# FUZZY/MULTIOBJECTIVE GENETIC SYSTEMS FOR INTELLIGENT SYSTEMS DESIGN TOOLS AND COMPONENTS

**Michael A. LEE and Henrik ESBENSEN***
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720 USA
leem@cs.berkeley.edu
fax: 510 642 5775

*Avanti Corporation
1208 East Arques
San Jose, CA 94086
henrik_esbensen@avanticorp.com
fax: 408 738 8508

**Abstract.** In this chapter, we develop robust and efficient fuzzy/genetic based design tools and components for intelligent systems. The hybrid techniques exploit the knowledge representation capabilities of fuzzy systems and the adaptive capabilities of genetic algorithms. The core of the techniques presented in this chapter is a multiobjective variation of genetic algorithms. We first demonstrate how the multiobjective genetic algorithm can be applied to fuzzy system design and then proposed techniques to enhance the genetic algorithm technique using fuzzy systems. The fuzzy genetic system techniques proposed in this chapter provide intelligent systems designers with approaches to perform structural and parameter identification simultaneously of fuzzy systems and to perform efficient genetic algorithms based multiobjective optimization.

## INTRODUCTION

Key ingredients for an intelligent systems are knowledge representation structures and methods to adapt these structures. In this chapter, we will present techniques aimed at providing tools and methods for obtaining these key ingredients. The chapter is divided conceptually into two sections: genetic approaches to adapting fuzzy knowledge structures and fuzzy knowledge structures for adapting genetic search techniques. The core of both techniques is a multiobjective optimization variation of a genetic algorithm.

---

* This work was performed while the author was a Visiting Research Fellow in the EECS Department at U.C. Berkeley.

Multiobjective optimization has applications in many real-world contexts because humans are forced to choose among solutions that vary in performance with respect to multiple competing objectives. For example, in the case of integrated circuit (IC) layout generation, geometrical and timing specifications interact and performance along each of these directions may need to be compromised to satisfy specifications [4]. In problems of this nature often more than one solution exists, each of which must be regarded as equal in the absence of information relating the relative importance between the competing objectives. The final solution chosen is ultimately based on the specifications of the application or subjective (i.e. marketing) criteria, which can be difficult to define.

The first step in multiobjective optimization is not to provide the users with a set of solutions that represent the set of best alternatives, or *Pareto Optimal* set, rather than a single solution. The second step in multiobjective optimization is to choose a final decision based on the generated solutions.

The notion of Pareto optimality is based on the concept of dominance. Let $f(x) = (f_1(x), \ldots, f_n(x))$ represent a vector valued objective function and $u$ and $v$ represent two solutions. $u$ dominates $v$, written $u <_d v$, if and only if $(\forall i\colon f_i(u) \le f_i(v)) \wedge (\exists i\colon f_i(u) < f_i(v))$. Solutions included in the *Pareto Optimal Set* are those that cannot be improved along any dimension without simultaneously being deteriorated along other dimension(s). In applications with linear constraints and linear objectives, the literature on generating solutions in the Pareto set is quite mature[6]. However, in nonlinear contexts, determining solutions in the Pareto set requires different tools.

In the first half of this chapter, we present an evolutionary algorithm approach to generating solutions in the Pareto set. This technique is demonstrated on the problem of fuzzy systems design, where structure and parameter identification are performed simultaneously while considering both performance and system complexity. Results from this example show how producing a final set of solutions can provide a better understanding of the trade-offs before in deploying a particular system.

In the latter half of this chapter we concentrate on the behavior of the adaptation algorithm, which does not always act as we expect or in an intuitively efficient manner. We propose using a dynamic control framework to modify the behavior of these algorithms. In particular, we will present techniques to control multiobjective evolutionary algorithms using a fuzzy systems approach.

## MULTIOBJECTIVE EVOLUTIONARY ALGORITHMS

Genetic algorithms are population based stochastic search strategies modeled after mechanisms in natural genetics. The search is implicitly directed based on collective information represented by the population. An individual of the populations encodes

a solution as a string of parameters, which is then subjected to mutation and crossover operations. The probability that an individual passes information to the next generation is determined by its ability to thrive in the environment: genetic algorithms implement a *survival of the fittest* policy.

A simple genetic algorithm proceeds in the following manner:

1. generate initial population randomly or through seeding with *a priori* knowledge.
2. evaluate the fitness of all individuals in the population.
3. select parents to mate and generate the next population
4. apply crossover and mutation operators and generate next populations.
5. if the stopping criteria is not met, goto step 2, else terminate.

Simple genetic algorithms usually start with randomly generated populations, however results found in the literature have shown that seeding the population with approximate solutions can significantly decrease the time to evolve a satisfactory solution[13].

The crossover operator is a mixing mechanism that aggregates information from parents and produces an offspring. For example, in human genetics, a child gets one set of genes from each parent. The mutation operator randomly changes the information held in a gene. Some researchers claim that mutation has the effect of reinjecting information, that may have disappeared through selection, into the population. Often application dependent operators are designed to enforce constraints. For example, order based crossover is used in many scheduling applications[18].

The stopping criteria is application dependent and can specified by when an individual meets some specification or after a given number generations has passed.

Using a simple genetic algorithms entails three main steps:

- design a solution representation
- design a genetic encoding of the solution
- design an appropriate evaluation function

The quality of the final solution and the speed with which it is evolved depends equally on the quality of all three designs.

