Evolving a Spacial Image Analyzer for Tree Detection Using Fitness Sharing

Alex Bailey Brock University
St. Catharines, Ontario
Email: ab04bf@brocku.ca

***Abstract*—This paper discusses a computer vision exercise in which Genetic Programming (GP) was employed to identify areas of trees from satellite images.Previous work showed that providing the GP with more and varied training data produced better results than GP runs with less training data. This work examines the effect of implementing *fitness sharing* and shows that *fitness sharing* produces results which are less varied, and whose individuals have generalized better such that they perform better on unseen data than those individuals produced by a GP which did not implement *fitness sharing*.**

I. Introduction

This assignment evolves filters using Genetic Programming (GP) to identify areas in satellite photographs which contain trees. GP has been applied to a range of similar applications, such as to evolve filters for image analysis[1], detecting ships in satellite photographs[2], and for target discrimination[3]. While these applications are similar, the task of finding trees offers a unique challenge because a cluster of trees in an image appears visually to contain a wide variation of intensities and inconsistent features. Work on the previous assignment showed that the evolution of such filters was possible and that GP produced tree filters that were able to classify the majority of both positive and negative examples on a set of training images and an unseen image. Although this offered a proof-of-concept the results were largely unsatisfactory.

The goal of this assignment is to improve on the methods of assignment 2 and produce some better classifications. Observationally, many of classifications produced by the fitness scoring method previously used seemed biased towards certain classes of training examples. When a certain set of training examples were found to be easily classified by the GP this was exploited during evolution and many individuals began to focus on classifying those easy examples at the cost of not being able to classify some others. Initially this actually created sets of entirely unusable results because, of course, the simplest way to classify a large subset of the training examples correctly is to classify every example as a tree or every example as not a tree. Since there were typically more negative examples than positive examples many runs produced individuals which simply classified everything as not trees. This was solved by an intuitive approach of weighting positive and negative examples so that the set of positive examples was worth as much as the set of negative examples. In this way an individual who classified all negative examples correctly and even one positive example correctly would have a higher fitness than one who classified all negative examples correctly. Of course, it also meant that those individuals who classified all examples as positive were just as fit as those who classified all examples as negative, thereby increasing population diversity.Results improved to a reasonable point after weighting the fitness of examples, however focus was still given to easy to classify examples which provided the same fitness reward as any other example.

One method to differently weight the training examples dynamically is known as *fitness sharing* first introduced by Deb and Goldberg[4]. The idea is that individuals which are similar to each other are reduced in fitness by the requirement to share their fitness scores among their similar counterparts. This brand of fitness sharing required an explicit distance metric to compare individuals, and is referred to as *explicit fitness sharing*. Methods which achieve the same goal which require no such metric are called *implicit fitness sharing*, and were first introduced by Smith, Forrest and Perlson[5]. *Implicit fitness sharing* works by sharing the value of a training example among members of the population which are able to correctly classify that example. The result is that examples which are easier to classify are worth less, and those which are harder to classify are worth more. This means individuals who can classify harder examples are rewarded for their unique behaviour.

This paper examines the implementation of *fitness sharing* to address the issue of individuals who exploit a multitude of easy-to-classify training examples for high fitness gains while choking out unique good solutions. Part of this examination will include a comparison of results produced by a GP which employed *fitness sharing* and one which did not.

II. Experimental Setup

A. Data

The data used for this problem was a set of satellite images captured from Google Maps. The images are provided in pairs – the query image, which the GP processes and the answer image, which provides the expected result while the GP is learning. Figure 1 shows a query image and figure 2 shows an answer image. The red areas on the answer image indicate the areas of the photograph where trees are known to appear and where the GP should classify those pixels positively. Query images were given only in colour.

Fig. 1. A Query Image

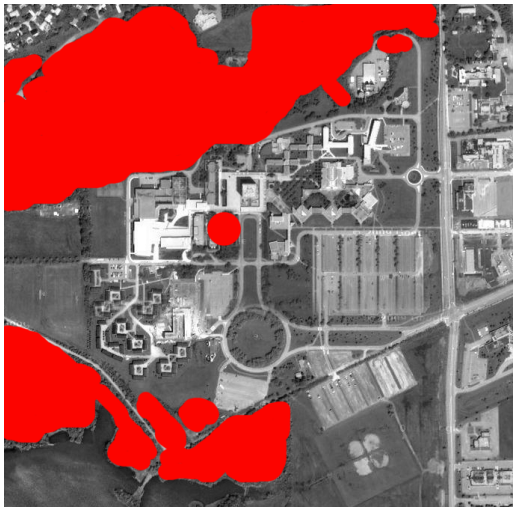

Fig. 2. An Answer Image

To avoid processing the entire images during training, which would be computationally expensive, only a sample of the image data was used to create a training vector for the GP. The sample was created by first dividing up the image into blocks. The number of blocks is referred to as the *sample density*, which was specified in both the x and y direction – x *sample density* being the number of columns, and y *sample density* being the number of columns. Once the blocks were determined, a single pixel from the block was selected randomly to represent that block. This way the image could be sampled evenly and fairly at a range of granularity. Figure 3 illustrates this.

