

EVOLUTIONARY SYNTHESIS OF MICROELECTROMECHANICAL SYSTEMS(MEMS) DESIGN

NINGNING ZHOU

Dept. of Mechanical Engineering
University of California at Berkeley
Berkeley, CA94720

BO ZHU

Dept. of Mechanical Engineering
University of California at Berkeley
Berkeley, CA94720

ALICE M. AGOGINO

Dept. of Mechanical Engineering
University of California at Berkeley
Berkeley, CA94720

KRISTOFER S.J. PISTER

Dept. of EECS
University of California at Berkeley
Berkeley, CA94720

ABSTRACT

This paper presents the preliminary research on the automatic synthesis of MEMS structures by using multi-objective genetic algorithms (MOGA). The problem model includes problem inputs, the cost function, the types and numbers of available components such as anchors and beams. The Multi-Objective Genetic Algorithm is applied to randomly generated populations to iteratively search for functional designs achieving the desired performance. Besides selection and crossover, elitism and immigration have been implemented in MOGA to produce designs for the next iteration. Initial results have been obtained for automatic synthesis of both topology and sizing of a MEMS 2D meandering spring structure with desired stiffness in certain directions. Preliminary results demonstrate the feasibility of this approach to MEMS structure synthesis.

INTRODUCTION

MEMS technology has been developed for decades to make miniaturized mechanical devices and components integrated with microelectronics on silicon chips or other substrates. Although the advancement is phenomenal, the development of new micro devices remains challenging because currently MEMS design still heavily relies on human knowledge and expertise in this area. Synthesis tools that could automate the MEMS design process have the potential to revolutionize MEMS devices in much the same way that integrated circuit (IC) synthesis tools transformed IC design. Previous work in MEMS synthesis includes the automated layout generation for commonly used device topologies (Mukherjee et al., 1998) (Lo et al., 1996) such as the folded-flexure microresonator, the fabrication sequences generation for surface micromachined structures starting from a two-dimensional geometrical description (Gogoi et al., 1994) and the mask layout generation and fabrication process synthesis for bulk micromachining (Li and Antonsson, 1998). All of these methods, however, are tailored to specific domains and tasks and are not general purpose MEMS synthesis tools.

We have developed an evolutionary method to synthesize functional MEMS devices by combining parameterized basic MEMS building components together. These building components include such primitive elements as anchors, beams and electrostatic gaps as well as higher level building blocks called subnets that are comprised of those primitive elements. Examples of subnets include a spring

composed of several beams or an electrostatic comb drive composed of a number of electrostatic gaps (Tang et al., 1989). This method incorporates a MEMS simulator SUGAR (Zhou et al., 1998) into a Multi-Objective Genetic Algorithm (MOGA) to automate the synthesis of MEMS designs. SUGAR is a MEMS simulation package developed at University of California at Berkeley (2000). Given a higher-level description of the device's desired behavior, both the topology and sizing of the devices are generated. The topology means the physical configuration that includes the number of gross building components, the types of components and their connectivity. The sizing of the designs entails assigning geometrical values to parameterized building components.

In this paper, we outline our approach and report some of our preliminary results. Our initial interest was to test the feasibility of using evolutionary algorithms to synthesize MEMS devices by combining the building components we defined. This work is illustrated by a 2D meandering flexure consisting of one anchor and several beams connected subsequently. Only the number of gross building components (number of beams) and their sizing are evolved.

PROBLEM STATEMENT

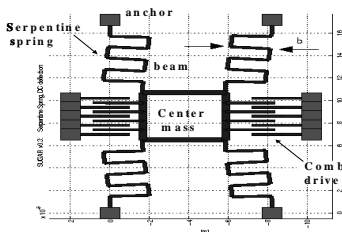


Figure 1: A MEMS resonator with four serpentine supporting springs

We start with a simple but fundamental MEMS supporting spring structure – meandering flexure. One example of it is the serpentine spring (Fedder, 1994) shown in Fig.1. A center mass is supported by four serpentine spring structures to form a MEMS resonator. Two sets of electrostatic comb drives on both sides drive this resonator vibrating horizontally. Our representation of the serpentine spring is shown in Fig. 2:

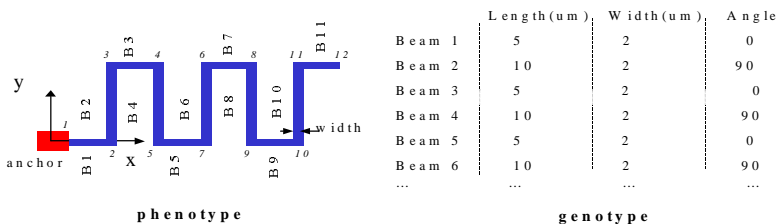


Figure 2: Coding scheme for meandering flexure

Beam 1 (B1) extends from anchor point node 1 to node 2 with length 5um, width 2um, at an angle of 0 with respect to the x axis; beam 2 (B2) extends from node 2 to node 3 with length 10um, width 2um, at an angle of 90 with respect to the x axis and so on. Each meandering flexure is encoded into a matrix of N by 3 as the genotype shown in Fig. 2, where N is the number of beams. Each row contains the length, width and angle of one beam. The matrix is ordered

so that the first row corresponds to beam 1 that is the nearest to the anchor, the second row corresponds to beam 2 that is the second nearest to the anchor and so on. With length, width and angle of each beam, we can fully describe the configuration and geometries of a meandering flexure. Our goal is to choose the right number of beams and the right length, width and angle parameters for each beam to achieve certain design objectives. For example, we could design a spring with specified stiffness along the x and y directions or design a spring with a certain ratio between the x stiffness and the y stiffness.

EVOLUTIONARY ALGORITHM

Genetic algorithms are global search algorithms based on the mechanics of natural selection and natural genetics (Goldberg, 1989). It applies selection, crossover and mutation etc. to the randomly generated population to iteratively search for optimal solutions. The traditional genetic algorithm and its many variations have been successfully applied in complex engineering design domains. To deal with multiple non-commensurable objectives, several multi-objective genetic algorithms (MOGA) adopt the concept of Pareto optimality (Goldberg, 1989) (Schaffer, 1985) (Tamaki, 1994). This paper applies MOGAs into MEMS design since most design cases involve more than one objective.

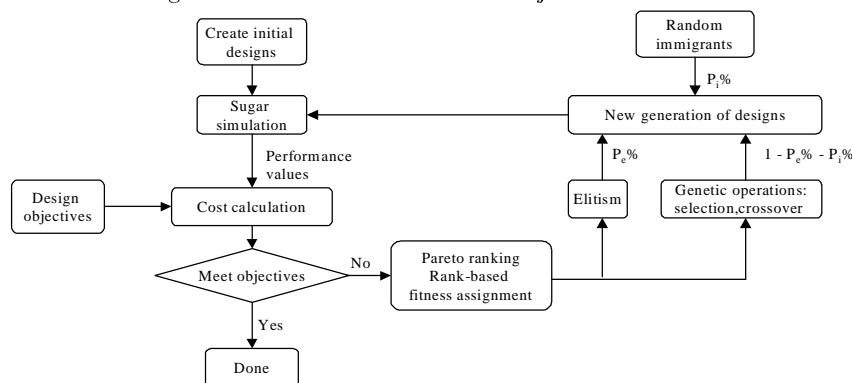


Figure 3: Flow chart for our evolutionary MEMS approach

Figure 3 illustrates an evolutionary algorithm iteration loop for MEMS design. Solutions have to be first encoded into a representation format that can be manipulated by genetic operations. An initial random population of designs is produced in this format. Each design in the current generation is checked for geometrical validity and its performance is evaluated by SUGAR. A cost vector is also calculated for each individual. If the performance doesn't meet the objectives, the whole population of the current generation is ranked using Pareto optimality. A fitness value is assigned to each design based on its rank. To increase the converging rate of the evolutionary algorithms, $P_e\%$ of the elites (individuals with rank 1) in the total population are directly copied to the new generation. As the complementary mechanism of mutation, randomly generated immigrants which are $P_i\%$ in the total population, are introduced into population to preserve diversity. The remaining $1 - P_e\% - P_i\%$ individuals are generated from genetic operations of selection/crossover. This MOGA loop iteratively searches for functional designs that meet constraints and optimize performance. The concept of Pareto optimality (Fonseca and Fleming, 1995) eliminates the use of pre-specified or explicit weightings and makes it possible to provide multiple optimal solutions to the decision maker. Pareto optimality also provide designers with a family

of 'equally best' or 'non-dominated' solutions therefore providing more design flexibility.

Rank-based Fitness Assignment and Fitness Sharing

A fitness value is assigned to every solution to measure its quality. In MOGA, the smaller the rank, the better the performance. We define the fitness to be the inverse of the rank.

