Self-Adapting Vertices for Mask-Layout Synthesis

Cin-Young Lee and Erik K. Antonsson*

California Institute of Technology
*Mail Code:104-44, 1200 E. California Blvd
Pasadena, CA 91125, USA, erik@design.caltech.edu

ABSTRACT

An efficient procedure for synthesizing MEMS mask-layouts for a desired 3-D shape is discussed. This method can greatly reduce the number of design and prototype iterations required to produce a desired device. The method is based on evolutionary algorithms, where the locations of vertices in the polygonal mask-layout are optimized, such that the resulting shape is 'closest' to the desired shape [2], [3]. This work has been extended here to include varying the number of vertices in the mask-layout polygon(s), to free the designer from having to make an initial estimate at the complexity of the mask-layout required. Preliminary results are presented.

Keywords: mask-layout synthesis, bulk etching, evolutionary algorithms

1 INTRODUCTION

Mask-layout creation for MEMS design is often an iterative process that requires expert knowledge. This difficulty arises from the highly nonlinear mapping of mask-layouts to device shapes as a result of anisotropic etchants. Hence, a mask-layout synthesis tool could significantly reduce the number of design and prototype iterations, while also freeing the designer from having to deal with fabrication issues.

Such a tool has been developed using genetic algorithms, a type of evolutionary algorithm. The genetic algorithm approach allows for a highly robust search for the optimal mask-layout that generates a desired device shape [2], [3]. However, the initial implementation of this approach had one significant drawback; the number of polygon vertices in the mask-layout had to be specified. By restricting the number of vertices to a fixed value, the synthesis tool confined its search to a small subset of all possible mask-layouts, perhaps leading to the inadvertent neglect of novel and interesting mask-layouts. As a result of these limitations, a new approach was developed here, based on evolutionary strategies (another type of evolutionary algorithm), that is capable of producing mask-layouts of variable dimensionality. This approach then frees the MEMS designer from having to make an initial estimate of the complexity of the mask-layout, leading to fully automated mask-layout synthesis.

We follow this introduction by reviewing evolutionary algorithms. Subsequently, we discuss the details of our evolutionary strategy approach to mask-layout synthesis. Preliminary results are shown next, followed by a summary. The paper concludes with some notes on future work.

2 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EA's) are population based optimizers that crudely imitate biological evolution. The idea is to manipulate population members using stochastic, evolutionary operators (*i.e.*, crossover and mutation), which should, in practice, increase the average population 'fitness' with each new population, or generation. Most often, this is accomplished through a 'survival of the fittest' reproduction scheme. EA's are robust optimizers in that they are highly effective at finding global optima in any search space, including highly nonlinear search spaces. The robustness of EA's can be primarily attributed to the population-type search; since, in population based searches, solution diversity can be maintained, allowing the avoidance of local optima.

The two subclasses of EA's previously mentioned, genetic algorithms (GA's [1]) and evolutionary strategies (ES's [4]), both utilize a coding or parametrization of solution space to create strings/genomes that are more amenable to search using evolutionary operators. GA's and ES's mainly differ in their use of binary or real valued coding schemes, crossover or mutation as primary evolutionary operator, and fitness proportional or rank based selection methods.

3 EVOLUTIONARY STRATEGY IMPLEMENTATION

The ES implementation can be described as the following iterative procedure, after starting from a randomly initialized population: (1) generate offspring from the population using stochastic operators, (2) determine individual fitness values, (3) select best performing in-

dividuals to survive to next generation, (4) repeat steps 1–3 until termination criteria are met. Briefly, details required for replication of our ES implementation are discussed. In particular, the coding scheme, initialization procedure, stochastic operators, fitness evaluation, and selection and termination criteria are all elucidated.

3.1 Coding Scheme

The mask-layout, or 2-D shape/polygon, is encoded as two real valued strings using a polar representation. For example, a square with vertices with the (x, y) coordinates at the permutations of $(\pm\frac{1}{4}, \pm\frac{1}{4})$ has the following two string representation:

angle:	45	135	225	315
radius:	$\frac{1}{\sqrt{8}}$	$\frac{1}{\sqrt{8}}$	$\frac{1}{\sqrt{8}}$	$\frac{1}{\sqrt{8}}$

A third string, denoting the proportion of each edge's length to the total length, is also added to the encoded polygon representation. Distances are taken relative to a starting angle of zero. So, using the above example, the distance string would be (.125, .375, .625, .875). Furthermore, every shape is normalized to have a perimeter of length 100.

3.2 Initialization

For a population of size m, individuals are randomly generated according to the following procedure until the initial population is filled.