Additionally, because some data from nearby pixels was utilized during image processing only pixels far enough away from the image border were selected for training. This distance is referred to as the *max offset*.

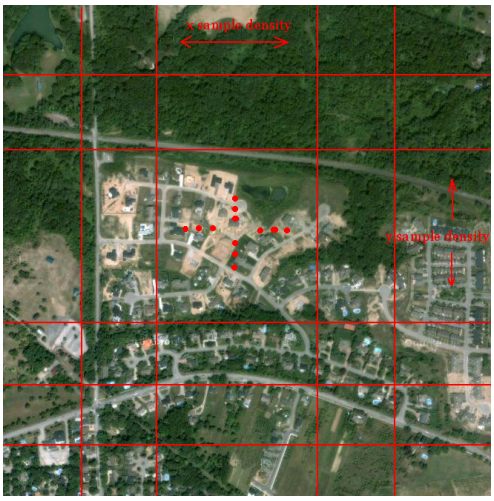

Fig. 3. The Sample Density

B. GP Language

The GP Language consisted of three types, booleans, doubles, and integers. No integer functions were provided to keep the integer values within the maxoffset range. However, integer values could increase or decrease with equal chance by a value of one in either direction due to the mutation operator.

The function set includes the relational $<$, $EqualTo$ and the boolean AND, OR, and NOT, an *IF_THEN_ELSE* function, as well as the following functions. Note that the image processing nodes were inspired by the descriptions given in Robert Flack's 2008 GP paper.

Note that this function set is essentially identical to the function set used for assignment two, with the inclusion of the boolean NOT operation and the modification of all image functions to produce values with respect to a specific colour mask.

True	A Boolean terminal returning the value TRUE.
False	A Boolean terminal returning the value FALSE.
One	An Integer terminal returning 1.
Zero	An Integer terminal returning 0.
DoubleERC	An Ephemeral Random Constant containing a double.
Integer	An Ephemeral Random Constant containing an Integer.
PixelIntensity	The intensity of a pixel.
R	The Red value of a pixel
G	The Green value of a pixel
B	The Blue value of a pixel
RMask	Specifies the use of the R channel
GMask	Specifies the use of the G channel
BMask	Specifies the use of the B channel
RGBMask	Specifies the use of the RGB channel
+	Addition of doubles.
-	Subtraction of doubles.
*	Multiplication of doubles.
%	Protected division.
Min(a, b)	Returns the minimum of two values.
Max(a, b)	Returns the maximum of two values.
Exp(a)	Returns e^a.
Ln(a)	Returns the natural log of a.

AvgIntensityMask/b$w(x, y, c, withCentre)$
Returns the Average Intensity of an area l by w offset by x and y from current pixel.
MaxIntensity(x, y, c)
Returns the maximum Intensity of an area l by w offset by x and y from current pixel.
MinIntensity/b$w(x, y, c)$
Returns the minimum Intensity of an area l by w offset by x and y from current pixel.
SquareSumDiff$(x, y, c, withCentre)$
Returns the difference of the sum of squared intensities of an area l by w offset by x and y from current pixel. If $withCentre$ is FALSE, then the centre pixel is not included.
VarianceMask$(x, y, c, withCentre)$
Returns the variance of the intensities of an area l by w offset by x and y from current pixel. If $withCentres$ is FALSE, then the centre pixel is not included.

Note: l and w are always one of 5, 11, 21 where $l = w$. The argument $c \in \{RMask, GMask, BMask, RGBMask\}$ indicates that the operation is performed with respect to the specified colour channel(s).

C. Training and Testing Sets

Training Images were all loaded into the GP system, and pixels were selected as training examples by first breaking up the images into blocks according to the sample densities specified. One random pixel was chosen from each block, and its (x, y) location on the image was stored, as well as a reference to the image to which it belonged. In this way, pixels from multiple images could be used as training examples. This was established empirically to give better results over the set of training images and on unseen images than simply training the system on pixels from a single image.

Once the training examples were selected, the set of examples was shuffled and the first 60% were used for training, and the last 40% were used for testing and validation. Finally, at every tenth run and after validation, the program was run on an entirely unseen image whose pixels were not mixed in with the training set. At this step, a result mask was generated for this image as well as the rest of the images in the training set.

Three images in total were loaded into the system. Two, were used to create the training data sample and one was not used during training at all. On completion of a run every non-edge pixel of each of the three images was processed to generate a result mask. The total correct positive classifications and the total correct negative classifications were summed during the creation of the result mask for all three images. The totals were then expressed as percentages, and given as additional validation data.

