MDL-based Genetic Programming for Object Detection

Yingqiang Lin and Bir Bhanu

Center for Research in Intelligent Systems
University of California, Riverside, CA, 92521, USA
Email: {yqlin, bhanu}@vislab.ucr.edu

Abstract

In this paper, genetic programming (GP) is applied to synthesize composite operators from primitive operators and primitive features for object detection. To improve the efficiency of GP, smart crossover, smart mutation and a public library are proposed to identify and keep the effective components of composite operators. To prevent code bloat and avoid severe restriction on the GP search, a MDL-based fitness function is designed to incorporate the size of composite operator into the fitness evaluation process. The experiments with real synthetic aperture radar (SAR) images show that compared to normal GP, GP algorithm proposed here finds effective composite operators more quickly.

1. Introduction

Object detection is one of the important steps in developing computer vision and pattern recognition systems. The major task of object detection is to locate objects in images and extract the regions-of-interest (ROIs). The quality of object detection is highly dependent on the features used in detection. There are many features and the ways of synthesizing composite features by combining primitive operations and primitive features are almost infinite. The human experts, relying on their experience, knowledge and limited by their speed and bias, consider only a small number of conventional combinations and ignore many combinations they regard as nonsense. Genetic programming (GP), on the other hand, may try many unconventional combinations that may yield exceptionally good results. However, GP is very computationally expensive. In the traditional GP (also called *normal GP*), crossover and mutation locations are randomly selected, leading to disrupting the effective components (subtree in this paper) of composite operators and greatly reducing the efficiency of GP. To improve the efficiency, it is very important for GP to identify and keep effective components.

In this paper, we use genetic programming (GP) to synthesize composite features, which are the output of composite operators, for object detection. A composite operator is represented by a binary tree whose internal nodes are the pre-specified primitive operators and the leaf nodes are primitive feature images. It can be viewed as a way of combining primitive operations on images for ROI extraction [1]. To improve the efficiency of GP, smart crossover, smart mutation and a public library are proposed in this paper to identify and keep the effective components of composite operators for later reuse. We also design a fitness function based on the minimum description length (MDL) principle [2] to take the size of composite operators into the evaluation process to address the well-known code bloat problem of GP while at the same time avoiding the severe restriction on the GP search.

2. Motivation and related research

• **Motivation:** Crossover and mutation are two major mechanisms employed by GP to search the composite operator space. Since the initial population is randomly generated, it is unlikely to contain large good components. The probability of crossover and mutation breaking up a good component is small and the fitness of composite operators is increased. As search proceeds, small good components are generated and assembled into larger and larger good components and composite operators become more and more fragile, since the large good components are easily broken up by subsequent crossover and mutation due to the random selection of crossover and mutation points. The crossover can damage the fitness of a composite operator by moving good components into inhospitable contexts in which their effectiveness is canceled by other nodes. It is highly desired that good components can be identified from disruption and saved in a public library for later reuse.

GP has a well-known code bloat problem in which the sizes of individuals become larger and larger. It takes a long time to execute a large composite operator and large composite operators may overfit the training data by approximating the noises in images. Usually in normal GP, a size limit on composite operator is set up to prevent code bloat. However, the hard size limit restricts the GP search and makes it unlikely to find effective composite operators, since after randomly selecting a crossover point in one composite operator, GP cannot select some nodes of the other composite operator as crossover points in order to guarantee that both offspring are within size limit. To overcome this problem, we design a MDL-based fitness function to take the composite operator size into evaluation process. According to MDL principle, large composite operators effective on training regions may not have high fitness, so we can take off the hard size limit without making composite operators grow too large. Large composite operators don't have high fitness and will be culled out by selection.

