Evolving Feature Extraction Transformations for Chain Code Based Shape Matching

Nancy Smith
Graduate School of Computer and Information Science
Nova Southeastern University
smitnanc@nova.edu

Abstract

In pattern recognition, there is a trade-off between the complexity of the feature extraction and the classification steps. The closer the feature extraction methodology can bring the feature vector to match the model, the easier it is to properly classify it. In the domain of image processing, the feature vector may take the form of a chain code. Chain codes are useful to reduce the dimensionality of shape representations. Chain codes are invariant to translation, but not to rotation or scale. One of the challenges of classifying a chain code is the many variations of transformations required to fit the model of the image. This paper presents a genetic algorithm that evolves an optimal transformation for the chain code of a shape that has been transformed in a prescribed way.

1. Introduction

Feature extraction is an important step in pattern recognition. Its purpose is to obtain an effective representation of an object from the raw data for input into a classifier. The following section will provide a context for where feature extraction lies within the field of pattern recognition, with a focus on image processing.

Pattern recognition can be separated into the following components: sensing, segmentation, feature extraction, classification, and post-processing [3]. In the context of visual pattern recognition, segmentation narrows the range of interest to a partition of the image. Feature extraction constructs a representation of the characteristics of that partition that can be analyzed for classification. Depending on the object to be classified, the features must be invariant to transformations such as size and rotation. Classification then uses that representation, or feature vector, to categorize the object.

There is a trade-off between feature extraction and classification. A perfect feature extractor would render the classification task trivial, whereas an omnipotent classifier would need no feature extractor [3]. However, the dimensionality of the data in an image makes the omnipotent classifier far from attainable. The performance of many classifiers is a function of the dimension of the dataset. In other classifiers, a high dimension causes an overfitting of the model that can result in misclassifications [3].

There are two major categories for reducing dimension, feature selection and feature transformation. Feature selection keeps a subset of features needed for classification, whereas feature transformation constructs new features out of the original ones. One of the common ways to perform feature selection from an image is to obtain the edges using an edge extraction algorithm. The edges may be further manipulated to find a promising subset of edges.

Feature transformation can further reduce the dimensionality by representing the edges in a more compact form. For example, a matrix of pixels can be converted into a list of line segments and angles. A line drawing can be converted into a sequence of chain codes. Further transformations may be applied to normalize the data by operations such as rotation and translation. The closer the feature extraction methodology can bring the feature vector to match the model, the easier it is for the classifier to correctly identify the object.

There are a number of variations on the preceding steps in the literature. A classifier may have a pixel map as input. Transforms can be performed on pixel maps as well as on more abstract representations [4]. Feature vectors may be represented in a form that is invariant to transforms [12]. The region of interest can be identified before edges are extracted [8]. Although the optimal methodology is still being sought, it is clear that reducing the dimensionality and representing the image in a form that can be matched to a model is a highly desirable goal which is still an area of much research.

One of the difficulties in generating the optimum feature vector for classification lies in the many combinations of transformations that may be required to fit the model of the image. A genetic algorithm (GA) is able to do a parallel search guided by a fitness function that measures the similarity between the features of the image and the features of the model. The problem with using a GA is that they are sensitive to parameters such as population size, crossover and mutation probabilities [9]. General guidelines

for GA parameters have been proposed in the literature [6] [2]. Several researchers have performed research on how to further optimize details of the GA, such as population size [7], crossover techniques [2], and selection methods [5].

In this paper, the application of genetic algorithms to feature extraction transformations is examined. The feature vector is in the form of a chain code. The genetic algorithm evolves transformations of rotation and scaling independently in the horizontal and vertical dimensions. The optimization of the GA parameters that are critical to the success of a GA in this domain is explored by comparing commonly used guidelines with more optimized approaches suggested in the literature.

2. The Chain Code Transformation Problem

A chain code is a sequence of directions that follows the outline of a shape. A code is assigned to each of the possible directions. In the following example, 2 indicates up, 0 indicates right, and so on. For example, the odd shape chain code for Figure 2 starting on the lower left corner is 221221076676655433.

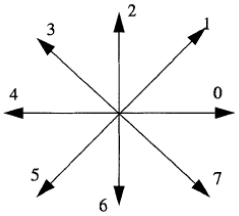

Figure 1: Chain code directions


Figure 2. Odd shape

It is clear that such a chain code is invariant to translation. But in the context of shape matching, small changes in orientation and scaling can cause large changes in the chain code. A simple 90 degree rotation results in a chain code of 211007007654454433 which is not similar to the original chain code.