The selection process is directly correlated with the fitness value obtained through evaluation, which in practice is a single valued function in virtually all evolutionary algorithms: the performance of an individual is aggregated into a single value. However, as mentioned in the introductory section, it can be difficult to combine satisfactorily the objectives into a single value.

In the genetic algorithms proposed in this chapter, the selection is driven by the notion of Pareto Optimality defined in the previous section. In this approach, the objectives are not aggregated using simple linear weighted sum schemes. Rather, a partial ordering is formed based on the number of solutions a particular solutions dominates. This dominance measure is then used a ranking which is subsequently transformed into a selection probability. In this way, the relative weightings of the different objectives do not affect the selection probabilities: solutions of the same rank (i.e., solutions that dominate the same number of other solutions), are equally likely of being selected for reproduction.

## GENETIC DESIGN OF FUZZY SYSTEMS

Fuzzy/Linguistic modeling is one of the major issues in fuzzy theory. The general methodology of fuzzy modeling consists of three steps: (1) external structure identification, (2) internal structure identification and (3) linguistic approximation. External structure identification involves the identification of relevant input variables. Internal structure identification involves partitioning of the space and identifying rule output parameters. Linguistic approximation is the final step of interpreting a fuzzy system by mapping its internal structure into linguistic terms (this step is often not performed).

In this section, we address the task of fuzzy system design by introducing a fuzzy system representation that is combined with a multiobjective evolutionary learning algorithm. The proposed technique is aimed at integrating the first two steps of fuzzy modelling outlined above: external and internal structure identification (partitions and parameters) of the fuzzy system are performed automatically and simultaneously. In addition, the technique yields a set of solutions that can assist the user in identifying the trade-offs between different objectives such as structural complexity and performance objectives.

The first subsection reviews fuzzy systems design and design techniques. Following this subsection, we will present an automatic fuzzy system design technique whose feature is to design multiple aspects of fuzzy systems in a single stage.

### Fuzzy Systems Design

A common way to employ fuzzy set methods is through fuzzy rule-based systems. In these systems, the idea is to partition the input space using combinations of fuzzy sets. In particular, the main idea is to produce multidimensional partitions and then assign actions to each of the partitions. In most applications, the partitions are synthesized from one-dimensional membership functions using fuzzy if-then rules. For example, if one dimensional Gaussian shaped membership functions were combined with a product operator, the result would be a hyper-ellipsoidal patch in the input space.

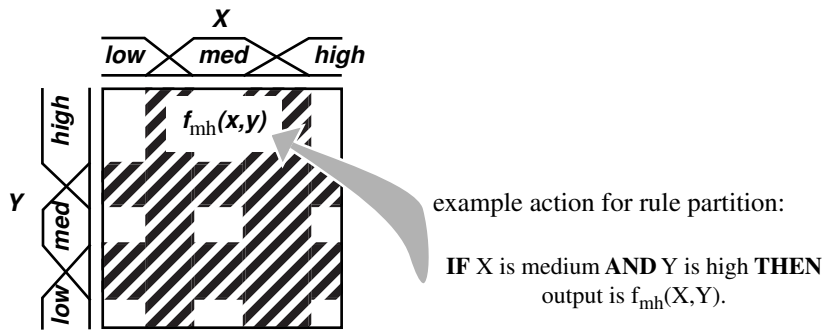


**Figure 1.** Example partitioning of a 2 Dimensional space using 3 trapezoidal membership functions per input dimension. The partitions are formed using fuzzy if-then rules such as:

**IF** X is medium **AND** Y is high **THEN** output is $f_{mh}(X,Y)$.

The consequent parts of the rules represent the actions associated with each partition. For example, partitions are formed by combining one-dimensional triangular membership functions. Then then method for resolving multiple active actions is necessitated. The fuzzy system design process can be summarized in the following five steps:

(1) decide the relevant inputs
(2) decide the number of rules
(3) decide the membership function shapes and locations
(4) decide the output actions associated with each partition
(5) decide the reasoning method used to aggregate multiple actions.

Step 1 is also considered external system identification. That is, the relevant inputs and outputs are external to the system. Common methods used to address this step are Principal Components Analysis or Group Method Data Handling. Although this represents an important step in system design, we will not treat it directly in the method described in this chapter.

From the description of the way multidimensional partitions are formed, it is clear that the membership functions and the number of rules are tightly related to the partitioning. For this reason, steps 2 and 3 are important to consider simultaneously.In the following sections, we describe the system and genetic representations of the fuzzy controller.

**Shared-Triangular Fuzzy System Architecture**

The *Shared-Triangular* representation uses asymmetrical triangular membership

functions and the *min* operator to synthesize multidimensional membership functions[12]. Each triangular membership function is specified by its center, left base width, and right base width.

Rules cover the input space by selecting and combining the one-dimensional membership functions from a globally defined set (all rules have access to the same set of membership functions). There is also a possibility for a rule to have no membership function associated with a particular input dimension, which implicitly forces the rule to cover entirely an input dimension.

Each rule has an additional parameter to modify, or hedge, the rule firing strength. If we define $\mu(\vec{x})$ as the raw rule firing strength, then the modified firing strength of a rule is given by:

$$\mu^p(\vec{x}), \tag{1}$$

where $p$ represents the degree of hedging. The consequent output of each rule has a Takagi-Sugeno-Kang (TSK) form with the output of the system computed by taking a weighted sum of the outputs of all firing rules[17].