- **Related Research:** Genetic programming has been used in image processing, object detection and recognition. To improve the efficiency of GP, Tackett [3] devises brood recombination to reduce the destructive effect of crossover; D'haeseleer [4] devises strong context preserving crossover (SCPC) to preserve the context; Smith [5] proposes a conjugation operator for GP to transfer genetic information from good individuals to bad ones. Ito et al. [6] propose a depth-dependent crossover for GP in which nodes closer to the root node have better chances of being selected as a crossover point to lower the chance of disrupting small good components near leaves. Unlike the work of Ito [6] that uses only the syntax of the tree (the depth of a node), the smart crossover and smart mutation proposed in this paper evaluate the performance at each node to determine the interactions among them and use these semantic information to choose crossover and mutation points.

Unlike standard genetic algorithms (GAs) working on fixed length strings, the individuals in genetic programming are complicated structures such as trees and graphs of dynamically varying sizes. Messy GAs manipulate on variable length chromosomes whose genes are represented by the pair (allele locus, allele values) and the new representation of chromosomes makes them more resistant to the destructive force of crossover, decreasing the chances of disrupting effective building blocks [7]. Unlike messy GAs, the individuals in this paper are composite operators represented by binary trees, not variable length chromosomes consisting of string of genes. Although the tree structures preclude the use of most traditional approaches, such as dynamic programming and reinforcement learning, to the learning of effective composite operators, they can represent more complex features than chromosomes.

3. Technical approach

- **MDL-based fitness function:** We design a MDL-based fitness function to incorporate the composite operator size into the fitness evaluation process. The fitness of a composite operator is defined as the sum of the description length of the composite operator and the description length of the segmented training regions with respect to this composite operator as a predictor for the label (object or background) of each pixel. Here, both lengths are measured in bits and the details of the coding techniques are relevant.

$$F(CO_i) = - (r\times\log (N_{po}) \times Size(CO_i) + (1 - r) \times (n_o + n_b) \times(\log(W_{im})+\log(H_{im}))) \quad (1)$$

where CO_i is the *i*th composite operator in the population, N_{po} is the combined number of primitive operators and primitive feature images available for GP to synthesize composite operators, $Size(CO_i)$ is the size of the composite operator, which is the number of nodes in the binary tree representing it, n_o and n_b are the number of object and background pixels misclassified, W_{im} and H_{im} are the width and height of the training image and r determines the relative importance of the above two terms, which is 0.7 for all the examples in this paper.

The trade-off between the simplicity and complexity of a composite operator is that if the size of the composite operator is too small, it may not capture the characteristics of the objects to be detected, on the other hand, if the size is too large, the composite operator may overfit the training image, thus performing poorly on the unseen testing images. With the MDL-based fitness function, the composite operator with the minimum combined description lengths of both the operator itself and image-to-operator error is the best composite operator and may perform best on the unseen testing images. In our previous work [1], the fitness function is defined as $n(G\cap G') / n(G \cup G')$, where G and G' are foreground in the ground truth image and the resultant image of the composite operator respectively and $n(X)$ denote the number of pixels within the intersection of region X and the training regions. It measures how the ground truth and detection results overlap. In this paper, this measure is called the *goodness* of a composite operator.

- **Primitive feature images and operators:** There are 16 primitive feature images: the first one is the original image (0); the others are mean (1–3), deviation (4–6), maximum (7–9), minimum (10–12) and median (13–15) images obtained by applying templates of sizes 3×3, 5×5 and 7×7. These images are the input to composite operators. GP determines which operations are applied on them and how to combine the results. A primitive operator performs a primitive operation on one or two input images and stores the resultant image. Currently, 17 primitive operators such as ADD, SUB, MUL, DIV, MAX2, MIN2, ADDC, SUBC, MULC, DIVC, SQRT, LOG, MAX, MIN, MED, MEAN and STDV, are used by GP to synthesize composite operators [1]. It is worth noting that the primitive feature images and primitive operators are domain-independent, so our method can be applied to a wide variety of images.

- **Parameters and termination:** The key parameters are the population size M, the number of generation N, the crossover rate, the mutation rate and the goodness threshold. The GP stops whenever it finishes the pre-specified number of generations or whenever the best composite operator in the population has goodness value greater than the goodness threshold.

