

# Interactive Evolutionary CAD System for MEMS Layout Synthesis

Raffi Kamalian, Alice M. Agogino, and Hideyuki Takagi, *Member, IEEE*

**Abstract**— We propose an interactive layout synthesis CAD tool. It allows a human user to graphically manipulate and simulate a layout, but also has the ability to apply two simple evolutionary synthesis methods: a simple random walk function and simulated annealing. These methods are capable of optimizing very complex functions difficult for a human to tune by hand. Additionally the tool incorporates the ability to reduce the dimension of the search space via variable locking, contributing to both focused search and faster convergence to a desired performance. The CAD tool has been written with flexibility in mind, allowing the application to a wide range of layout synthesis problems. In this paper its effectiveness is demonstrated on a MEMS Vibratory Rate Gyroscope example.

## I. INTRODUCTION

Design Synthesis is the process of finding or generating the design that will supply a desired performance [1]. Advanced synthesis techniques have been developed to assist engineers in this task by automatically generating optimal solutions to complex problems. These techniques are particularly useful for problems containing many variables, constraints and competing objectives. One such application is the design of the layout for Microelectromechanical Systems (MEMS) devices, such as inertial sensors, optical communications and RF filters.

Much of the work in the area of applying advanced synthesis and optimization techniques to the area of layout generation has been focused on the use of genetic algorithms (GA). GAs, a class of stochastic algorithm that mimic the evolutionary process found in the biological world, evolve a population of designs towards one or more fitness goals over many generations. In academic research this approach has been shown to be effective in layout generation for such fields as VLSI circuit design [2], architectural interior design [3] and MEMS [4],[5].

A key limitation to the traditional GA approach is that the program's interaction with the engineer during the design evolution is very limited after the problem formulation (objective functions, constraints, genetic encoding, etc.). In our experience we have found this to be a significant barrier for adoption by designers in industry.

Manuscript received July 1st, 2006. This work was supported in part through US NSF grant CCR-DES/CC-0306557, the JSPS Postdoctoral Fellowship program and Grant-in-Aid for Scientific Research.

Raffi Kamalian is a JSPS post-doctoral fellow at Kyushu University, Fukuoka 815-8540, JAPAN (e-mail: raffi@design.kyushu-u.ac.jp).

Alice M. Agogino is on the faculty of the Mechanical Engineering Department at the University of California, Berkeley, CA 94720, USA (e-mail: agogino@berkeley.edu).

Hideyuki Takagi is with the Faculty of Design, at Kyushu University, Fukuoka 815-8540, JAPAN (e-mail: takagi@design.kyushu-u.ac.jp).

A second, related limitation is the 'one-off' nature of many of the design optimization efforts to date. The programming of the objectives, constraints and most importantly the data structure used to encode the parameters that describe the geometry of a design (known as a *genotype* in GAs) are generally specific to each problem and must be recreated for each. Progress in the field of creating flexible generic data structures for synthesis tools has been made [6], but this is an area that needs further attention for a successful layout design tool.

Based on these two issues, we propose a simple design tool that will give a designer a wider range of interaction during the layout design process than previously available. This tool is made to be as flexible as possible, allowing minimal effort for application to a wide range of problems.

Our proposed solution, simply dubbed the Interactive Layout Synthesis (ILS) tool, is an interactive computer aided design (CAD) tool using a simple graphical user interface (GUI) to interface between a graphical representation of a design and the parameters required to describe it. This geometry can then be interfaced with a simulation package to provide performance predictions on a number of objective functions.

The key feature that sets this tool apart from current CAD or simulation packages is the inclusion of automated synthesis functionality. Rather than the manual design manipulation of a conventional CAD modeler or simulator package, the proposed tool allows the user to also call upon evolutionary synthesis algorithms as desired to help their design reach their design goals. This functionality is expanded further by allowing the user to reduce the number of dimensions to be searched by the synthesis algorithm, thus converging faster towards optimal designs. We will provide a concrete example in the domain of MEMS design, but this approach is extendable to other forms of layout design (e.g. VLSI, architecture, etc) or the design of any object that can be both visualized and simulated on a computer.