**Genetic Representation**

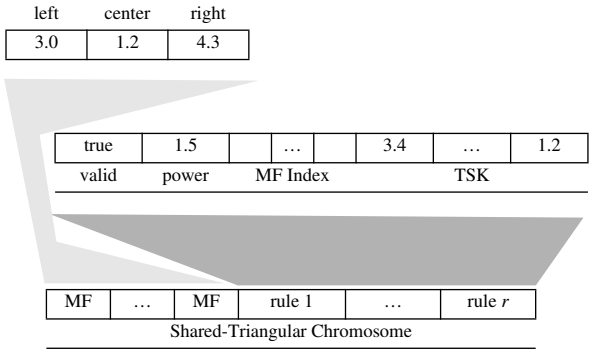


**Figure 2.** Shared-Triangular genetic representation showing rule validity parameter and rule power parameter. Note that there are $b + 1$ TSK parameters.

The genetic representation for the Shared-Triangular fuzzy systems is shown in Figure 2. The genetic code is made up of two different macro structures: membership function genes and rule genes. The membership genes specify distance between adjacent triangle centers and the left and right triangle base widths [13]. The rule genes

have locations for the membership function set indices for each input dimension as well as a validity bit. There is also a provision for completely covering an input dimension when the index is set to a special value.

The total number of system parameters in an $b$-dimensional system with $r$ rules and a maximum of $c$ membership functions per input dimension is:

$$3bc + r(2b + 3)\,. \tag{2}$$

Each rule requires $2b + 3$ parameters, because the membership function associated with each dimension must be specified (we implement this via an indexing scheme).

**Experimental Results**

As a validation of the proposed approach, we consider the problem of robot arm control. This system is well known in the literature as a benchmark for different learning approaches [8]. We use a data set of 1024 examples, divided into a training and testing data: the first 500 are used for training and the rest for testing. The system is described as follows: the position of the arm depends on the oil pressure *p(t)*, which itself a function of a valve opening. The variation of this opening is noted *u(t)*. The goal is to model a transfer function *f*,

$$f\colon u(t),\, u(t-1),\, p(t),\, p(t-1),\, p(t-2) \rightarrow p(t+1) \tag{3}$$

The high dimension of the input space, five, makes necessary the use of learning approaches for fuzzy systems. The higher the dimension, the more difficult it is to identify manually the structure and parameters of the desired fuzzy system. The performance error is defined by:

$$E = \frac{1}{n}\sum_{i=1}^{n} |y_i - d_i| \tag{4}$$

where $E$ is the measured error, $n$ is either $n_{traning}$ or $n_{test}$ for training error or test error. $n_{traning}$ and $n_{test}$ are respectively the size of training and test sets, $y_i$ the obtained value and $d_i$ the desired value for a particular example $i$.

In the first experiment, two objectives are defined: the usual error minimization as it is generally the case in supervised learning and minimizing the number of rules in the fuzzy system. The set of best alternatives live on the pareto set that compromises between the training error and the number of rules to achieve a performance level. Figure 3(c) shows the distribution of the solutions in the two dimensional objective space. Note that the frontier traces the edge that indicates the set of best alternatives. Figure 3(a) and (b) also show the distribution of the error and number of rules for the different agents after 1000 generations. Finally, it presents the evolution of the error through the different simulation generations for the best, average and worst individ-

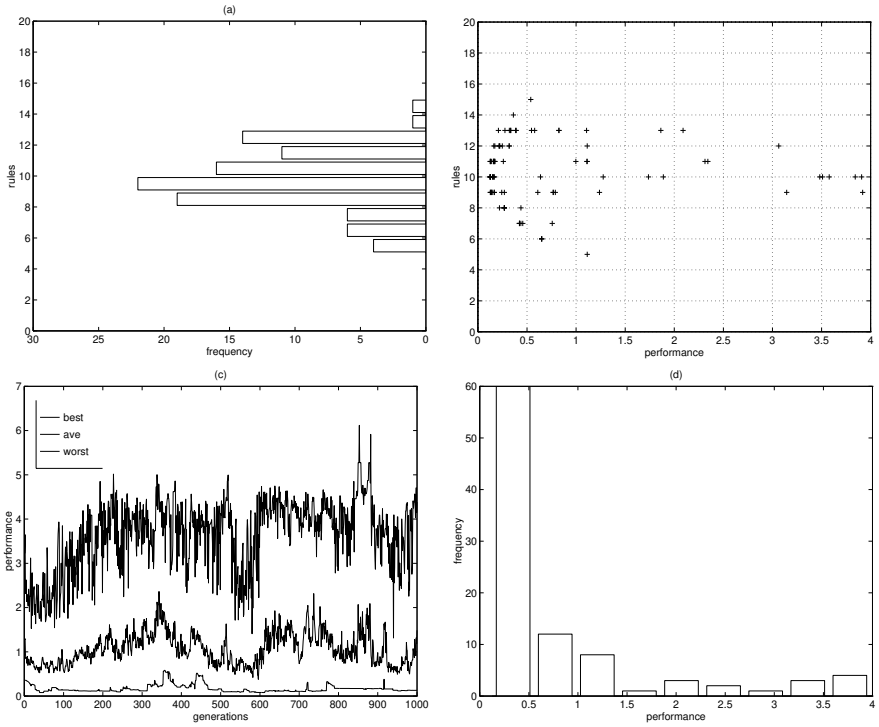


**Figure 3.** Results of the multiobjective evolutionary algorithm for designing fuzzy systems. (b) shows the solutions in the 2 dimensional objective space. (c) algorithm progress over time.

ual of the population. Figure 4 presents the real output and the predicted output through time.