Since the number of beams N is a design variable in this problem, designs with less beams have a smaller design space and thus converge faster. Designs with more beams have a larger design space with more free variables and may converge slower, although they may find better solutions. Given the same limited computing time and computing power (as we let them evolve in the same run), we found that the population tends to drift to the solutions with less beams because these solutions converge faster. This imbalance can be accumulated with the evolutions such that it is impossible eventually to find a better solution with more beams. We counteract this drift by using Fitness sharing to re-distribute the fitness among the candidate solutions with the same rank. In this paper, the sharing function is defined as

$$f'(K^j) = \frac{c^{N^j}}{\sum_j (c^{N^j})} \sum_j f(K^j) \quad c > 1 \quad (1)$$

where $f(K^j)$ is the fitness value of an arbitrary solution K^j within the same rank, N^j is the number of beams of K^j , and $f'(K^j)$ is the fitness after sharing. c is an experimental factor to adjust the population distribution of solutions with differing number of beams N^j . This function penalizes the fitness of individuals with less beams. The larger the c , the larger population of the individuals with more beams. c was chosen to be around 1.4 in this problem to balance the population distribution.

Crossover

After two parent designs are selected by Roulette wheel selection, crossover is carried out to generate two new child designs. The crossover scheme comprises three steps: parametric arithmetical crossover, cut and splice. Two parent designs with differing number of beams, are represented as two matrices: $N_1 \times 3$ and $N_2 \times 3$. Suppose $N_{min} = \min(N_1, N_2)$, as shown in Fig.4,

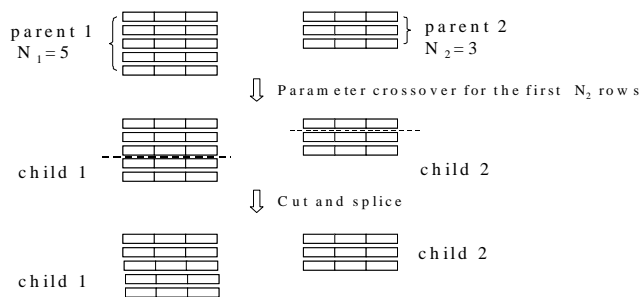


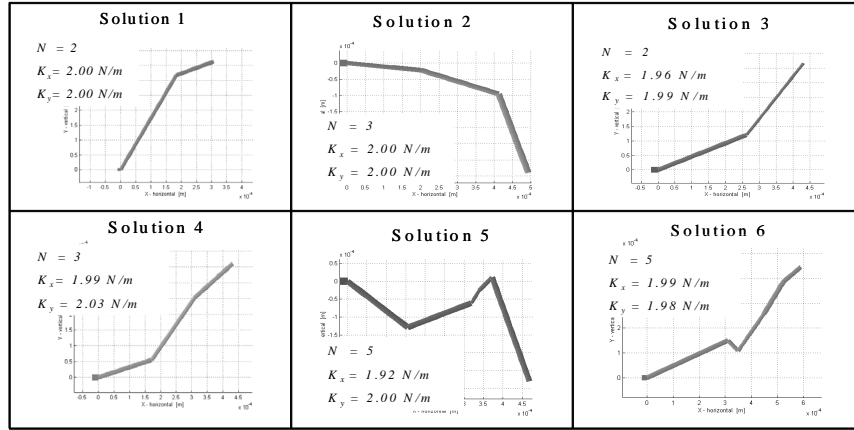
Figure 4: Crossover Scheme

Arithmetical crossover is carried out between the first N_{min} rows of two parents. It is separately applied to the pairs of length, width, and angle to ensure valid decoding. For example, length can't be swapped with angle.

Table 1: Design example configuration

Design variables	w_{\min}	w_{\max}	l_{\max}	θ_{\min}	θ_{\max}	N_{\max}
	2um	20um	400um	-90°	90°	6
MOGA Configuration	n_{pop}	n_{gen}	P_i	P_e	λ	
	200	40	40%	5%	0.3	

Table 2: Synthesis results for meandering springs from 2 MOGA runs



DESIGN EXAMPLES

We illustrate our approach on a design problem with a meandering spring composed of one anchor and N beams.

Design Objectives: stiffness in x direction $K_x = 2N/m$; stiffness in y direction $K_y = 2N/m$.

Design variables: number of beams N , length of beams l , width of beams w , angle of beams θ .

Design Constraints: $w_{\min} < w < w_{\max}$, $w < l < l_{\max}$, $\theta_{\min} < \theta < \theta_{\max}$, $1 \leq N \leq N_{\max}$. w , l , θ are real numbers, N is an integer.