In this paper we will first give background information on automated synthesis and the previous use of evolutionary interaction for layout design. We then give details on the proposed ILS tool, its synthesis capabilities and how its flexible interface is implemented. The application to the MEMS layout design of a vibratory rate gyroscope is presented to demonstrate the capabilities of this approach in a real world example. Finally further improvements and application areas are discussed.

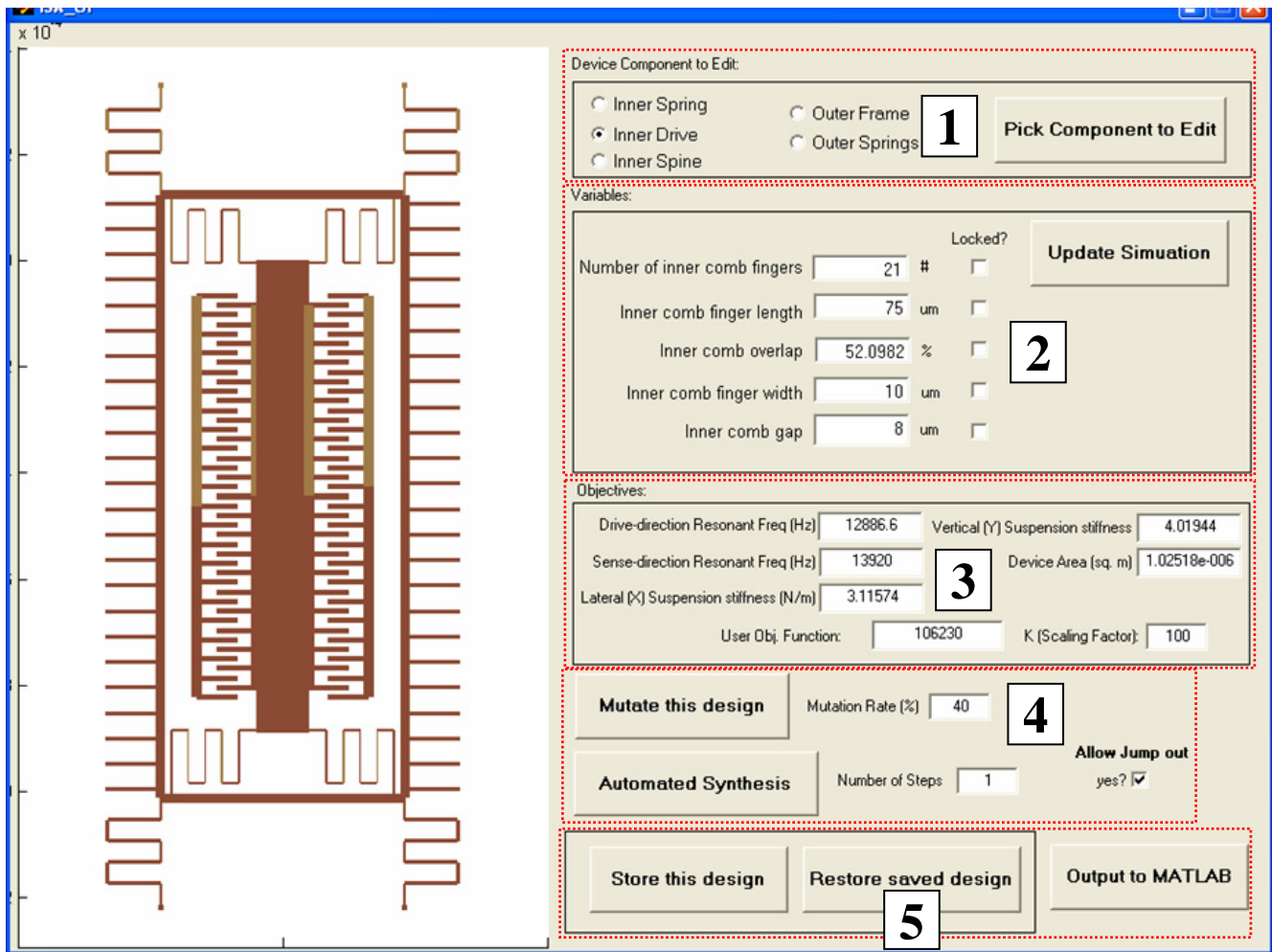


Fig. 1. User interface Interactive Layout Synthesis tool. The numbered regions correspond to the functionality described in section III.

## II. BACKGROUND

### A. Interactive Evolutionary Synthesis

Conventional evolutionary algorithms use a simulation engine or mathematical function to obtain the fitness of a given design automatically [7]. An alternate approach, known as Interactive Evolutionary Computation (IEC), uses a human user to evaluate the fitness of a design [8]. This approach has primarily been used for aesthetic or sensory-perception related design, but has also been adapted to engineering design [9]. This allows the human designer a limited amount of interaction with the evolution process. From observations of user studies, the greatest impact of this approach is giving the human the ability to add/remove/shift their own constraints during the process of evolution.

While this added interactivity is beneficial, there is still a barrier between the human designer and the design. They can only indirectly affect the design via their fitness evaluation (which impacts the likelihood the design passes its traits to the next generation), they can not specifically edit a portion of the design they may dislike. Likewise they have no ability to

freeze portions of the design they are satisfied with and do not wish to see changed in subsequent evolution.

In their Interactive Evolutionary Design System (IEDS), Parmee et al. [10] present an alternate approach to IEC, where the user interacts with the controlling parameters of an evolutionary algorithm rather than providing the evaluation of a fixed algorithm. IEDS therefore can contribute to the overall understanding of the search space and the relative importance of various objectives and constraints. In the same vein is the concept of computational steering, a method where a human can interactively control simulations during the course of the computation rather than only before or after [11]. This has been applied to interactive visualization during GA search [12].