The number of training examples present in the system is dependent on the sample densities and the number of images provided. In general the number of training examples t_n is equal to:

$$t_n = \quad SD_x \cdot SD_y \cdot I \cdot t_p$$

Where SD_x is the *x sample density* and SD_y is the *y sample density*, I is the number of training images, and $0 < t_p \leq 1$ is the portion of examples to be used for the training set.

In this case there were 2250 examples sampled, and a portion of 60% of those were used during the training process leaving a total of 1350 examples used for training.

D. Fitness Calculation

Two different methods for calculating fitness were used. The first fitness calculation which, for convenience, will be referred to here as *vanilla fitness* was computed in a similar way to fitness in assignment two. The second method used was *fitness sharing*. Both methods are detailed in the following sections.

1) Vanilla Fitness: The raw fitness for individual x $f(x)_{raw}$ was computed as a weighted sum of classification errors. The standardized fitness $f(x)_{sd}$ is also given.

$$f(x)_{raw} = \Sigma_{i=0}^{n} pw(Error_{pos}) + (nw)(Error_{neg})$$
$$f(x)_{sd} = frac11 + f(x)_{raw}$$

where n is the number of training examples, $Error_{pos}$ are the false positives, $Error_{neg}$ are the false negatives, and pw and nw are weights computed as follows:

$$pw = \quad \frac{NumberofNegativeExamples}{NumberofPositiveExamples}$$

$$nw = \quad \frac{NumberofPositiveExamples}{NumberofNegativeExamples}$$

2) Fitness Sharing: One of the advantages to using fitness sharing is that unique behaviour is rewarded. This leads to populations with higher diversity, and fitness sharing has even been shown to be an effective niching technique[6]. *Implicit fitness sharing* was implemented for this problem with a slight modification.

Since Sean Luke's ECJ system by default attempts to minimize fitness the reward given for each training example was simply equal to the number of individuals who correctly classified it. In this way examples which were classified correctly less often were worth more than those which were classified correctly very often. Then the fitness of each individual was based on the sum of rewards given for the examples correctly classified. However, the obvious flaw in this approach is that the ideal fitness of zero is awarded to those examples which classify nothing at all correctly. This problem could be solved by performing some book-keeping on incorrectly classified examples and modifying the fitness in a negative way to reflect the difficulty of the example incorrectly classified. While this makes some sense, intuitively it does not reflect the end goal. The ideal individual would be one who classifies all examples correctly, and so it seems incorrect classifications should incur maximum penalty regardless of their difficulty. This was the approach taken, and so incorrect classifications incurred a fitness penalty equal to the greatest penalty possible and positive classifications were rewarded proportionately to how many individuals correctly classified that example. This method was found to both favour those individuals who correctly classified hard-to-classify individuals and those individuals who classified as many examples as possible correctly. In addition it was found that weighting the fitness penalty given by incorrect classifications as in the *vanilla fitness* function was also beneficial. Standardized fitness was computed as in *vanilla fitness*, and the *implicit fitness sharing* function for raw fitness $f(x)_{raw}$ is computed as follows:

$$f(x)_{raw} = \Sigma_{i=0}^{n} Reward_i + pw(Error_{pos}) + (nw)(Error_{neg})$$

where n is the number of training examples, $Error_{pos}$ are the false positives, $Error_{neg}$ are the false negatives, and pw and nw are weights computed as in *vanilla fitness*. $Reward_i$ is the reward given for training example i and is computed as follows:

$$Reward_i = \Sigma_{k=0}^{N} correct(Individual_k)$$

where $correct(Individual_k) = 1$ if $Individual_k$ produces a correct classification for example i and 0 otherwise, and N is the number of individuals.

E. Experimental Setup and Results

The goal of the experiments performed was to examine the effects of *fitness sharing* on the quality of results and the resulting behaviour of the GP. To this end, two sets of data were collected. Each set was gathered using identical parameters and only the measure of fitness was changed. One set was gathered with fitness sharing, which will be referred to here as the *sharing set* and one set was gathered without fitness sharing, which will be referred to here as the *unsharing set*. The parameters used for both the *sharing* and *unsharing* set are given in table I.

TABLE I
GP PARAMETERS

Parameter	Value
Generations	20
Population Size	300
Population Initialization	Ramped Half and Half
Grow Min	2
Grow Max	8
Selection	Tournament, k=4
Crossover	Subtree Crossover
Mutation	Point Mutation, Grow, min depth=3, max depth=7
Number of Training Images	2
Number of Unseen Images	1
x-sample density	45
y-sample density	25
training-portion	0.6
number of training examples	1350
Max Offset	31
Runs	20

1) Convergence: Analysis of the convergence plots is decidedly uninteresting. Comparing the two raw fitness scores is an exercise in comparing apples and oranges because they represent two very different things. The standardized fitness score was computed based on the raw fitness in both cases and also provides little means for comparison. However, the shape of the convergence plots can be compared by visual inspection. The convergence plots are based on the average of the average raw fitnesses per generation over 20 runs as well as the average of the best raw fitnesses per generation over 20 runs.

