitive Features and Improved Performance

One significant observation is that the performance obtained in Experiment 2 is actually better than that of Experiment 1: for a p(D) of 75%, the GP tree of Experiment 2 obtained 30% p(FA) while the Backprop net obtained 27%. This is unlikely to be due to coincidence since both methods showed improved performance. Why else, then? One hypothesis is that both of these powerful nonlinear adaptive methods may be discovering inherent features which are better suited to the data than human-synthesized features based upon measurements

[2]Other trees of comparable (but slightly lower) performance were usually 2-4 times as large.

(incorrectly) presupposed to be invariant. We may also speculate that the segmentation process introduces artifacts and variations which do not exist in the raw data. Other factors must be considered: we have mentioned that the image resolution is reduced prior to feature extraction, compressing groups of 3x3 (i.e., nine) pixels into one. This reduction in bandwidth may be harmful. Finally, the reduction in the number of input features itself may exponentially reduce the complexity of problem space to be searched, making relatively good solutions easier to find. Regardless of the cause we may conclude that this appears to be a result of direct benefit to the development of future systems. It also suggests that applying GP to raw pixel imagery is a promising area for follow-on research.

7.2. GP Offers Problem Insights

We have seen that in the case of experiment 2, the structure of the solution tree offers clues as to which features assist in pattern discrimination. Although it is more complex, the same analysis can be applied to Experiment 1. This indicates that GP can provide insights into what aspects of the problem are important: we can understand the problem better by examining the solutions that GP discovers. Thus the symbolic nature of Genetic Programming offers some advantage over more opaque methods of neural learning.

7.3. A GP Schema Theory?

Upon examination, we see that there are many repeated structures in the tree of experiment 1 as well as that of experiment 2. By probability, these are *not* identical structures randomly generated at the outset, but rather successful subtrees, or *schema*, which were frequently duplicated and spread throughout the population at an exponential rate due to their contribution to the fitness of individuals containing them. These suggest that a schema theory may be developed for Genetic Programming analogous to that for binary GA originally proposed by Holland **[7].** Future work will focus on identification and analysis of the specific contributions of high-value schemata to the classification process.

7.4. Parsimony

Parsimony, or simplicity of solution structures, has previously been shown by Koza **[1]** to be achievable by adding the tree size (expression length) as a component of the fitness function to be minimized. In both experiments it was observed that a high degree of correlation exists between tree size and performance: among the set of "pretty good" solution trees, those with highest performance were usually the smallest within that set. It was also observed that most runs achieved a point where the size and complexity of trees eventually began to grow increasingly larger, while performance tapered off to lower values. We suggest that for an open-ended exploration of problem space, parsimony may be an important factor not for "aesthetic" reasons or ease of analysis, but because of a more direct relationship to fitness: there is a bound on the "appropriate size" of solution tree for a given problem.

8. Acknowledgments

The author wishes to thank Dr. Kenneth Friedenthal and Dr. Knut Kongelbeck of Hughes Missile Systems Company for supporting this work under the 1992 Innovative Initiative and Internal Research & Development funds.

9. Bibliography

[1] Koza, John R. *Genetic Programming* . Cambridge, MA: MIT Press, 1992.

[2] Daniell, CE, Kemsley, DH, Lincoln, WP, Tackett, WA, Baraghimian, GA "Artificial Neural Networks for Automatic Target Recognition," *Optical Engineering*, v31 n12, Dec. 1992, pp. 2521-2531.

[3] Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading MA: Addison Wesley, 1989.

[4] Hertz, J., Krogh, A., Palmer, RG *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison Wesley, 1991.
[5] Kanal, LN "Problem-solving models and search strategies for pattern recognition," IEEE Trans. Pattern Anal. Machine Intell., vol. PAMI-1, pp. 194-201, Apr. 1979.
[6] -, *Bandwidth Reduction Intelligent Target-Tracking / Multi-function Target Acquisition Processor (BRITT / MTAP): Final Technical Report, CDRL B0001.* El Segundo, CA: Hughes Aircraft Co., May 1990.
[7] Holland, John H. *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1992.
[8] Goldberg, David E., "Zen and the Art of Genetic Algorithms," in *Proc. of the Third International Conference on Genetic Algorithm*s. San Mateo, CA: Morgan Kaufman, 1989.