With these concepts in mind, we propose a similar method for layout synthesis. To maintain simplicity and flexibility, we will focus only on the simplest forms of evolutionary synthesis, but give the user not only the capability to steer the automated evolutionary synthesis, but also to manually synthesize the design as with a traditional CAD/simulation system. In addition to providing better understanding of the

problem, this type of hands-on control of design synthesis is also attractive to designers who may prefer hands-on participation in the evolution, rather than only programming a GA and waiting to inspect the results upon completion.

### B. Evolutionary Search Algorithms

Perhaps the simplest notion of a stochastic search algorithm is the naïve search, also known as the *random walk* – a search technique where a candidate design is randomly perturbed to produce a new version. If that perturbation causes the design to perform better (higher fitness), the new version is kept, otherwise it is rejected and the previous design is kept. This process can continue until either a viable solution is reached, a certain number of perturbations have occurred and/or no improvements can be found after a certain number of perturbations.

Perturbations can be of fixed or random size. The type of perturbations allowed and the number of parameters affected by each perturbation can be adjusted to suit a particular problem. A key limitation of this random walk approach is that it is only suited for global optimization if the solution space is convex, e.g. – there are no local minima, as this method has no way to prevent getting stuck in the first local minima encountered.

A second approach, known as *simulated annealing* (SA) is similar except it incorporates a mechanism to jump out of local minima. SA is similar to a random walk approach, except it is willing to take a ‘worse’ solution (lower fitness) with a certain probability. As the number of perturbations increases or the proposed solution gets closer to the desired goal, the probability of accepting a lower fitness solution decreases [13].

Simulated annealing exploits an analogy between the way in which a heated metal cools into a minimum energy state and a stochastic optimization algorithm that slowly “lowers the temperature” in stages to eventually “freeze” at the global optimum.