- **Selection, crossover and mutation:** The selection operation selects composite operators from the current population to let them survive into next generation. In this paper, we use tournament selection.

To perform *crossover*, two composite operators (called parents) are selected on the basis of their fitness values. The higher the fitness value, the more likely the composite operator will be selected for crossover. In normal GP, due to the random selection of crossover point, crossover becomes destructive at the later stage of GP search when composite operators contain large effective components. To avoid the above problem, we propose smart crossover to identify and keep the effective components. In smart GP, the output image of each node is evaluated and its fitness value is recorded. We define the fitness of an edge as the fitness difference between the parent node and the child node linked by the edge. An *Edge* is classified as good edge if its fitness is positive. Otherwise, it is a bad edge. During crossover, all the bad edges are identified and one of them is selected by random selection or roulette selection (based on the fitness of the bad edges) invoked with equal probability. The child node of the selected bad edge is the crossover point and the subtree rooted at the crossover points are swapped between parents. If a composite operator has no bad edge, the crossover point is randomly selected.

A *public library* is established to store good components for later reuse by smart mutation. The larger the library, the more effective components can be kept for later reuse, but the likelihood of each effective component being reused is reduced. In this paper, the public library stores 100 good components.

To avoid premature convergence, *mutation* is introduced to change the structure of some randomly selected composite operators. There are three random mutations invoked with equal probability: (a) randomly select a node of the composite operator and replace the subtree rooted at this node by another randomly generated binary tree; (b) randomly select a node of the composite operator and replace the primitive operator stored in the node with another primitive operator randomly selected from the primitive operators of the same number of input as the replaced one; (c) randomly selected two subtrees of the composite operator and swap them. Of course, neither of the two sub-trees can be the sub-tree of the other.

In the smart mutation, mutation point is the parent or child node of a bad edge or a bad node whose goodness value is below the average goodness value of all the nodes. The mutation point is selected among those qualified nodes at random. There are four smart mutations invoked with equal probability: (a) select a parent node of bad edge as mutation point. If the parent node has only one child, the parent is deleted and the child node is linked to the grand parent node (parent node of the parent node), if no grand parent node exists, the child becomes the root node; if the parent node has two children, the parent node and the sub-tree rooted at the child with smaller fitness value are deleted and the other child is linked directly to the grand parent node, if no grand parent node exists, the child becomes the root node; (b) select a parent node of bad edge as mutation point and replace the primitive operator stored in the node with another primitive operator of the same number of input as the replaced one; (c) select two subtree whose roots are child nodes of two bad edges within the composite operator and swap them. Of course, neither of the two sub trees can be the sub-tree of the other; (d) select a bad node as mutation point. Replace the subtree rooted at the node with another randomly generated tree or with an effective component randomly selected from the public library. The first two mutations delete a node that cancel the effect of its child or children; the third mutation moves two components away from unfriendly contexts that cancel their effects and inserts them into new contexts; the fourth mutation deletes a bad component and replace it with a new component or a good one stored in the public library.

We use ε-greedy policy to determine whether a smart operator (smart crossover or mutation) or a random operator (random crossover or mutation) is used. The smart operator and random operator are invoked with probability ε and 1 - ε, respectively. In this paper, ε is a variable and can be adjust by the following formula:

$$\varepsilon = \varepsilon_{min} + (\varepsilon_{max} - \varepsilon_{min}) \times Good_{popu} \qquad (2)$$

where ε_{min} is 0.5 and ε_{max} is 0.9, $Good_{popu}$ is population goodness (the average goodness of composite operators in the population). The reason for using random operators is that smart operators restrict the GP search by biasing the selection of crossover and mutation points. At the beginning, GP should search actively to generate effective components and assemble them together. It is harmful to apply smart operators to restrict the search. Only after effective components are gathered in composite operators, smart operators should be applied to avoid disrupting them and keep them in a public library for later reuse. So, in this paper, smart operators are not used in the first 20 generations.