Both sets exhibit what seems to be stagnation after less than ten generations and have more or less the same shape. As will be shown in this section, good results were still obtained by the GP runs so the stagnation is likely because fit individuals or individuals which are much more fit than the individuals generated in the initial population are quickly found and the GP quickly converges there. It is worth noting that there seems to be slightly more activity in the population as a whole in the sharing set, as indicated by the slight oscillation of the mean raw fitness line. This may indicate a more varied set of individuals exist in the sharing population.

2) Result Mask Discussion: A comparison of the best and worst result masks generated by the best and worst individuals of the sharing and unsharing set suggest some interesting differences between the individuals generated. Each result mask generated by the sharing set is followed by its corresponding partner produced by the unsharing set for ease

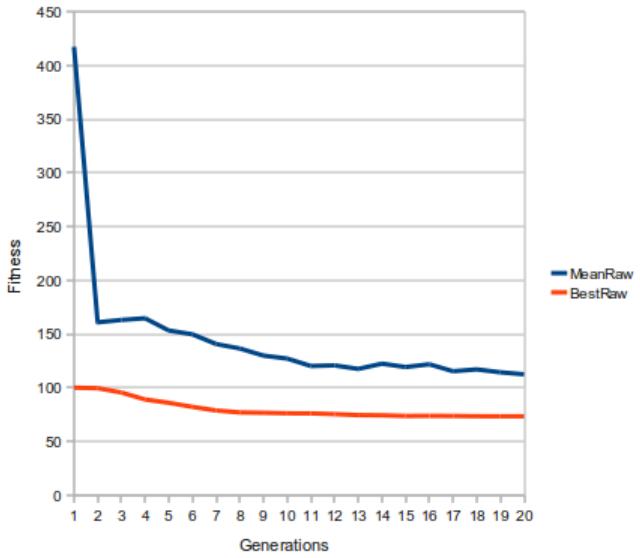

Fig. 4. Average convergence of the average and best individuals in the n

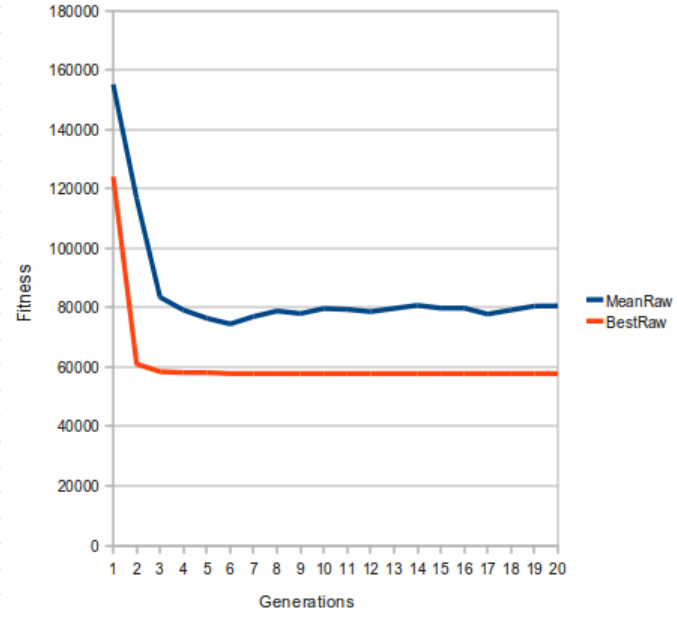

Fig. 5. Average convergence of the average and best individuals in the sharing set over 20 runs

of visual inspection. Due to space constraints the result masks are located in Appendix A.

Table II gives a legend for reading the result masks.

TABLE II
RESULT MASK LEGEND

GREEN	True Positive
BLACK	True Negative
RED	False Negative
BLUE	False Positive

Consider the best and worst results from the sharing set. Even the worst results from the sharing set are better than the best results from the unsharing set on the unseen image.

Examining the remaining result masks shows a tendency

for the unsharing set to optimize for a particular image, and at times produces good results for one image, but entirely neglects other image. This strategy works in the *vanilla fitness* model because all training examples are worth the same amount, so a greedy strategy of specializing to produce the most correct classifications works well to generate a good fitness score. However, this strategy comes at the cost of loosing generalization. This becomes apparent when examining the masks produced for the unseen image. The unsharing set at times produces very good results for the training images but nearly always very poor results on the unseen image. This could be attributed to overtraining, however similar behaviour is observed in even the worst individuals. More likely this disparity is due to some difference between the images, perhaps in overall pixel intensity or the colours present. Since the unsharing set does not emphasize a variety of different kinds of training examples it appears to not generalize as well as the sharing set.