In addition to their simplicity, we have chosen to use these approaches because they both evolve a single design at a time, attempting to minimize a single objective function. This is in contrast to genetic algorithms, which evolve a large population simultaneously towards one or more objectives. In the next section we will provide the details of the implementation of these two methods.

One criticism of GAs are their inability to focus the evolution on specific parts of a design while leaving the rest of the design fixed. One solution to this is the concept of variable locking. This has been demonstrated by a simple photomontage system [14], which uses IEC to evolve a picture of a person’s face for tasks such as identifying criminals. The locking capability allows the user to fix facial features that they feel are a good match and continue evolving the rest of the face. By reducing the number of variables involved, this allows the evolutionary algorithm to more quickly converge on the best solution.

## III. IMPLEMENTATION

In this section we will review the features currently available in the tool, the software architecture and the implementation of a generic data structure that can be applied to many different layout design problems. The interactive synthesis was produced in MATLAB using the GUI Development Environment (GUIDE) feature. This allows the simple creation and modification of platform independent GUIs and provides the functionality of the MATLAB environment.

### A. Features

A simple but important feature of this program is the fact that it provides a graphical link between the geometric parameters of a design, its graphical representation and its performance when simulated. In many simulation environments, a pre-generated CAD file is input and then simulated, any alterations must be made externally. This program allows a user to click on the desired region of the layout to edit (or manually select it from a list in region 1 of Fig. 1) and manually adjust whichever geometric parameters they desire (region 2 in Fig. 1). At any point they can then perform a simulation to find the performance of the modified geometry. The results of the simulation are presented to the user (region 3 in Fig. 1). These performance objectives are also combined in a user-defined objective function to provide an overall quantitative value of the current design, this value is used by the synthesis program, which seeks to minimize it.

In addition to the manual manipulation of individual dimensions, controls for automated synthesis of the design are also available (see region 4 in Fig. 1). This user can chose either a single step mutation or a multi-step synthesis. The rate of mutation can be input (the probability of any single variable to be changed) as well as the number of steps in a multi-step automated synthesis. The user can chose between the random walk algorithm and SA by checking the box labeled ‘Allow Jump Out.’ If the user chooses to freeze certain variables (via check boxes next to each variable in region 2 in Fig. 1), this information is also passed along, fixing portions of the design while allowing for the stochastic evolution of others.

Finally, the tool also allows the user to store a geometry at any point and recall it at a later time (region 5 in Fig. 1). This frees the user to explore radical changes in the geometry or aggressive mutation rates without fear of losing a good design. When a final design is reached, this can be output to the MATLAB workspace (for further editing, simulation, output to a CAD file, etc).

### B. Data Structure

In order to maintain generalizability, a generic structured class, a container holding multiple arrays, is used to transfer the geometrical data and problem specific information between the ILS tool and the helper functions.