Figure 3 illustrates the problem underlying transformations of chain codes. Both shapes are scaled up 100% in the horizontal direction, using slightly different algorithms for whether to insert horizontal edges before or after sloping edges. The effect of unguided decision making is exaggerated when the length of the chain code is small.


Figure 3: Horizontal Scaling of Odd Shape

3. Chain Code Matching Techniques

To avoid the problem of chain code transformations, a variety of methods have been developed for shape matching with the chain code representation.

Researchers [1] developed a similarity measure using a histogram of the number of each direction in a chain code. For example, the chain code 2120766543 has one 0, one 1, two 2's, one 3, one 4, one 5, two 6's, and one 7. It's histogram would therefore be 11211121. Differences in scale were addressed by dividing the histogram by the length of the chain code, with the assumption that the length of the chain codes changes proportionately to the scaling. The problem illustrated in Figure 3 shows this assumption has limitations due to the discrete or step nature of chain codes when the grid size is constant. Rotation was compensated for by sorting the histograms. For the shape in Figure 2, the histogram of 221221076676655433 is 12421242, while its rotated chain code of 211007007654454433 has a histogram of 42124212. The histograms are matches when sorted. The histogram is not unique to these figures and cannot be used to identify a match, but is useful in identifying possible matches.

In other research [10], a technique for normalizing a chain code was developed. To account for rotation and scaling, they determine the bounding box and find the greatest length of each of the two chain codes. They reorient and resize one of the boxes to match the other, and recalculate its chain code. The shape is rotated 180 degrees in the horizontal and vertical directions to get a total of 4 variations possible with this technique. The researchers noted that chain codes are difficult to compare. They developed a similarity measure based on the grid cells covered by the shapes.

3. Genetic Algorithm

A genetic algorithm uses the principal of natural selection to evolve the fitness of a population under the given environmental pressures. The algorithm starts with a population of random solutions. The best solutions mate and recombine to form new solutions that become part of the population. The basic algorithm follows:

```
initialize the population
calculate the fitness of each individual
while ( (solution not found) and (number of runs < max))
{
    select parents
    crossover pairs of parents to form children
    mutate the children
    calculate the fitness of the children
    select individuals to form next generation
}
```

There are no strong conclusions on when a genetic algorithm will outperform traditional methods [11]. If the search space is large or not understood, or if the fitness function is noisy, and if a global optimum is not required, GAs can be effective. Their performance is highly dependent on the details of the algorithm.

Two components unique to each problem that are critical to the success of a genetic algorithm are genotype (chromosome representation) and the fitness function used to guide the evolution of the population of potential solutions.

Other GA parameters have guidelines backed by research, but are also highly dependent on the specific problem. These include selection, crossover, mutation, and population size. More often, researchers use guidelines and tweak the parameters until the results are good enough, and this is reflected in the literature. Often in papers that emphasize the use of a GA, it is clear upon reading the research that the GA is perceived as a means to the end, and not the focus of the research. This makes it difficult to determine optimum GA settings. For example, researchers have independently found the best mutation rate to be .01 and .005-.01 [6][14]. However, it is quite common to find rates up to 30% in the literature.

Another issue is the interdependence of the GA parameters. Population size, crossover rate, and mutation rate cannot be optimized independently. The prospect of varying 3 parameters to find the optimal combination makes the analysis more complicated.

4. Shape Matching GA's

The shape matching problem has been defined as falling into the following four categories: searching a database for a similar shape; determining whether a given shape matches a model or class closely enough; constructing a shape of fewer elements that maintains the salient features of the model; and transforming a shape to minimize the dissimilarity between it and the model [14]. In a genetic algorithm, the similarity measure becomes the fitness function that guides the evolution of the population.

The fitness function is critical to the success of a genetic algorithm. For chain codes, the Hamming Distance between two chain codes would be considered a deceptive measure for a GA since a small change in rotation or scaling often results in a large change in the chain code.

In a GA used to evolve transformations on shapes represented by the x-y coordinates, the Hausdorff distance was used as a fitness measure [4]. This measure works well since it accurately reflects the degree of rotation or scaling based on point location. Its use cannot be extended to chain codes since the distance does not correlate with the changes in a chain code caused by scaling or rotation.