In comparison with previous experiments using neural and neurofuzzy learning, the system performs very well with quite similar error measure for training and test sets. The main point is that the same performance is obtained with fewer rules, due to considering the second objective in the modeling. However, the learning time seems to be much longer. A deeper discussion of the advantages and disadvantages of this learning algorithm when compared to other learning approaches is beyond the scope of this chapter.

To test the robustness of the learning technique, other experiments have been conducted. A completely random vector is added as an additional input variable, extending the dimension of the input space to six. In this case, the system also performs well. The best solution, with respect to the error measure, found by the algorithm is

close to the solution found without the additional noisy input vector. The results appear in Figure 5. We can give different possible interpretations for this and they all conclude on robustness properties of the system. The first one is that the system has the ability to select appropriate variables for the modeling. The second is that the system adapts his parameters taking in account the noisy data.

**Extensions**

In the following, we are more concerned with the external structure identification issue. In many approaches to fuzzy modeling, the GMDH method is used to choose the variables incrementally[11]. The evaluation is based on the description and prediction power of the obtained fuzzy system. The main problem with this approach is that we have to perform an internal structure identification at each step of the incremental process and evaluate the system.

In the proposed approach, we address this problem using a multiobjective optimization technique. The proposed genetic coding has the ability to design rules that ignore input dimensions. This feature can be used to modify the system complexity objective to consider complexity on a finer grain: we can express complexity in terms of rule dimension along with rule number, which is used currently. If an *a posteriori* analysis is performed and shows that a particular input variable is ignored in some, but used in other rules, the system will have design rules with different structures. If a variable does not appear in any rule, then the system has performed variable elimination. Research in this direction is currently being conducted.

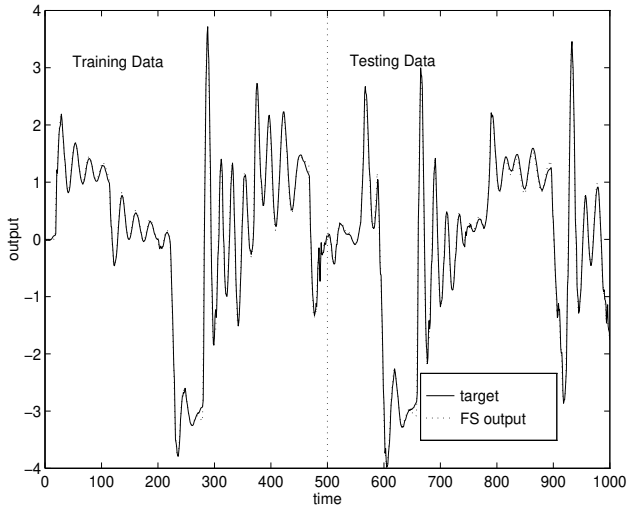


**Figure 4.** Target and model prediction for controlling a 5 dimensional input robot.

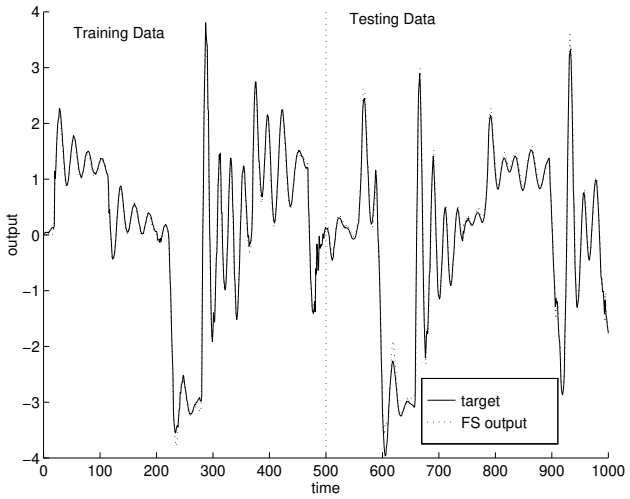


**Figure 5.** Target and model prediction with additional noisy input vector.

## FUZZY-GENETIC ALGORITHMS

In our work, we extend hybrid fuzzy system/evolutionary algorithm techniques developed in [13,14]. The previously developed approach used a fuzzy system to monitor and control the behavior of a genetic algorithm. A schematic diagram of the system is shown in Figure 6. An example fuzzy system can include rules that relate

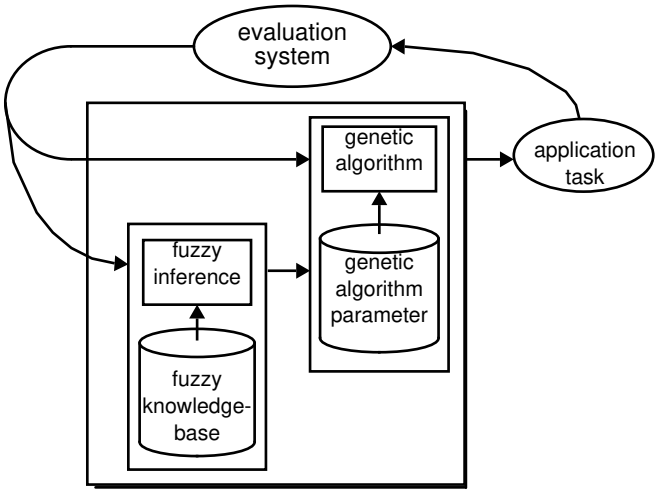


**Figure 6.** Schematic diagram of the Dynamic Parametric Genetic Algorithm. Genetic algorithm search behavior is monitored and controlled by a fuzzy system.