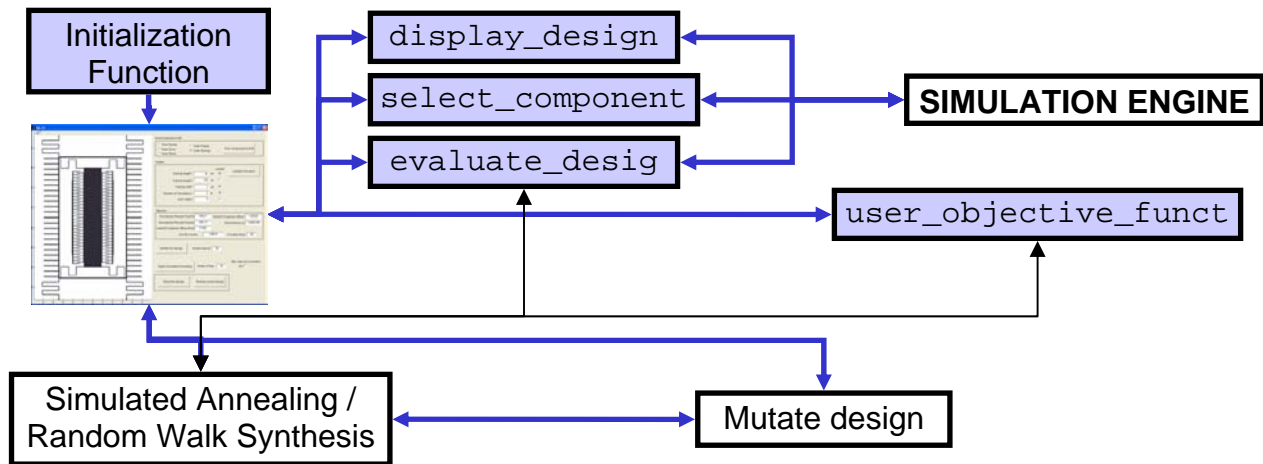


Fig. 2. Block diagram of ILS tool, showing the interaction between the GUI and the helper functions. Arrows represent information passed during function calls. Shaded functions must be modified for specific problems, unshaded are generic.

Table 1. Structured class data structure used by ILS tool.

Name	Description
geo	$m \times n$ array of geometric values
var_mask	$m \times n$ array of 1/0 values, identifying which locations in geo contain geometric parameters
locked_vars	$m \times n$ array of 1/0 values, identifying which variables in geo are locked to modification by the synthesis algorithm
bounds	a class containing two $m \times n$ arrays, corresponding to the min and max values allowed in geo
objectives	$o$ -dimension vector of performance numbers returned by the simulator
labels	a class containing arrays with the names and units for all geometry and objective values

Based on our experience in automated layout synthesis in the MATLAB environment, we have found matrices to be well suited for the description of the geometry. If a system is comprised of  $m$  components, each defined by up to  $n$  variables, an  $m \times n$  matrix can be used to store all the variables for all the components, one for each row. In rows where that component is described by less than  $n$  variables, zeros can be padded. As long as the order of the components and their variables in the rows and columns is used consistently throughout the ILS tool and its helper functions, this data structure is sufficient for our needs. This array format can also be used for the remaining class fields (see Table 1)

See section IV for an example of this data structure in a MEMS layout application.

### C. Architecture

There is a trade off between making a program easy to use and making it flexible. Previous attempts at synthesis tools were all ‘one-offs’, requiring extensive reprogramming for

each new design problem. To reduce this, we designed the system architecture (along with the data structure mentioned in the previous section) to minimize the number of problem-specific functions that must be generated to use this tool on a new problem.

Fig. 2 shows a block diagram of the interaction between the ILS and the external functions. The shaded blocks correspond to functions which must be modified for each application, but the main GUI window and the unshaded blocks are universal. A key issue in the user defined functions is interfacing between the MATLAB environment and the simulation engine, which can be either within MATLAB or an external program, accessible via API.

In the following paragraphs we describe each external function:

The initialization function generates the proper data structure for the given problem. An initial design can either be randomly instantiated or loaded from the workspace or a file.

The display\_design function receives the geometry and produces a graphical representation to the display. Likewise the evaluate\_design function calls the simulation engine the objective values and displays those in the appropriate boxes of the ILS display.

The select\_component function allows for the graphical selection of a component. This function works within the MATLAB graphical environment to get the X and Y coordinates of a point on the plot clicked on by the user. The function can then calculate what region of the design the user clicked on. This information can be calculated completely based on the information within the geometry matrix or in conjunction with the simulation engine to find the coordinates of the various components in the design.