The histogram method [1] works well for chain codes. An unsorted histogram was used as the fitness function for the chain code GA.

Another decision unique to the shape matching domain is the genotype. The genotype is the set of genes in a chromosome, which result in the phenotype, or characteristics of the individual. The chromosome in a genetic algorithm is a potential solution. The genes are the set of transformations that comprise that solution. The phenotype is the resulting chain code after the chromosome has been applied to the input shape.

In another GA that was developed to evolve transformations [4], the genotype consisted of a number of bits for the x-translation, y-translation, x-scale, y-scale, and rotation. These bits represented number of pixels, except for rotation which was in degrees. It gave rise to a phenotype of a pixel map of the image.

The genotype chosen for this research is comprised of genes in the form of floats for x-scale, y-scale, and rotation. The scaling alleles represent percentage of increase or decrease in size, while the rotation allele represents degrees. The goal of the GA is to evolve a set of genes that when applied to the input chain code creates the phenotype of a chain code matching the original model.

5. Transforming Chain Codes

At the heart of the chain code GA is a new algorithm for transforming chain codes. As illustrated in Figure 3, scaling scenarios can have different options which are equally valid. With a GA, it isn't necessary to choose one option over another. Instead, we randomly choose one of the valid options. Over the course of many generations, the best option will appear and be chosen for mating, as it will appear more fit than its counterparts as measured by the fitness function.

The size of the object can be increased or decreased independently in the horizontal and vertical dimensions. In this preliminary research, the granularity of the scaling is restricted to factors of two. In each generation, the size of the dimension being scaled will either double, halve, or stay the same. It is anticipated that further work will enable the generalization of the scaling factor.

A high-level description of the scaling algorithm follows:

For each direction in a chain code, perform one of the following actions based on the values of the horizontal and vertical scaling parameters:

- Keep this direction
- Keep this direction and duplicate it as next direction
- Keep this direction and add a new direction as next.
- Add a new direction and keep this direction as next.
- Skip this direction and keep the next direction.
- Keep this direction and skip the next direction

Rotation is implemented as a step function, at 22.5 degree intervals. This is equivalent to eight intervals around a 360 circle. An allele of 80 falls closest to the 90

degree interval and each chain code is modified to reflect two steps around the circle. For the chain code of 217644, a 90 degree rotation becomes 075422.

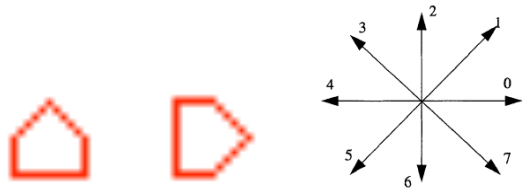

Figure 4: Rotation of Hut Shape Chain Code

6. The Chain Code Evolution GA

The research can be described as follows: given the chain code of a polygonal shape that has been modified in a prescribed way, evolve the transformations that most closely matches the original model. Vary one of the GA components and analyze the difference in performance. Repeat for each of the following components: fitness function, selection method, crossover method, mutation rate, and population size.

Genotype

The encoding of the chromosomes, or genotype, is one of the most important factors in the success of a genetic algorithm [11]. The genotype used is a list of chain codes. The transformations that were considered are rotation and scale. The chromosome is comprised of three genes, each of which is a four-byte float variable.

Fitness Function

The fitness function used is an unsorted histogram of chain code directions since it is fast and effective. This is compared with the Hamming distance function.

Selection Methods

The ideal selection method has been described [5] as one that balances exploitation and exploration. Converging too quickly can lose building blocks through spurious linkage or bad luck. One of the ways to obtain correct convergence is to slow the growth rate down, particularly in the beginning of the evolutionary cycle, and then allowing it to speed up to reduce the time complexity.

The ranking selection method was used to choose the mating population in the chain code GA. The fitness of each chromosome is calculated, and parents are selected at random from the top 20% of the population. The top 20% are carried into the next generation without modification (elitism), so that 80% of the population is replaced with modified chromosomes in each generation. This prevents the population from converging too quickly and losing genetic diversity. Ranking keeps the selection pressure low when the fitness variance is high, and vice versa [11].