- **Generational GP:** Generational GP [1] is used to synthesize composite operators. In *generational GP*, two composite operators are selected on the basis of their fitness values for crossover. The two offspring from crossover are kept aside and won't participate in the following crossover operations on the current population. The above process is repeated until crossover rate is met.

Then, mutation is applied to the composite operators in the current population and the offspring from crossover. Finally, selection is applied to select some composite operators from the current population and combine them with the offspring from crossover to get a new population of the same size as the old one. An elitism method is used to keep the best composite operator from generation to generation. At the end of each generation, GP replaces composite operators by their best components to reduce the size of composite operators and avoid overfitting.

4. Experiments

Experiments are performed with real SAR images of sizes 128×128 and 80×80 (tank images). To synthesize composite operators, GP is applied to a region (or regions) carefully selected from the training image to reduce the training time. The generated composite operator is then applied to the whole training image and other testing images. For the purpose of objective comparison, *normal GP* (GP with random crossover and random mutation) and *smart GP* (GP with smart crossover and smart mutation) are invoked ten times with the same parameters and training regions in each experiment and only the average performances are used in comparison. The results from the run in which GP finds the best composite operator among the best composite operators found in all ten runs are reported. The parameters are: population size (100), the number of generation (70), the goodness threshold value (1.0), the crossover rate (0.6), the mutation rate (0.05), and the segmentation threshold (0). The GP program ran on Sun Ultra 2 workstation.

• **Road extraction:** Training image contains horizontal paved road and field, as shown in Figure 1(a); two testing images contain unpaved road vs. field and vertical paved road vs. grass, as shown in Figure 5(a) and 5(d), respectively. Two training regions locate from (5, 19) to (50, 119) and from (82, 48) to (126, 124). Figure 1(b) shows the ground truth. The white region corresponds to the road and only the ground truth in training regions is used in the evaluation.

For normal GP, the goodness values of the best composite operator in the initial and final populations are 0.60 and 0.94, respectively. The goodness of the extracted ROI from training image is 0.90. For smart GP, the fitness and goodness of the best composite operator in the initial population are –2303.6 and 0.45. The corresponding values in the final population are –325.4 and 0.94. The goodness of extracted ROI is 0.91. The best composite operator has 18 nodes and its depth is 13. It has three leaf nodes containing 7×7 median image, which contains less speckle noises due to the median filter's effectiveness in eliminating speckle noise. It is shown in Figure 2, where PFIM15 represents 7×7 median image. Compared to smart GP, the best composite operator from normal GP has 27 nodes and its depth is 16. Figure 3 shows how the average fitness of the best composite operator and average fitness of population over all 10 runs change as GP explore the composite operator space. 10 best composite operators are obtained in the initial and final generations of 10 runs, respectively. Figure 4 shows the utility of primitive operators and primitive feature images in the best composite operators of initial and final generations. To compute utility, we first compute the combined numbers of each primitive operator and primitive feature image in the 10 best composite operators, then divide them by the total number of internal nodes and leaf nodes of these 10 best composite operators, respectively. From Figure 4(b), it can be seen that MED operator has the most frequent occurrence in the best composite operators learned by GP.

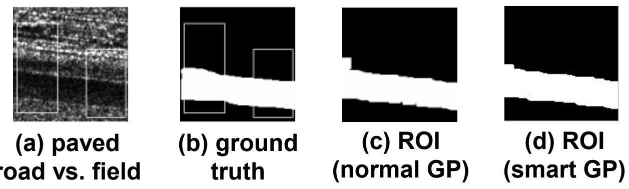

Figure 1. Training SAR image containing road.

(MAX (MAX (MAX (MAX (MAX (SUBC (MUL (DIVC (ADDC (MAX (MAX (MAX (ADDC PFIM15)))))) (DIV PFIM15 (STDV PFIM15)))))))))

Figure 2. Learned composite operator tree in LISP notation.