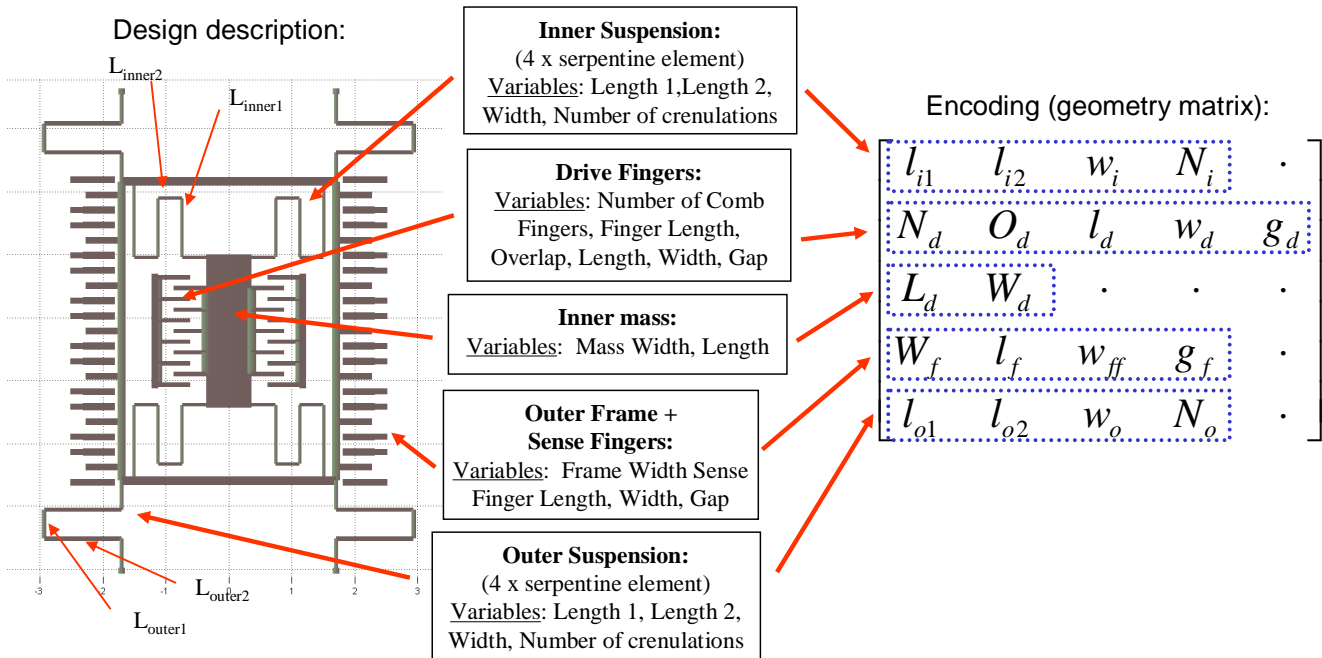


Fig. 3. Example of the encoding of a MEMS gyroscope design geometry in the ILS data structure.

The `user_objective_function` is the user created function that combines the performance objectives into a single objective value. By allowing the user to create their own function, this allows for much more flexibility than just using a simple weighed sum. One beneficial aspect of the use of helper functions for a GUIDE program is that functions can be edited online. While the ILS program is running, the user can change combination of objectives in their objective function.

Lastly, the fixed functions shown in Fig. 2 pertain to synthesis. A single step mutate (perturb) function that changes the unlocked variables of a design at the user defined mutation rate and a function to do multistep synthesis using the random walk or SA algorithms presented in section II. The synthesis function also directly utilizes the `evaluate_design` function and the `user_objective_function` to get the fitness value of a mutated design. This information is used to decide whether to accept or reject the altered version.

#### IV. TEST APPLICATION

This method and the fixed functions within it are compatible with a wide range of layout synthesis problems, such as VLSI design, architectural design, etc. We have chosen a MEMS design problem to illustrate its implementation.

A vibratory rate gyroscope (VRG) is a type of MEMS inertial sensor, capable of measuring rotation rate. These sensors are finding their way into many systems including the Segway scooter, car stability systems and autonomous vehicles.

To simulate the performance and also generate the graphical representation of the device, we use an open, freeware MEMS simulation tool known as SUGAR [15].

SUGAR uses modified nodal analysis, making it ideal for iterative design synthesis, where finite element modeling may be too slow to be practical. Also it can create graphics directly in the MATLAB graphics format, avoiding the need for writing bitmaps to the disc in an external program and then reading them to the display.