indicators measuring diversity or distribution characteristics to control parameters

such as population size or mutation rate. A detailed view of example inputs and outputs to a typical fuzzy system in this context is shown in Figure 7. This approach has

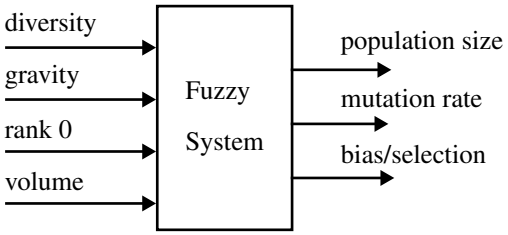


**Figure 7.** Inputs and outputs to a typical fuzzy system used to monitor and control a multiobjective evolutionary algorithm. Details of the particular inputs and outputs of this system are discussed in the text.

been shown to improve search performance of single objective optimization algorithms. The techniques proposed in this chapter aim at applying these techniques to multiobjective optimization algorithms.

The next subsection focuses on developing indicators that characterize the search process and use them as inputs to a fuzzy system that effects various parameters of the search algorithm, such as selection operator parameters. The following subsection describes a meta-level evolutionary algorithm technique for automatically designing search control strategies for multiobjective evolutionary algorithms.

**Characterizing Solution Sets**

Search performance is difficult to define and can be dependent on the final application; what constitutes a good search depends on the quality of the output within a given context. However, once some performance measures are defined, the search performance can be optimized using techniques such as those proposed in [9] and [14].

Several generic search performance measures have been invented such as online and offline measures [3]. Online measure is a running average of all solutions evaluated up to a given point in time and offline measure is a running average of the best solution evaluated up to a given point in time. These measures have been used to tune genetic algorithm parameters and develop fuzzy system controllers to improve the performance of single objective optimization algorithms. As a result they must be adapted for use in multiobjective optimization applications. A discussion of the modification of these measures will be given below.

In the case of multiobjective optimization, the goal is to provide the users with a set of solutions to choose among. The following properties of the final solution set, in the objective space $f(X)$, may indicate how well the Pareto optimal set is represented (for minimization):

- the diversity of the non-dominated solutions in the final set is maximized
- the number of non-dominated solutions in the final set is maximized
- the bounding volume of the non-dominated solutions set is maximized
- the center of gravity of the final solution set is close to the origin

These properties correspond to moving the cloud of points toward the origin, but also spreading out uniformly, in a sense, to cover the true Pareto optimal set. The pressure to pull the center of gravity toward the origin corresponds to simultaneous minimization of all constraints.

In the next section we will define solution set quality implicitly by modeling the choice process ultimately performed by the user of the algorithm. Preliminary results using the new measure follow.

*Solution Set Quality based on a User Choice Model*

From one or more given sets of solutions the user will ultimately select a single solution as the "best". For example, in the case of IC placement, a single layout will ultimately be selected for production. However, it is not known *how* the user makes the final choice; it depends on preferences which are probably never formally expressed.

The basic idea of the proposed alternative solution set quality measure is to model the final selection performed by the user by a selection function parameterized to account for different possible preferences with respect to the relative importance of the optimization criteria. By systematically varying the parameters of the selection function, a class of functions corresponding to a wide range of possible user-preferences is obtained. The quality $q(X)$ of a set $X$ is then defined as the expected value of the selection function when selecting from $X$ while traversing the class of selection functions.

The measure $q$ is introduced and described in detail in [5]. Here we just outline the basic definition and its properties. To compute this measure, a normalization of the cost dimensions is needed. Let $\overline{f}_i(x) \in [0, 1]$ denote the normalized cost of $x \in X$ with respect to the *i'th* dimension. The selection function $s_w(x)$ is then defined as

$$s_w(x) = \sum_{i=1}^{n} w_i \overline{f}_i(x) \tag{5}$$

where $w = (w_1, \ldots, w_n) \in W$ is a weight vector. $s_w$ can be considered to be a simple model of the final choice process where $w$ represent the relative importance of the optimization criteria and the solution selected is the one minimizing $s_w$. The solution set quality measure $q$ is then defined as

$$q(X) = E(min\{s_w(x) \mid x \in X\}) \tag{6}$$

where the expected value $E$ is computed over all $w \in W$, a given weight space. Varying $w$ over $W$ generates a class of selection functions $s_w$, thus accounting for a range of different user preferences. For example, $W$ can be defined by assuming that each $w_i$ is uniformly distributed on [0,1] and that the $w_i$ s are independent.

For practical computation, $q(X)$ can be estimated by

$$\hat{q}(X) = \frac{1}{N} \sum_{k=1}^{N} min\{s_{w(k)}(x) \mid x \in X\} \tag{7}$$

where each $w(k)$ is generated according to the definition of $W$ and $N$ is the sample size.