1. Randomly determine the size, or number of vertices, n, of the shape.

2. Choose the number of out-of-order vertices, n_o, randomly from a uniform distribution with range 0 to $n/3$.

3. For $i = 1$ to $n - n_o$: Randomly generate vertices by choosing an angle between 0 and 360 and a positive radius.

4. Sort the existing vertices in ascending angle order. The current polygon will then be either convex or star-shaped.

5. For $i = 1$ to n_o: Randomly pick a vertex, v. Obtain a new angle by adding a random Gaussian variable with zero mean to v's angle. Insert this new angle with a randomly generated radius after v.

6. Normalize perimeter to 100 by dividing each radius by the perimeter, then multiplying by 100. The distance, or identifying, string is now easily calculated.

The third step can create out-of-order vertices (*i.e.*, vertices not ordered by angle size). In addition, this step may create a self-intersecting polygon, leading to an invalid mask. Fortunately, these polygons typically have poor fitnesses and are weeded out during the selection process. Hence, no measures are taken to enforce non-self-intersecting polygons.

3.3 Stochastic Operators

Two stochastic operators, which are variations of the canonical operators mutation and crossover, are used. Note, however, that crossover is not normally used in evolutionary strategies, such that our implementation can be thought of as a hybrid GA/ES.

3.3.1 Mutation

For our application, the canonical mutation operator suffices. The canonical mutation operator simply modifies every value in the radius and angle strings by adding a Gaussian random variable with zero mean to each value. The variance of the random variable is also self-adapted over time. More detailed descriptions of the mutation operator can be found in [4].

3.3.2 Crossover

Crossover, as compared to mutation, is a recombination operator that swaps sections, or building blocks, between two or more parents. Here, a two parent crossover is implemented that allows the size, or number of vertices, of the offspring to vary from the parents' sizes. The distance string is taken as the identifying string; so, swaps occur depending on the range of distances chosen. For example, if the range [.2, .6] is selected, all distance genes, or values, that are in the range, along with the respective radii and angles, are swapped between parents. The importance of using the distance string is that it only lets compatible sections of polygons to be interchanged between parents. Note that after both crossover and mutation, each shape needs to be renormalized in order to maintain consistency of the coding scheme.

3.4 Fitness

The problem of mask-layout synthesis can be restated as a search in 2-D shape space for the mask that maps to the device shape most 'closely' resembling the desired device. So, the fitness function should quantify the similarity between three dimensional shapes. The approach taken here, though, is to create a fitness function that quantifies the similarity between 2-D shapes. A 3-D fitness function can be realized by slicing both target and test shapes at different depths and summing the 2-D fitnesses of the resulting contours.

The 2-D fitness function has the following structure. For each test shape vertex, the closest target shape vertex, l, is found. The distance between these vertices is added to the fitness value (FV). Penalties are also added to the FV if the closest target vertices are unordered or repeated. For example, say that the l's for the first three test shape vertices are the third, first, and first target shape vertices. Penalties would be added for the unordered and repeated target shape vertices. Thus, smaller FV's denote higher similarity between target and test shapes.

3.5 Selection and Termination Criteria

A $(\mu+\lambda)$ ES selection scheme is used, where μ is the number of individuals in the population and λ is the number of offspring. The '+' denotes that the new population members are selected from the combined pool of parents and offspring. More specifically, a $(20+80)$ selection scheme is implemented, where offspring are generated from parents through crossover and mutation. This selection scheme has the property that parents are never lost if the offspring all perform worse than the parents, a so-called 'elitist' property. Elitist selection then requires that the average population FV be a monotonically decreasing function of time, or generation number.

The ES terminates if the number of generations exceeds 500. Results show convergence beyond 500 iterations is poor. However, the ES is allowed to terminate prior to 500 generations if the population's average FV stays within 0.0001 of the previous generation's. In both cases, the highest performing individual is chosen as the optimal mask-layout.

4 RESULTS

As proof of concept, a 2-D shape matching experiment was completed. The problem can be stated as, given a desired target shape, find this shape in the space of all possible 2-D shapes. Results of the experiment are presented here.

Figures 1–3 show the target shape and best shapes at different iterations of the experiment. It is apparent that there is good convergence, as the final shape (shown at the right in Figure 3) is the same as the target shape (shown at the left in Figure 1). These shapes were obtained from an initial population whose polygon sizes ranged between 3 and 53 vertices. Similar convergence is also exhibited for other size ranges; although, extremely small ranges (*i.e.*, 3 to 6) and extremely large ranges (*i.e.*, 3 to greater than 500) show poor convergence to the target shape. The likeness of the target shape can often be distinguished in the best shape after 60-70 generations. Beyond this point, shape improvement is very slight from generation to generation, as the vertices are perturbed small distances in attempts to find 'closer' matches. These results are not unexpected, since the behavior of rapid convergence to the global optimum's neighborhood followed by gradual convergence to the optimum is ubiquitous in evolutionary algorithm implementations.