The design parameters are shown in Table 1, where n_{pop} is the population size in each generation, n_{gen} is the number of generations to be run before stopping the MOGA program. We ran several tests with n_{pop} varying from 100 to 800 and n_{gen} varying from 20 to 200. n_{pop} and n_{gen} were chosen to be 200 and 40 respectively in the problem because increasing them beyond 200 and 40 did not improve the performance significantly. Table 2 shows six different synthesized designs with different number of beams. Solution 1 and 2 are from the Pareto set (with rank 1) of one MOGA run 1. They are the equally best solutions with a different number of beams $N = 2$ and 3. Solution 3, 4, 5, 6 are from the Pareto set of another MOGA run 2. They have a different number of beams $N = 2, 3$ and 5. All of these closely meet the design objectives.

In MOGA run 1, we applied the fitness sharing scheme with $c = 1.3$. The best designs with relatively small number of beams ($N = 2, 3$) were evolved. In MOGA run 2, we increased c to 1.4. It produced designs with 5 beams as well as 2 and 3 beams. Without fitness sharing, most of the MOGA runs generated solutions of 2 beams or sometimes even 1 beam. Fitness sharing plays an important role in diversifying the solutions.

CONCLUSION

Thus far we have completed the iterative design synthesis loop by combining the evolutionary algorithm and SUGAR simulation. We have also completed the initial testing stage by designing meandering springs, resulting in reasonable and promising results. During the initial testing stage, we found that it is not likely to generate complicated devices by only combining a small number of primitive building blocks. Current research is targeted towards extending this evolutionary algorithm approach to more complex device synthesis by incorporating more complex building blocks than beams and anchors, concurrent evolution of structure connectivity as well as parameters. Different encoding, random design initialization and crossover schemes are being developed. We are also exploring methods for adding expert design knowledge into the initialization process to make it less random and to reduce the search time. Our long term goal is to develop the evolutionary algorithmic approach into a simulation and synthesis CAD package for robust and efficient MEMS design.

REFERENCES

- Fedder, G., 1994, Simulations of Microelectromechanical Systems, Ph.D. Thesis, UC Berkeley, pp.105-111.
- Fonseca, C.M. and Fleming, P.J., 1995, Multiobjective Genetic Algorithm Made Easy: Selection, Sharing and Mating Restriction, *Genetic Algorithm in Engineering Systems: Innovations and Applications*, pp. 45-52.
- Goldberg, David E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.
- Gogoi, B., Yeun, R. and Mastrangelo, C., 1994, The Automatic Synthesis of Planar Fabrication Process Flows for Surface Micromachined Devices, *Proc.IE-EE Micro Electro Mechanical Systems Workshop*, pp 153-157, Oiso, Japan.
- Li, H. and Antonsson, E.K., 1998, Genetic Algorithms in MEMS Synthesis, *ASME International Mechanical Engineering Congress and Exposition, (MEMS)*, Anaheim, CA, pp.299-303.
- Lo, N.P., Berg, E.C., Quakelaar, S.R., Simon, J.N., Tachiki, M., Lee, H.J. and K.S.J. Pister, 1996, Parameterized Layout Synthesis, Extraction, and SPICE Simulation for MEMS, *Proc. ISCAS*, pp. 481-484, Atlanta, GA.
- Mukherjee, T., Iyer, S. and Fedder, G., 1998, Optimization-based Synthesis of Microresonators, *Sensors and Actuators, A* 70, pp.118-127.
- Schaffer, J.D., 1985, Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *Proceedings of the first International Conference on Genetic Algorithms*, pp. 93-100.
- Tamaki, H., Kita, H. and Kobayashi, S., 1996, Multi-Objective Optimization by Genetic Algorithm: A Review, *Proc. 1996 IEEE International Conference on Evolutionary Computation*, pp. 517-522, Nagoya, Japan.
- Tang, W. C., Nguyen, T.-C H., Judy, M.W. and Howe, R.T., 1989, Electrostatic -comb drive of lateral polysilicon resonators, *Proc. of 5th International Conference on Solid-State Sensors and Actuators*, Montreux, Switzerland, pp. 328-331.
- University of California at Berkeley, SUGAR, <http://www-bsac.EECS.Berkeley.EDU/~cfm/>, 2000.
- Zhou, N., Clark, J.V. and Pister, K.S.J., 1998, Nodal Simulation for MEMS Design Using SUGAR v0.5, *1998 International Conference on Modeling and Simulation of Microsystems Semiconductors, Sensors and Actuators*, pp. 308-313, Santa Clara, CA.