A Clark style VRG [16] contains over a dozen variables that must be chosen. Fig. 3 presents the components that comprise the design and the representation of those geometric parameters in the ILS data structure. Once this structure is set up, a function to read the parameters and convert it to a SUGAR input file is written. Now the design can be displayed graphically and its performance values can be simulated.

In this case objectives of interest include: resonant frequency in the drive direction, resonant frequency in the sense direction, drive direction suspension stiffness, sense direction suspension stiffness and device area. These five values can be fed back to ILS tool via the `'objectives'` field of the data structure.

For optimum performance, the drive and sense resonant frequencies of the device should match as close as possible to a goal frequency. In order to achieve this, a possible user objective function,  $F(x)$  to minimize is the following:

$$F(x) = \left| \omega_{r-sense} - \omega_{goal} \right| + k \left| \omega_{r-sense} - \omega_{r-drive} \right|$$

where  $\omega_{r-drive}$ ,  $\omega_{r-sense}$  are the two resonant frequencies and  $\omega_{goal}$  is the frequency goal.  $k$  is a scaling factor, chosen to balance the relative importance of frequency difference to the matching frequency goal, it can be set in the ILS window.

It is difficult to quantify the performance of this tool, or compare it to other approaches, as use patterns vary from user to user. In our testing, we found some users may prefer very

active control at the beginning, setting as many variables as possible to desired values and then running an automated synthesis on a reduce search space to hone the design. At the other end of the spectrum some may prefer to let the automated synthesis tool run in groups of steps to get close to the goal performance and then make manual adjustment of key variables (usually one at a time) to nudge the performance in the right direction. Another pattern is the active adjustment of the objective function in the course of an evolution (either simply adjusting the  $k$  scaling factor or changing the objective function equation directly).

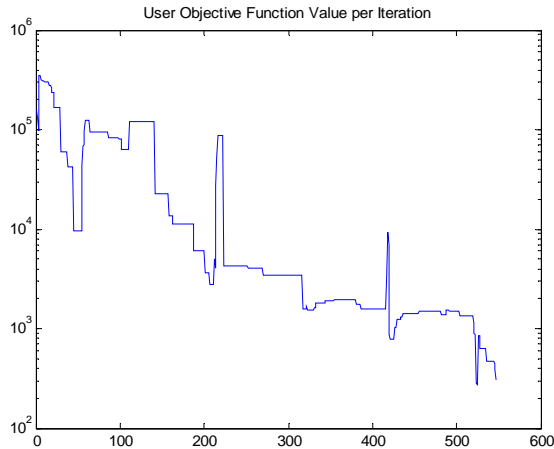


Fig. 4: Two example outputs of user objective function per iteration. Two different types of user behavior are presented.

Fig. 4 displays a plot of the user objective function at each step in an example optimization. Several random walk synthesis are performed, each 50 steps each (with varying mutation rates). Between sets of 50 steps, the user made a number of manual adjustments to the dimensions, which in some cases caused the fitness value to jump upwards. They also adjusted the relative weighting between the  $(\omega_{r-sense} - \omega_{goal})$  factor and the  $(\omega_{r-sense} - \omega_{r-drive})$  factor in the objective equation. In addition to shifting the weighting of the objective function and the geometry, another user action is changing which variables are locked and which are unlocked. The usage pattern in Fig. 4 shows overall each group of manual adjustments is capable of leading to better designs using automated synthesis. By approximately 550 steps, the design produced by this synthesis run had resonant frequencies of 10136Hz and 10309Hz, within the accuracy range of the SUGAR simulation tool.

## V. CONCLUSION

In this paper we have proposed an interactive layout synthesis tool that allows a human user the ability to interactively synthesize a design through a combination of manual or automated steps. The automated algorithms offered are two simple stochastic algorithms – the random walk and simulated annealing – as well as a single step

mutation to randomly change a design.