At present, the ES implementation has not yet been interfaced to a bulk-etching simulator. Hence, results of mask-layout synthesis are not presented here.

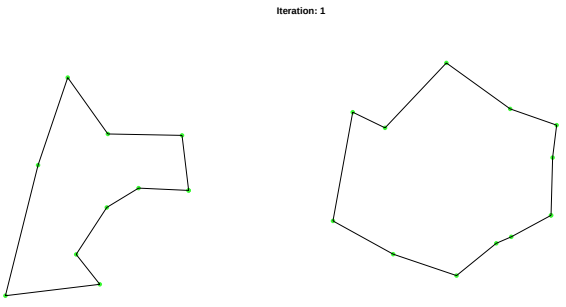

Figure 1: Target shape (left) and best shape at iteration 1 (right).

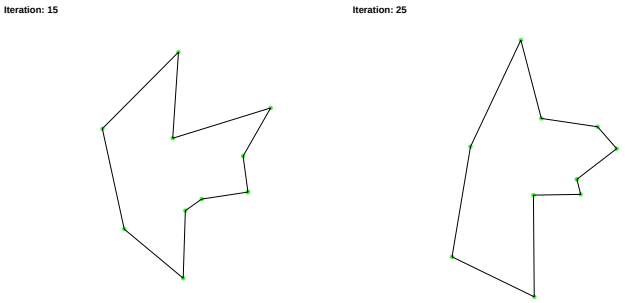

Figure 2: Best shapes at iterations 15 and 25.

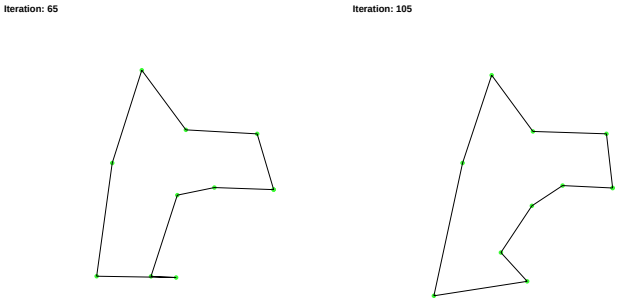

Figure 3: Best shapes at iterations 65 and 105.

5 CONCLUSION

An evolutionary strategy has been developed to automate the synthesis of mask-layouts with arbitrary number of vertices. Results of the 2-D shape matching problem show that the ES implementation is effective in searching 2-D shape space. Furthermore, since 2-D shape space is also the solution space of the mask-layout synthesis problem, we expect that the ES implementation, when interfaced with a bulk-etching simulator, will exhibit good performance in mask-layout synthesis.

6 FUTURE WORK

At the moment, the most pressing concern is to interface our ES implementation to a bulk-etching simulator. A mask-layout synthesis tool can then be developed and tested to determine its performance and worth. It is hoped that the approach presented here will be capable of finding novel layouts that are time consuming to obtain using the GA approach [2], [3].

Time reductions can also be made by improving the gradual convergence properties exhibited in the latter portions of ES runs. Perhaps, a greedy search can be used. A better idea, though, would be to restart the ES with individuals in the converged neighborhood. The reason for this idea lies in the fact that ES's, when converged to a neighborhood, act approximately as gradient searches. Consider, for example, an ES that has converged to a neighborhood such that all population members are nearly identical. It is apparent that crossover has little effect on the makeup of new generations. Mutation, on the other hand, has a great effect on the makeup of new generations, as mutation modifies an individual by moving it in a random direction. Thus, in selecting the best parents and offspring to survive, one is in fact selecting the best direction to move in. This is exactly the premise of gradient search. The step size of the search is the variance of the mutation parameter, which, as described previously, is self-adapted from generation to generation. As the ES converges, this step size decreases, such that only tiny steps can be made in the approximate gradient direction, leading to slow convergence to the global optimum. Hence, it would be wise to restart the ES in the vicinity of the global optimum while also resetting mutation variance.

REFERENCES

[1] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, Inc., New York, 1989.

[2] Li, H., and Antonsson, E. K. Evolutionary Techniques in MEMS Synthesis. In *ASME International Mechanical Engineering Congress and Exposition* (Anaheim, CA, Nov. 1998), ASME.

[3] Li, H., and Antonsson, E. K. Mask-Layout Synthesis Through an Evolutionary Algorithm. In *MSM'99, Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators* (San Juan, Puerto Rico, Apr. 1999), IEEE.

[4] Schwefel, H.-P. *Evolution and Optimum Seeking.* John Wiley, New York, 1995.