Ranking will be compared with fitness-proportional selection. Holland's original GA used this selection method, also known as roulette wheel, and it remains a commonly employed selection method in the current literature. In the roulette wheel method, individuals are assigned a slice of a roulette wheel proportional to their fitness. When the wheel is spun, the individual assigned to the slice the wheel lands on is chosen to be a parent. Elitism is again used to prevent highly fit individuals from being lost to crossover and mutation.

Crossover

The ideal number of crossover points is that which balances the pressure between exploitation and exploration. The higher number of crossover points cause more disruption and are important in the initial exploration of the search space. Later in the evolutionary cycle, a lower number of crossover points are more beneficial to exploit the current observations [2]. A reduced number of crossover points prevent the disruption of schemas with co-adapted alleles from forming.

Single point crossover was used in the chain code GA, where a single locus is chosen at random for the crossover point. The crossover rate will be .6 based on existing research [2]. This crossover rate will be compared with rates of .3 and .8.

Mutation

Mutation is intended to preserve genetic diversity against losing desired alleles during the course of the evolution. It is a powerful disruptor of schemas. The chain code GA uses a very low probability of .001, based on existing guidelines [2]. This will be contrasted with a higher probability of .01, also based on existing research [6].

Population size

The population must be sized so that an adequate number of building blocks exist to evolve into the desired schema. If a population is too small, a GA will not find a solution to the problem [7]. If it is too large, time is wasted processing unnecessary individuals and may be unacceptably slow. Population size is dependent on the number of genes and the quality of the fitness function [7]. Moreover, it interacts nonlinearly with crossover and mutation probabilities [11]. Population sizes of 50-100 are commonly used, based on published guidelines [2]. The proposed research will use a population size of 100. It will be compared with a population size of 200 and 30, the latter being based on results in [6].

7. Results

The following four chain codes were used in the experiment: a 6-sided hut, an 8-sided octagon, an airplane with 99 edges, and an odd shape with 18-edges.

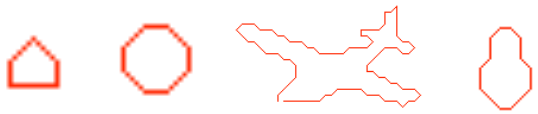

Figure 5: Chain Codes Used In Experiment

Eight tests were run with different GA parameters. Each test was run 12 times, 3 times for each chain code.

The transformations applied to the input start shape were held constant to eliminate the random effects that result because some transformations are not handled as well as others. For example, the transformation algorithm does not yet handle uneven scaling and rotation with 100% accuracy. Figure 6 shows an example of the odd shape chain code scaled up in the vertical direction and rotated 45 degrees, along with the two best attempts to reverse the scaling and rotation. The resulting fitness values of 3 and 2 are a weakness of the transformation algorithm rather than a failure of the GA. In cases where scaling was even in both dimensions, or where rotation did not occur, the transformation algorithm is able to reverse the transformation 100% of the time. .

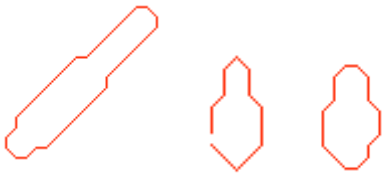

Figure 6: Uneven Scaling and Rotation of Odd Shape

The first start shape used in the experiments was scaled evenly in both directions and rotated 90 degrees. The second start shape was scaled in the horizontal direction and rotated 23 degrees. The third start shape was scaled in the vertical direction and rotated 45 degrees.

The results that are reported show the highest fitness value that was evolved and which run it occurred in. The best fitness value possible is 0, indicating no differences exist between the evolved chain code and the original model.

Test 1 is defined as the standard model. It is used as a basis of comparison with the remaining tests, which vary one component of the GA.

Test 1: Standard Model

This version of the GA converges quickly. Typically, the best solution is found in the first run, and the population is comprised of 100% of the highest fit individuals by the fourth run. This likely reflects the fact that the initial population contains individuals with a high fitness value. Since the algorithm considers five ranges of values for the scaling parameters, and eight ranges for the rotation parameter, there is a total of 200 variations of the chromosome. The initial population contains 100 random variations. The crossover function is quickly able to find the ideal chromosome with such a rich selection of building blocks.