The following intuitively desirable properties of $q$ are proven in [5]:

1. Only non-dominated solutions of a set $X$ can contribute to $q(X)$. Consequently, adding a dominated (i.e. poor) solution to a set can not improve its quality.
2. Adding a non-dominated (i.e. good) solution to $X$ either leaves $q(X)$ unchanged or improves $q(X)$.
3. If a set $X$ is clearly better than $Y$ in the sense that every solution in $Y$ is dominated by some solution in $X$, then the quality of $X$ is always at least as good as that of $Y$, i.e., $q(X) \leq q(Y)$. This holds independently of the sizes of $X$ and $Y$.

The last property assures that algorithm $A$ (e.g. a random walk) can never obtain better solution set qualities than algorithm $B$ (e.g. a genetic algorithm) when the sets generated by $B$ dominates those of $A$.

A remaining serious limitation of (6) is that, as reflected in 2 and 3 above, $q(X)$ is not guaranteed to improve when a non-dominated solution is added; it may stay unchanged. Specifically, if a non-dominated solution is non-convex relative to the other non-dominated solutions in $X$, it can never improve $q(X)$ (see [5] for a more detailed discussion of this phenomenon).

Figure 8 illustrates the set quality measure on eight constructed sets s1 through s8, assuming two optimization dimensions. The corresponding estimated set quality values are given in Table 1. Except for sets s5 and s6, set quality decreases strictly with the set number, as seems intuitively reasonable. However, s5 is not better than s6

because the solution with cost (8,7) is non-convex relative to s5.

**Table 1.** Estimated set qualities for sets s1 through s8, using (7) with $N = 50,000$.

| set | s1 | s2 | s3 | s4 | s5 | s6 | s7 | s8 |
|---|---|---|---|---|---|---|---|---|
| $\hat{q}$ | 0.0001 | 0.0762 | 0.2127 | 0.2642 | 0.4130 | 0.4130 | 0.5232 | 0.5253 |

*Characterizing Dynamic Search Behavior*

In single objective evolutionary algorithms, it is common practice to store the best solution found either external or internal to the population. In a multiple objective optimization setting, the final result desired is not just one solution, but a family of solutions. The analogous procedure is to store the non-dominated solutions. There is an extra operation when merging the non-dominated solutions of the current population with the saved set, which is to remove any solutions that are replaced by incoming solutions. That is, the non-dominated set determination must be performed on the saved set. When the evolutionary algorithm terminates, we are left with a set that, for our particular run, approximates the true Pareto optimal set.

Because we are interested in the dynamic behavior of search algorithms, we require a performance metric that takes time into account. We present the following two dynamic set quality measures:

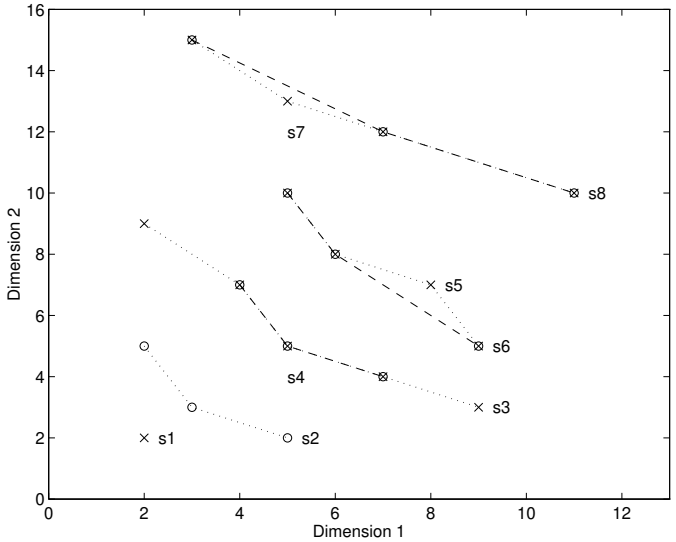


**Figure 8.** Each set is indicated by either circles connected by a dotted line or crosses connected by a dashed line. s4 is a subset of s3, s6 a subset of s5 and s8 a subset of s7.

$$online_{mo} = \frac{\sum_{t=0}^{T} q(X^t)}{T+1} \tag{8}$$

$$offline_{mo} = \frac{\sum_{t=0}^{T} q(X_0{}^t)}{T+1} \tag{9}$$

where $X^t$ is the set at time $t$. These measures give us an indication of how the set quality changes over time. Both of these measures should be maximized when the target algorithm performs minimization on all objective dimensions. The online measure effectively computes the expected set quality value of all sets generated by an algorithm up to a given point in time and the offline measure computes the expected set quality value of the best set ever generated by an algorithm up to a given point in time. (These measures can be seen as generalizations of performance measures used to compare single objective search algorithms [3].)

**Designing Search Control Strategies**

An approach to designing the fuzzy system used to monitor and control the search behavior is a meta-level technique. The underlying task at hand is that of designing a fuzzy controller for a dynamic process. Many techniques such as neuro-fuzzy and evolutionary techniques for adapting fuzzy system exist. In this work, we will use a technique proposed in [13].

The fuzzy system is coded as a string of information which is then searched using a meta-level genetic algorithm. In this case, the fitness of a fuzzy system will be how well it is able to control a search algorithm to maximize a search performance measure. For example, we can use the offline measure formulated in (9) and search for a fuzzy system that maximizes this measure when the algorithm is run on several test functions (where the test functions represent the types of problems we are ultimately interested in). A diagram showing the relationship between the meta-level search and evolutionary search algorithm is shown in Figure 9.