Overall, the sharing set seems to exhibit less of a difference between extremes than the unsharing set. This suggests the sharing set produces more consistent results than the unsharing set. Sub-subection II-E3 examines this possibility in more detail.

3) Numeric Results: After each run the portion of total correct positive classifications over all three images was recorded as well as the total correct negative classifications. This figure included all the pixels used to generate the result masks for all three images. This represents a substantial amount of unseen data as the images are approximately 600 by 600 pixels multiplied by three images gives 1.08×10^6 pixels. A vast number more than the training set which only contained 1350 pixels and was only sampled from two of the three images.

These portions of correct classifications were then averaged over the 20 runs and some summary statistics were generated and are presented in tables III, and IV.

TABLE III
SHARING AND UNSHARING: POSITIVE CLASSIFICATIONS

Statistic	Sharing Value	Unsharing Value
min	0.58	0.16
max	0.98	0.82
mean	0.81	0.55
variance	0.0158	0.0367

TABLE IV
SHARING AND UNSHARING: NEGATIVE CLASSIFICATIONS

Statistic	Sharing Value	Unsharing Value
min	0.49	0.89
max	0.89	0.99
mean	0.75	0.94
variance	0.0176	8.892E-4

The sharing method exhibits less variance in terms of positive classifications as well as less difference between the least and most positive classifications made on average. It classifies negative examples correctly on average with approximately the same frequency and with near the same spread. Compared with the unsharing method which seems to classify positive examples correctly much less of the time on average and with a spread of data larger than the sharing method as indicated by the variance and with a much wider range. This confirms the suspicions which arose during visual inspection of the result masks.

Simply comparing average classifications is likely enough to convince the reader that the sharing method outperforms the unsharing method. The sharing method on average classifies 81% of the positive examples correctly and 75% of the negative examples correctly. The unsharing method only manages to classify 55% of positive examples correctly but does manage to classify 94% of the negative examples correctly. The unsharing set's low success with positive classifications and high rate of success with negative classifications suggests that the good classification rate of negative classifications may simply be due to the fact that negative examples are more plentiful so underclassifying positive examples or simply always classifying as negative is a "cheap" way to yield some successful classifications and may not indicate that the method was actually any better at learning how to correctly classify a negative example.

For completeness an unpaired two-sample t-test was done to compare the classification results produced by the unsharing and sharing set. One test was done to compare the means of correct positive classifications and one test was done to compare the means of correct negative classifications. The results are given in table V

TABLE V
P-VALUES, UNPAIRED TWO-SAMPLE T-TEST TO COMPARE CLASSIFICATION RATES BETWEEN SETS

	P-Value
Positive Classifications	1.36E-5
Negative Classifications	4.80E-6

The p-values are low enough to state that the means are significantly different at even a 99% confidence interval.

4) The Best Individuals: The best individual produced by the sharing set was a concise program which primarily made use of red and blue colour information and only required two filters to produce its impressive 0.94 correct positive rate, and 0.86 correct negative rate.

```
Tree 0:
 (OR (EqualTo 0.7128291716728568 R)
     (LessThan
      (AvgIntensityMask11b11 1 2 BMask true)
      (% (MaxIntensity23b23 0 -25 RMask)
  (Ln (If_Else true
PixelIntensity R)))))
```

The best individual produced by the unsharing set was a larger tree which also appears to rely on red and blue colour information. This program had a very good 0.82 correct positive rate, and a 0.95 correct negative rate.

```
Tree 0:
 (LessThan (* (SquareSumDiff5b5 0 1 BMask
    (LessThan
    (VarianceMask23b23 0 0 BMask true)
    (VarianceMask23b23 0 0 BMask
    (AND (OR false true)
    (LessThan (MinIntensity5b5 -16 0 BMask)
    (MinIntensity23b23 1 27 GMask))))))
    (If_Else (AND (AND true true)
    (LessThan (SquareSumDiff23b23
        -13 8 RMask (AND (OR
    (LessThan PixelIntensity G)
    (AND true true)) (OR true true)))
    (MinIntensity23b23 1 27 GMask)))
    (AvgIntensityMask5b5 1 29 GMask
    (EqualTo PixelIntensity
    0.9603733026293302)) (If_Else
    (AND true true) (AvgIntensityMask11b11
       0 -8 RGBMask true)
      (Max PixelIntensity R))))
    (SquareSumDiff23b23 -13 8 RMask
      (AND (OR
(LessThan PixelIntensity G)
(AND true true))
         (OR true true))))
```

III. Conclusion

Fitness sharing seemed to produce more consistent results over multiple runs than the results produced without *fitness sharing* and the positive classification average rate increased dramatically and the increase was shown to be statistically significant even at the 99% confidence level. The individuals produced with *fitness sharing* also seemed to exhibit better ability to generalize when faced with previously unseen data than the individuals produced by the runs without fitness sharing. Visual inspection of the result masks produced by the best and worst individuals from each set showed that the fluctuation in result quality was visibly apparent between runs with the unsharing set and that there was less disparity between runs with the sharing set. Overall the application of *fitness sharing* behaved roughly as *fitness sharing* is expected to perform and proved to be the superior of the two GP fitness models examined here for the tree filter problem.