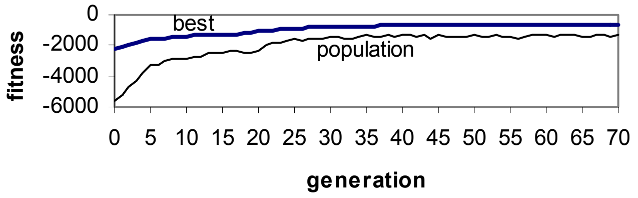

Figure 3. Fitness versus generation (road vs. field).

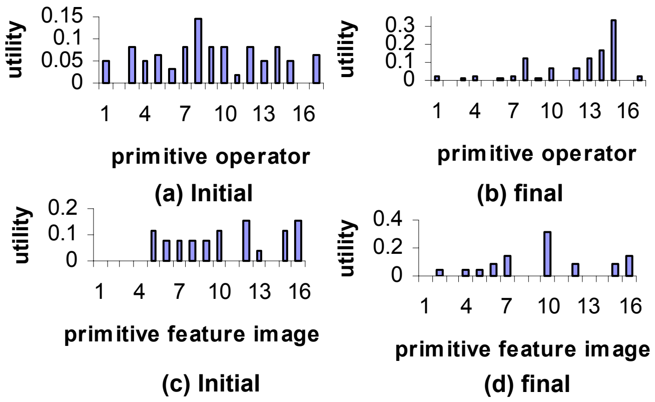

Figure 4. Utility of primitive operators and primitive feature images.

We applied the composite operator to the testing images. Figure 5(b) and 5(e) show the ROIs extracted by normal GP. Their goodness values are 0.90 and 0.93. Figure 5(c) and 5(f) show the ROIs from smart GP. Their goodness values are 0.91 and 0.93. The average running time of the best composite operators from normal GP on training and testing images is 4.2 seconds; the corresponding time of that from smart GP is 2.6 seconds.

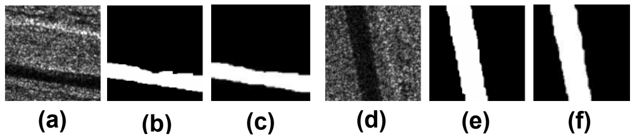

Figure 5. Testing SAR image containing road.

- **Field extraction:** Two SAR images contain field and grass. Figure 6(a) and 6(b) show the training image and the ground-truth. The training regions are from (17, 3) to (75, 61) and from (79, 62) to (124, 122). For normal GP, the goodness values of the best composite operator in the initial and final populations are 0.52 and 0.78. The goodness value of the extracted ROI is 0.88. For smart GP, the fitness and goodness of the best composite operator in the initial population are −7936.2 and 0.39. The corresponding values in the final population are −1999.4 and 0.79. The goodness value of the extracted ROI is 0.90. We apply the composite operator to the testing image containing field and grass shown in Figure 7(a). The goodness values of extracted ROIs from normal GP and smart GP are 0.8 and 0.84, respectively. The average running time of the best composite operators from normal GP on training and testing images is 7 seconds; the corresponding time of that from smart GP is 11 seconds

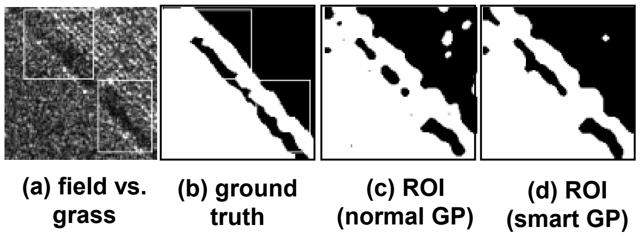

Figure 6. Training SAR image containing field.

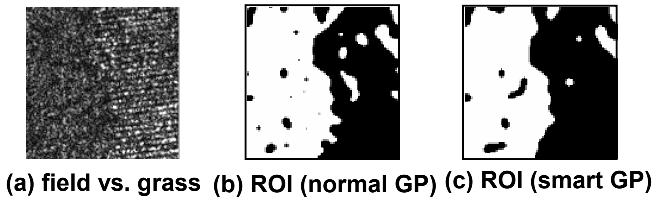

Figure 7. Testing SAR image containing field.