Test 1 Configuration : Standard Model		
Fitness Function	Histogram	
Selection Method	Ranking	
Crossover	0.60	
Mutation	0.001	
Population Size	100	
Test 1 Results		
Shape	Best Fitness Evolved	Run Number Best Fitness Occurred
Hut - first start shape	0	1
second start shape	0	1
third start shape	1	1
Octagon	0	1
	0	1
	3	1
Airplane	0	1
	9	1
	18	2
Odd Shape	0	1
	2	1
	3	2

Test 2: Roulette Selection Method

This version of the GA used a roulette selection method. Results were not significantly different from the standard GA using the ranking method.

Test 2 Configuration: Roulette Selection		
Fitness Function	Histogram	
Selection Method	Roulette	
Crossover	0.60	
Mutation	0.001	
Population Size	100	
Test 2 Results:		
Shape	Best Fitness Evolved	Run Number Best Fitness Occurred
Hut	0	2
	0	1

	1	1
Octagon	0	1
	0	1
	3	1
Airplane	0	2
	9	1
	18	1
Odd Shape	0	1
	2	1
	3	1

Test 3: Hamming Fitness Function

This version of the GA used a deceptive fitness function, and was typically not able to evolve better solutions than those found in the initial population. There was one striking exception for the first start shape of the Odd Shape, where the GA evolved the most fit individual in run 32, which shows that even an inappropriate fitness function can sometimes produce desired results. Since this only occurred once, it was treated as an outlier and was not reflected in the reported results.

Test 3 Configuration: Fitness Function Varied	
Fitness Function	Hamming
Selection Method	Ranking
Crossover	0.60
Mutation	0.003
Population Size	100

Test 3 Results:		
	Best Fitness Evolved	Run Number Best Fitness Occurred
Hut	0	1
	0	1
	0	1
Octagon	0	1
	0	1
	2	1
Airplane	0	1
	88	1
	126	1
Odd Shape	0	1
	14	1
	5	1

Test 4: Crossover Rate Lower

This version of the GA lowered the crossover rate from .60 to .30. Results were not significantly different from the standard GA.

Test 4 Configuration: Crossover Rate .30	
Fitness Function	Histogram
Selection Method	Ranking
Crossover	0.30
Mutation	0.001
Population Size	100

Test 4 Results:		
Shape	Best Fitness Evolved	Run Number Best Fitness Occurred
Hut	0	2
	0	1
	1	1
Octagon	0	1
	0	1
	3	1
Airplane	0	1
	9	1
	18	1
Odd Shape	0	1
	2	1
	5	1

Test 5: Crossover Rate Higher

This version of the GA raised the crossover rate from .60 to .80. Results were not significantly different from the standard GA. Interestingly, the high dimensional airplane shape evolved its most fit value after run 50 in 2% of the tests.

Test 5 Configuration: Crossover Rate Higher	
Fitness Function	Histogram
Selection Method	Ranking
Crossover	0.80
Mutation	0.001
Population Size	100

Test 5 Results:		
Shape	Best Fitness Evolved	Run Number Best Fitness Occurred
Hut	0	1
	1	1
	1	1
Octagon	0	1
	0	1
	3	1
Airplane	0	1
	9	2
	18	2
Odd Shape	0	1
	2	1
	3	1

Test 5: Mutation Rate Higher

This version of the GA raised the mutation rate from .001 to .01. Results were not significantly different from the standard GA.

Test 6: Mutation Rate Varied Higher	
Fitness Function	Histogram
Selection Method	Ranking
Crossover	0.60
Mutation	0.01
Population Size	100

Test 6 Results:		
Shape	Best Fitness Evolved	Run Number Best Fitness Occurred
Hut	0	1
	0	1
	1	1
Octagon	0	1
	0	1
	3	1
Airplane	0	1
	9	2
	18	2
Odd Shape	0	1
	2	1
	3	1

Test 7: Population Size Higher

This version of the GA raised the population size from 100 to 200. Results were not significantly different from the standard GA. The run time was noticeably longer.

Test 7 Configuration: Population Size Higher	
Fitness Function	Histogram
Selection Method	Ranking
Crossover	0.60
Mutation	0.001
Population Size	200

Test 7 Results:		
Shape	Best Fitness Evolved	Run Number Best Fitness Occurred
Hut	0	1
	0	1
	1	1
Octagon	0	1
	0	1
	3	1
Airplane	0	1
	9	3
	18	1
Odd Shape	0	1
	2	1
	3	1

Test 8: Population Size Lower

This version of the GA lowered the population size from 100 to 30. Results were not significantly different from the standard GA most of the time. In approximately 5% of cases, the optimal solution would take 20-30 runs to evolve. This occurrence was erratic, as it did not correlate with the chain code or the transformation type. This behavior is not reflected in the report which is intended to capture the normal behavior. It likely reflects the lower number of highly fit building blocks in the initial population.