IV. References

References

[1] R. Poli, "Genetic programming for image analysis," in *Proceedings of the First Annual Conference on Genetic Programming*, ser. GECCO '96. Cambridge, MA, USA: MIT Press, 1996, pp. 363–368. [Online]. Available: http://portal.acm.org/citation.cfm?id=1595536.1595587

[2] D. Howard, S. C. Roberts, and R. Brankin, "Target detection in sar imagery by genetic programming," *Adv. Eng. Softw.*, vol. 30, pp. 303–311, May 1999. [Online]. Available: http://portal.acm.org/citation.cfm?id=315021.315022

[3] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 303–311. [Online]. Available: http://portal.acm.org/citation.cfm?id=645513.657740

[4] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 42–50. [Online]. Available: http://portal.acm.org/citation.cfm?id=645512.657099

[5] R. E. Smith, S. Forrest, and A. S. Perelson, "Searching for diverse, cooperative populations with genetic algorithms," *Evolutionary Computation*, vol. 1, pp. 127–149, 1993.

[6] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Champaign, IL, USA, 1995, uMI Order No. GAX95-43663.

APPENDIX

A. Best Result Masks

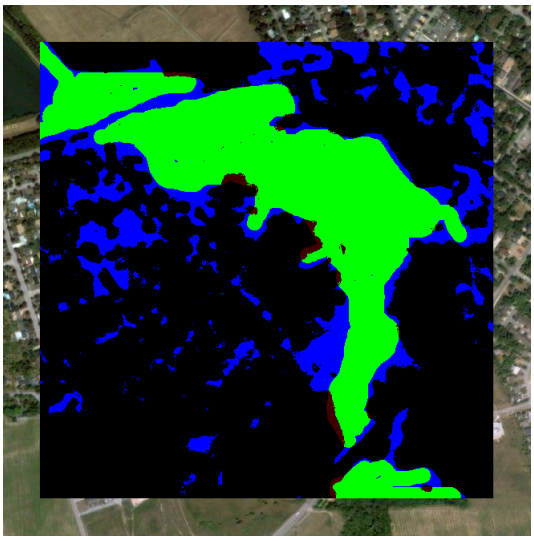

Fig. 6. Best Area 3 result mask produced by the sharing set

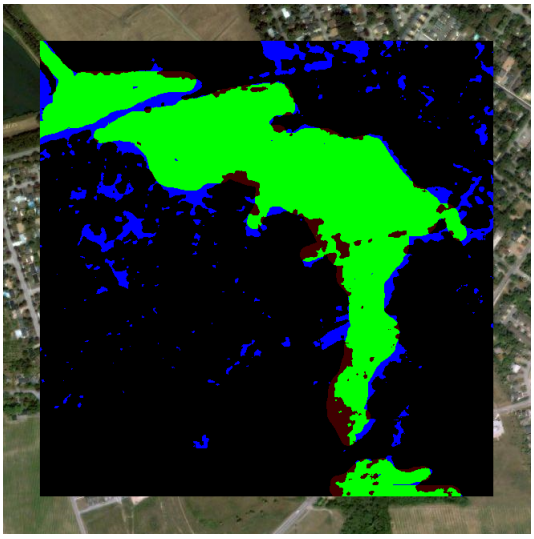

Fig. 7. Best Area 3 result mask produced by the unsharing set

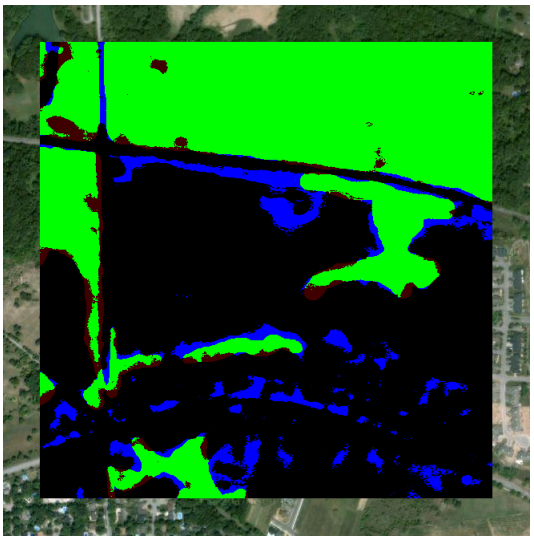

Fig. 8. Best Area 4 result mask produced by the sharing set

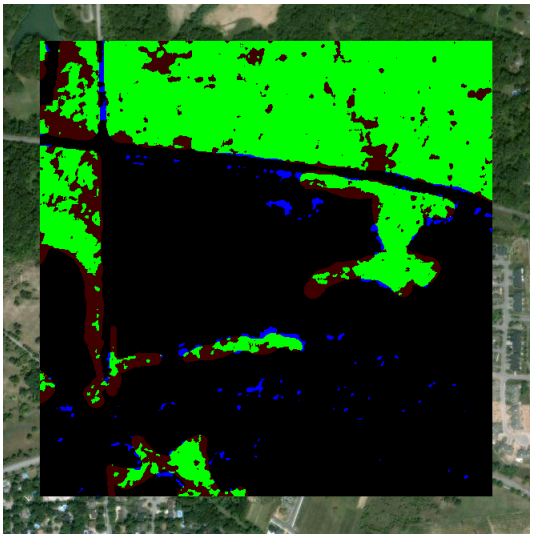

Fig. 9. Best Area 4 result mask produced by the unsharing set

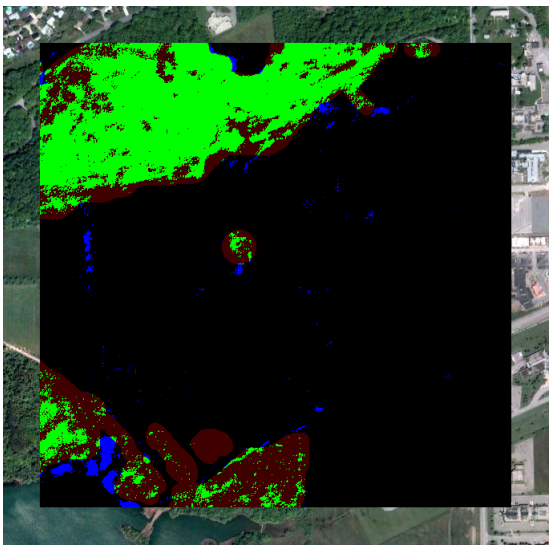