- **Tank extraction:** The training image contains T72 tank under depression angle 17° and azimuth angle 135°, which is shown in Figure 8(a). The training region is from (19, 17) to (68, 66). The testing image contains a T72 tank under depression angle 20° and azimuth angle 225°, which is shown in Figure 11(a). The ground-truth is shown in Figure 8(b). For normal GP, the goodness values of the best composite operator in the initial and final populations are 0.65 and 0.88, respectively. The goodness value of extracted ROI is 0.88. For smart GP, the fitness and goodness of the best composite operator in the initial population are −807.2 and 0.54. The corresponding values in the final population are –190.8 and 0.89. The goodness of extracted ROI is 0.89. The best composite operator has 5 nodes and its depth is 4. It has one leaf node containing 3×3 maximum image. Two internal nodes are MED operator, which is useful in eliminating speckle noise in SAR images. It is shown in Figure 9. The best composite operator from normal GP has 28 nodes and its depth is 17. Figure 10 shows how the average fitness of the best composite operator and average fitness of population over all 10 runs change as GP proceeds.

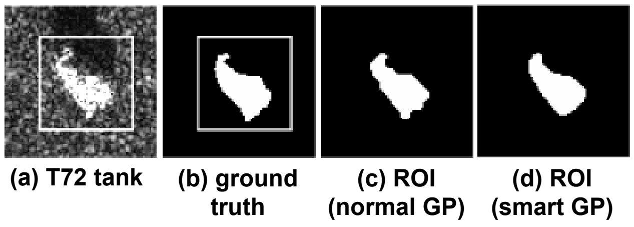

Figure 8. Training SAR image containing tank.

```
(MED (MED (SUBC (DIVC PFIM7))))
```

Figure 9. Learned composite operator tree in LISP notation.

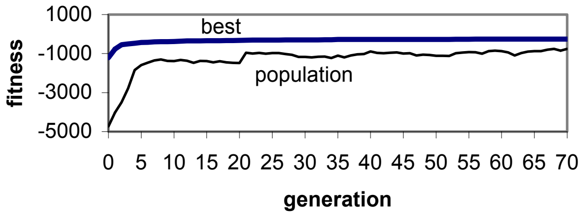

Figure 10. Fitness versus generation (T72 tank).

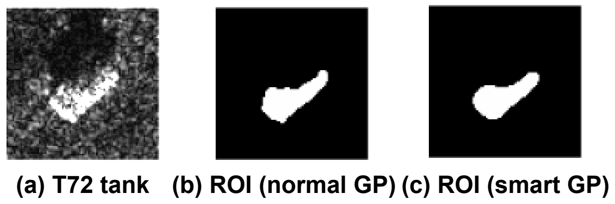

Figure 11. Testing SAR image containing tank.

We apply the composite operator to the testing image. The goodness values of extracted ROIs from normal GP and smart GP are 0.8 and 0.84, respectively. The average running time of the best composite operators from normal GP on training and testing images is 3 seconds; the corresponding time of that from smart GP is 2 seconds.

• **Comparison between normal GP and smart GP:** We compare the performance of smart GP with that of normal GP and only the average performance over all ten runs is used in comparison. Figure 12 shows how the average goodness of the best composite operator improves as the normal GP and smart GP proceed. The thick and thin lines show the goodness of smart GP and normal GP, respectively. It can be seen that the performance of smart GP is always better than normal GP, even in the first 20 generations when both smart and normal GPs use the same random crossover and mutation. The reason is that with MDL-based fitness function used in smart GP, the hard size limit on the composite operator is taken off, greatly alleviating the restriction on the GP search. Table 1 shows the average goodness of the best composite operator in the initial and final populations. Table 2 shows the average size of the best composite operators from normal GP and smart GP. It also shows the average goodness of best composite operators on the whole training image (TrG) and other testing image(s) (TeG). It can be seen that the best composite operators learned by smart GP have better performance and smaller sizes, thus reducing the computational expense in future detection.