We present details on the implementation of this program, with an emphasis on a generic data structure that can be applied to as wide a range of design problems as possible. As an example a MEMS gyroscope problem is presented, and the results of two synthesis runs are presented. We hope to introduce this “easy to use” tool to industry as a practical method for layout design with the potential to add more complex algorithms in other facets of their MEMS design work.

## REFERENCES

- [1] E. K. Antonsson, J. Cagan, eds., *Formal Engineering Design Synthesis*, Cambridge University Press, Cambridge, 2001
- [2] P. Mazumder, E. Rudnick, *Genetic Algorithms for VLSI Design, Layout & Test Automation*, Prentice Hall, 1998.
- [3] A.M. Brintrup, H. Takagi, J. Ramsden, “Evaluation of Sequential, Multi-objective, and Parallel Interactive Genetic Algorithms for Multi-objective Floor Plan Optimisation”, *Proceedings of the Evolutionary Computation in Combinatorial Optimization, 6th European Conference*, 2006.
- [4] H. Li, E.K. Antonsson, “Evolutionary Techniques in MEMS Synthesis” *Proc. DETC’98, 1998 ASME Design Engineering Technical Conferences*, Atlanta, GA 1998.
- [5] N. Zhou, B. Zhu, A.M. Agogino, K.S.J.P. Pister, “Evolutionary Synthesis of MEMS MicroElectronicMechanical Systems Design,” *Proc. of the Artificial Neural Networks in Engineering (ANNIE2001)*, 2001, pp.197-202
- [6] Y. Zhang, R. Kamalian, A.M. Agogino, C.H. Sequin, “Hierarchical MEMS Synthesis and Optimization,” *SPIE Conference on Smart Structures and Materials*, March 7-10, 2005, San Diego CA, 5763-12.
- [7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman, Boston, MA, 1989.
- [8] H. Takagi, “Interactive Evolutionary Computation: Fusion of the Capacities of EC Optimization and Human Evaluation”, *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275-1296, 2001.
- [9] R. Kamalian, H. Takagi, and A.M. Agogino, “Optimized Design of MEMS by Evolutionary Multi-objective Optimization with Interactive Evolutionary Computation”, *Proceedings of GECCO 2004, Genetic and Evolutionary Computation Conference*, pp. 1030-1041, 2004.
- [10] I.C. Parmee, D. Cvetkovic, A.H. Watson, C.R. Bonham, “Multi-Objective Satisfaction within an Interactive Evolutionary Design Environment,” *Evolutionary Computation*, vol. 8, pp. 197–222, 2000.
- [11] C.R. Johnson, S.G. Parker, C. Hansen, G.L. Kindlmann, Y. Livnat. “Interactive Simulation and Visualization,” *IEEE Computer*, Vol. 32, No. 12, pp. 59-65. Dec, 1999.
- [12] A. Shenfield, M. Alkarouri, P.J. Fleming, “Computational Steering of a Multi-Objective Genetic Algorithm using a PDA”, Dept. of Automatic Control and Systems Engineering, University of Sheffield, UK, pp. 878
- [13] P. J. M. van Laarhoven, E.H.L. Aarts, *Simulated Annealing: Theory and Applications*, Reidel Publishing Company, Dordrecht, Holland, 1987.
- [14] H. Takagi, K. Kishi, “On-line Knowledge Embedding for Interactive EC-based Montage System,” *Third Int. Conf. on Knowledge-Based Intelligent Information Engineering Systems (KES’99)*, Adelaide, Australia, pp.280-283, 1999.
- [15] SUGAR, Simulation Research for MEMS, <http://www-bsac.eecs.berkeley.edu/cadtools/sugar/sugar/>
- [16] W. A. Clark, R.T. Howe, R.T. Horowitz, “Surface Micromachined Z-axis Vibratory Rate Gyroscope,” *7th Solid-State Sensor and Actuator Workshop*, Hilton Head Island, S. C., June 1996, pp. 299-302.