Fig. 10. Best Unseen result mask produced by the sharing set

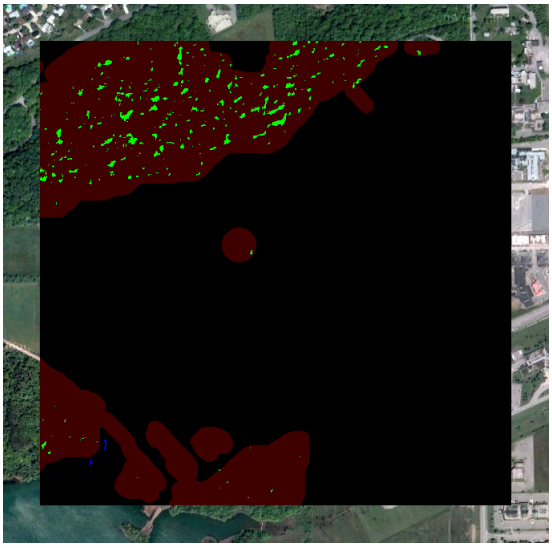

Fig. 11. Best Unseen result mask produced by the unsharing set

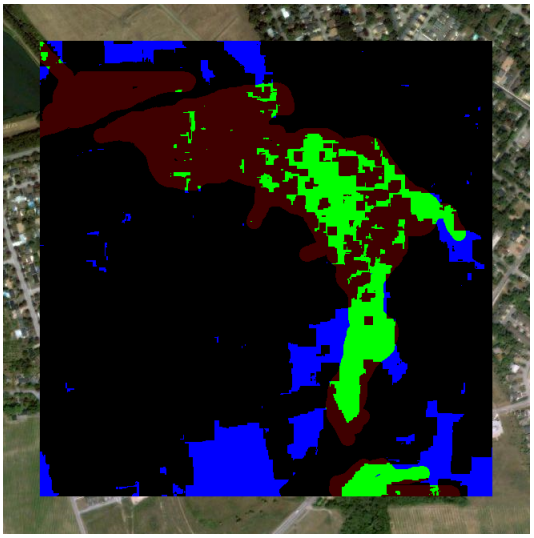

Fig. 12. Worst Area 3 result mask produced by the sharing set

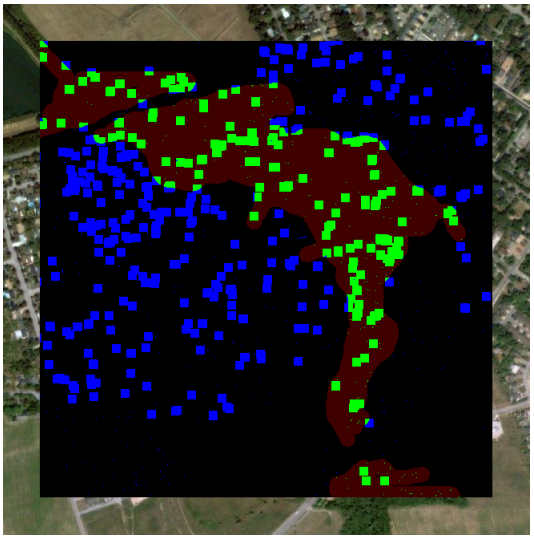

Fig. 13. Worst Area 3 result mask produced by the unsharing set

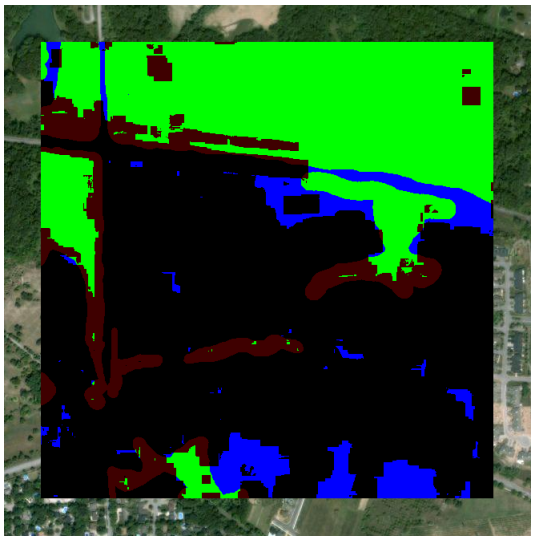

Fig. 14. Worst Area 4 result mask produced by the sharing set

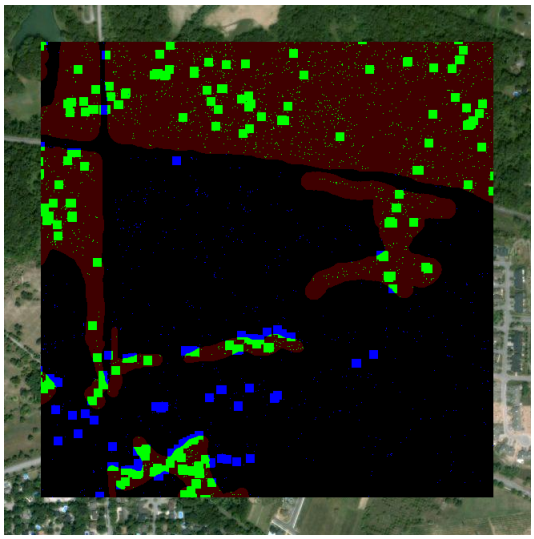

Fig. 15. Worst Area 4 result mask produced by the unsharing set

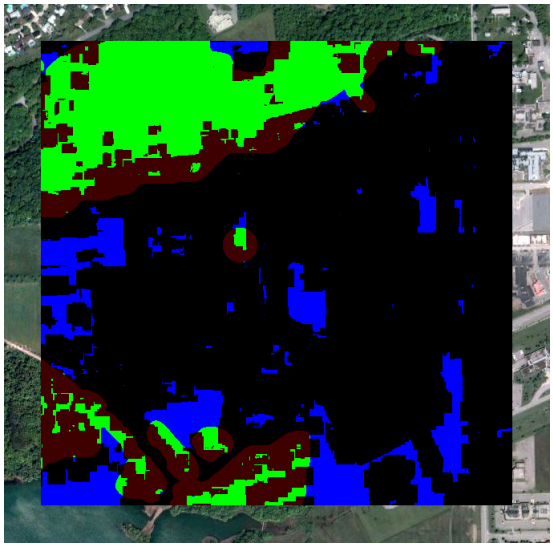

Fig. 16. Worst Unseen result mask produced by the sharing set

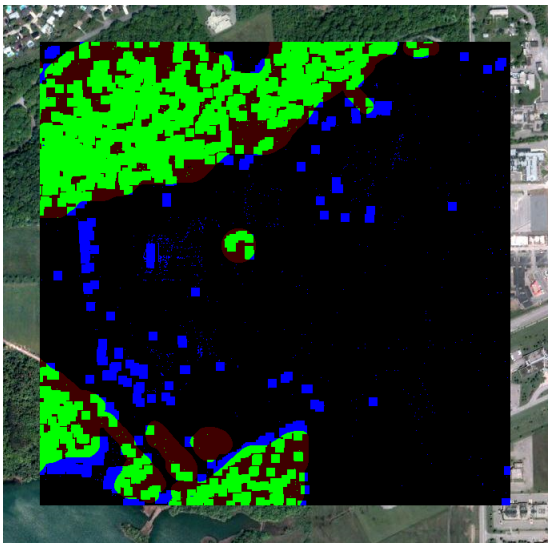

Fig. 17. Worst Unseen result mask produced by the unsharing set