**Experiments and Results**

The techniques presented in this chapter have been investigated using the application briefly described below as a case study. For a detailed description of the application the reader is referred to [4].

*The IC Placement Problem*

A certain type of integrated circuits (ICs) consist of a number of interconnected blocks. Each block is rectangular and contains a number of interconnected transistors implementing some part of the functionality of the circuit. For example, in a CPU, RAM and ALUs constitute blocks. The IC placement problem is that of placing the blocks in the plane such that a number of competing objectives are optimized while satisfying given geometrical constraints, of which the most important is that blocks

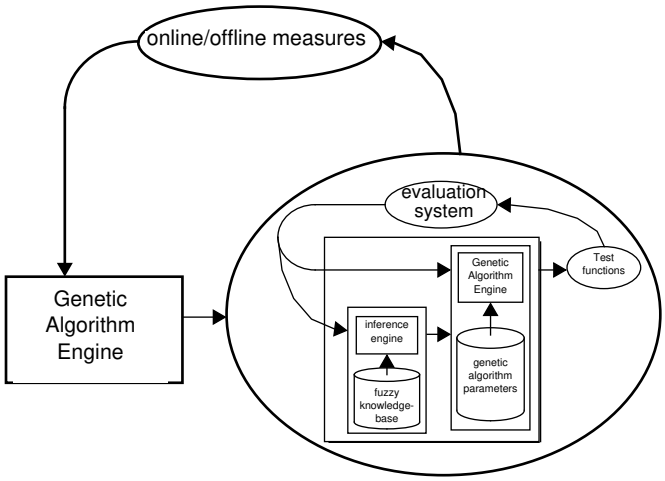


**Figure 9.** Meta-level technique for evolving hybrid fuzzy-genetic multiobjective evolutionary algorithms. Algorithms are optimized with respect to search performance measures such as the online and offline measures formulated in eq. 8 and 9 in the text. The test problems are representative of the class of problems

cannot overlap. Communication between blocks is physically made possible by (a large number of) wires connecting specific points within each block to points within other blocks. Since the locations of connection points within a block moves as the block is rotated and/or mirrored in one or both of the axes, determining which one of eight possible orientations for each block is also part of the placement problem. A sample placement is shown in Figure 10.

The algorithm in [4] minimizes three competing criteria:

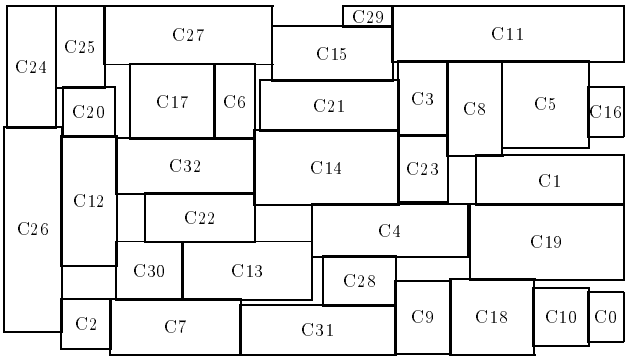


**Figure 10.** A layout of 33 blocks named C0 through C32. No wires or connection points are shown.

- The layout area, i.e., the area of the smallest bounding box containing all blocks. The area is the main factor determining the production cost of the circuit.
- The deviation of the aspect ratio of the layout (height divided by width) from a user-defined target value. The aspect ratio has to be within a certain range in order for the layout to fit in a given circuit package.
- The maximum estimated time it may take some signal in the circuit to propagate from one memory element to another, passing through a sequence of blocks an interconnecting wires. Minimizing this quantity is equivalent to maximizing the clock frequency at which the circuit will be able to operate correctly.

The placement problem is of significant interest to the IC design industry, and is very challenging due to its complexity. It is NP-hard and the search spaces represented by practical problem instances are usually extremely large. Furthermore, the criteria considered, especially the clock frequency maximization, are ill-behaved functions of the block placement. Consequently, high-performance state-of-the-art circuits are still being placed manually.

*Algorithm Characteristics*

The placement algorithm is a steady-state, hybrid GA, making extensive use of problem-specific knowledge. The essential part of the phenotype is an inverse Polish expression specifying the relative positions of all blocks. Only placements known as *slicing structures* are considered, for which a unique representation by Polish expressions were first introduced in [19]. A placement is a slicing structure if it is possible to recursively partition the placement into two parts by a cut-line such that, in the end, each partition contains only one block. Each cut-line has to be horizontal or vertical, span the entire layout, and not intersect any block. Given a slicing structure, the absolute positions and orientations of blocks are determined using problem-specific algorithms. Blocks are oriented by an algorithm assuring that the minimum area for the given Polish expression is obtained, and a compaction algorithm is then applied to further reduce layout area. The phenotype and its interpretation is illustrated in Figure 11.

Crossover and mutation of the Polish expression parts of the phenotypes are performed using operators introduced in [2]. A key feature of these operators is that they all preserve feasibility, that is, only feasible solutions, which can be interpreted as valid placements using the scheme outlined above, are ever considered by the GA.