Table 1. The average goodness of best composite operators from normal and smart GPs.

Average goodness	Normal GP			Smart GP		
	Road	Field	Tank	Road	Field	Tank
Initial	0.47	0.54	0.49	0.46	0.46	0.41
Final	0.82	0.73	0.85	0.88	0.75	0.86

Table 2. Average size and goodness of best composite operators from normal and smart GPs.

	Normal GP			Smart GP		
	Road	Field	Tank	Road	Field	Tank
Size	29.4	20.2	24.6	24.6	14.9	5.7
TrG	0.79	0.79	0.83	0.86	0.84	0.85
TeG	0.62 0.80	0.67	0.77	0.83 0.92	0.79	0.82

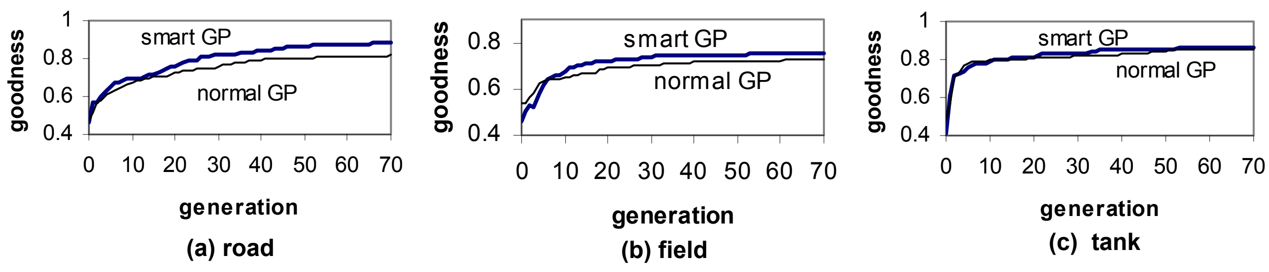

Figure 12. The average goodness of the best composite operators versus generation.

5. Conclusion

In this paper, we evolve composite operators for object detection using smart crossover and smart mutation operators in genetic programming. We design a MDL-based fitness function to account for the size of composite operator into the fitness evaluation. The new fitness function prevents composite operators from growing too large while at the same time imposes relatively less severe restrictions on the GP search. Our experimental results with real SAR images show that with MDL-based fitness function and smart search operators, GP can learn good composite operators more quickly, improving its efficiency. Compared to normal GP, the composite operators learned by smart GP have better performance on the training and testing images and have smaller sizes, reducing the computational expenses during detection.

Acknowledgment: This research is supported by the grant F33615-99-C-1440. The contents of the information do not necessarily reflect the position or policy of the U. S. government.

References

[1] B. Bhanu and Y. Lin, "Learning composite operators for object detection," *Proc. Genetic and Evolutionary Computation Conference*, pp. 1003-1010, July, 2002.
[2] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *The Annals of Statistics*, Vol. 11, No. 2, pp. 416 – 431, 1983.
[3] W. Tackett, "Recombination. selection, and the genetic construction of computer programs," Ph.D thesis, Univ. of Southern California, Dept. of Electr. Engg. Systems, 1994.
[4] P. D'haeseleer, "Context preserving crossover in genetic programming," *Proc. IEEE World Congress on Computational Intelligence*, Vol. 1, pp. 256 – 261, 1994.
[5] P. Smith, "Conjugation – A bacterially inspired form of genetic recombination," In Koza, J. R., editor, *Late Breaking Papers at the Genetic Programming Conf.*, pp. 167 – 176, 1996.
[6] T. Ito, H. Iba and S. Sato, "Depth-dependent crossover for genetic programming," *Proc. IEEE Int. Conf. on Evolutionary Computation*, pp. 775-780, 1998.
[7] D. Knjazew and D. Goldberg, "Solving permutation problems with the ordering messy genetic algorithm," *Advances in Evolutionary Computing – Theory and Applications*, A. Ghoth and S.Tsutsui (Eds), Springer Publisher, pp. 321 – 350, 2002.