Test 8 Configuration: Population Size Lower	
Fitness Function	Histogram
Selection Method	Ranking
Crossover	0.60
Mutation	0.001
Population Size	30

Test 8 Results:		
Shape	Best Fitness Evolved	Run Number Best Fitness Occurred
Hut	0	1
	1	1
	2	1
Octagon	0	1
	0	1
	3	1
Airplane	0	1
	9	1
	18	2
Odd Shape	0	1
	2	1
	3	1

Comparison of Best Fitness Evolved							
Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
1	1	0	1	1	1	1	2
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
3	3	2	3	3	3	3	3
0	0	0	0	0	0	0	0
9	9	88	9	9	9	9	9
18	18	126	18	18	18	18	18

0	0	0	0	0	0	0	0
2	2	14	2	2	2	2	2
3	3	15	5	3	3	3	3

8. Conclusions

The GA was able to evolve the best solution or very close to it that the chain code transformation algorithm was capable of producing in every test that used the histogram fitness function. However, the GA converges very quickly. Typically, the highest fit individual occurs in the first run, and the population is comprised of 100% of the highest fit individuals by the fourth or fifth run.

Changing the GA parameters had no noticeable effect on the algorithms' performance. The results of all the tests using the histogram fitness function were similar to one another. Lowering the population size caused an occasional delay in evolving the best solution. This suggests that the number of highly fit individuals in the initial population is the cause of the quick convergence.

The inability to reach a fitness level of 0 reflects shortcomings in the accuracy of the chain code transformation algorithm when uneven scaling and rotation are combined. Future work on this algorithm will improve this functionality. Also, the chain code transformation algorithm was implemented with a limited degree of granularity. It should be possible to generalize the step functions to a continuous domain.

References

[1] Ahmad, M., Park, J., Chang, M., Shim, Y., Choi, T. Shape registration based on modified chain codes.

[2] De Jong, K.A., Spears, W.M. (1992). A formal analysis of multi-point crossover in genetic algorithms. Anals of Mathematics and Artificial Intelligence, 5(1), 1-26.

[3] Duda, R.O., Hart, P.E., Stork D.G. (2001) Pattern classification (2nd ed). New York: John Wiley & Sons, Inc.

[4] Escalera, A., Armingol, J.M., and Mata, M. (2003). Traffic sign recognition and analysis for intelligent vehicles. Image and Vision Computing. 21:247–258.

[5] Goldberg, D.E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In Foundations of Genetic Algorithms. San Mateo, CA: Morgan Kaufmann, 1991, 69–93.

[6] Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern. SMC-16, 1 (Jan./Feb. 1986), 122–128.

[7] Harik, G., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. Evolutionary Computation. 7, 3 (Sep. 1999), 231-253.

[8] Hsu, S. and Huang, C. (2001). Road sign detection and recognition using matching pursuit method. Image and Vision Computing, 19:119–129.

[9] Jain, A. K., Murty, M. N., and Flynn, P. J. 1999. Data clustering: a review. ACM Comput. Surv. 31, 3 (Sep. 1999), 264-323.

[10] Lu, G. Chain code-based shape representation and similarity measure.

[11] Mitchell, M. (1998). An introduction to genetic algorithms. Cambridge: MIT Press.

[12] Ozcan, E. and Mohan, C. K. (1997). Partial shape matching using genetic algorithms. Pattern Recognition Letters. 18, 10 (Oct. 1997), 987-992.

[13] Veltkamp, R. C. (2001). Shape Matching: Similarity Measures and Algorithms. In Proceedings of the international Conference on Shape Modeling & Applications (May 07 - 11, 2001). Shape Modeling International. IEEE Computer Society, Washington, DC, 188.

[14] Schaffer, J. D. Caruna, R. A. Eshelman, L. J. Das, R. 'A study of control parameters affecting online performance of genetic algorithms for function optimization', In Schaffer, pp 51-60, 1989.