Selection for crossover is rank based following [7,18], i.e., no explicit fitness computation is performed. Given a population $\Phi = \{\phi_0, \ldots, \phi_{M-1}\}$, and assume $r(\phi_0, \Phi) \leq \ldots \leq r(\phi_{M-1}, \Phi)$. Each parent $\phi$ is selected such that: 1) the probability that $r(\phi, \Phi)$ equals $r(\phi_k, \Phi)$, $P[r(\phi, \Phi) = r(\phi_k, \Phi)]$, decreases linearly with $k$, and $P[r(\phi, \Phi) = r(\phi_0, \Phi)] = \beta P[r(\phi, \Phi) = r(\phi_{M/2}, \Phi)]$, where a user-defined parameter,

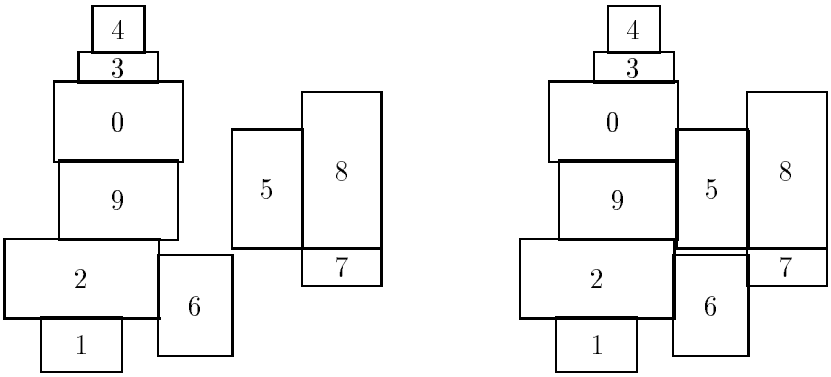


**Figure 11.** Given 10 blocks and the Polish expression *1 2 + 6 * 9 0 + + 3 4 + + 5 * 7 8 + *, where + and * represent a vertical and horizontal cut-line, respectively, and the numbers are block identities. The placement to the left is the result of placing and recursively centering subtrees of the expression while orienting each block so that the minimum layout area is obtained. Further reduction of the layout area is then attempted using a compaction algorithm, yielding

$1 < \beta \leq 2$, controls the selection pressure. 2) all individuals having the same rank have the same probability of being selected.

Generated offspring replace poor solutions, yielding an elitist strategy in which a currently non-dominated solution $\phi$ is never removed from the population unless its replacement dominates it.

**Previous Results**

It is difficult to compare an algorithm performing multi-dimensional optimization to a traditional one-dimensional optimization, e.g., minimizing a weighted sum of the criteria considered. However, in special cases, where a comparison to simulated annealing (SA) can be performed, the results obtained by the GA are shown in [4] to be comparable to those of SA, and in this sense, the GA approach to placement is very promising.

**Results**

Our goal is to develop a genetic algorithm that performs well in the context of multi-objective optimization. We performed several IC placement experiments with random walks and previously designed genetic algorithms to evaluate the effectiveness of our proposed dynamic genetic algorithm methods. The results reported in the following sections are obtained from experiments using a real world design consisting of 20 blocks and 248 interconnections[1]. The computational resources for each search was set to 200 cpu seconds on an HP715/64 workstation, which would be much higher in practice.

The random walk used the external Pareto Set mechanism describe in a previous section to build up a final solution set. The static genetic algorithm used static parameter settings for population size, mutation rate, and bias, whose values were set according to experience.

The optimized static genetic algorithm used a different set of static parameters for population size, mutation rate, and bias. These values were obtained by optimizing the dynamic search performance measure presented at the beginning of this experimental section. The meta-level optimization was set up as previously described and was allowed to run for 500 generations.

The structure of the controller used in the dynamic genetic algorithm were: set quality, set quality change since the previous control action, and time left (See Figure 12).

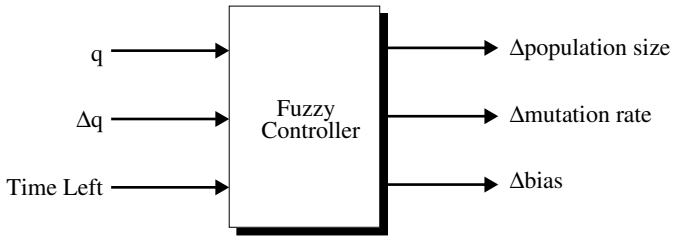


**Figure 12.** Inputs and outputs of the fuzzy controller used to modulate genetic algorithm parameters.

The change in set quality since the previous control action, is measured as follows:

$$\Delta q(X^t, X^{t-1}) = \frac{q(X^t) - q(X^{t-1})}{q(X^{t-1})}. \quad (10)$$

Based on this controller structure, we then tried to find a set of rules, and initial conditions for the outputs, that optimizes the performance measure. The internal struc-

ture of the controller used the shared-triangular-membership function representation presented in an earlier section. There was a maximum of 10 rules per output variable.

The results of the random walk, static genetic algorithm, optimized static genetic algorithm, and dynamic genetic algorithm are given in Table 2 below. Each of the entries were compiled from ten independent runs. The first and second columns report the mean and standard deviations of the various algorithms according to (7). The third and fourth columns report the same statistics on the final output set, $X^{t_{final}}$, obtained by the algorithms.

**Table 2.** Dynamic and Final $q$ Performance Measures

| Algorithm | $\overline{DSM}$ | $DSM_\sigma$ | $\hat{q}(X^{t_{final}})$ | $q_\sigma(X^{t_{final}})$ |
|---|---|---|---|---|
| Random Walk | 0.406 | 0.036 | 0.376 | 0.031 |
| Genetic Algorithm | 0.311 | 0.029 | 0.273 | 0.030 |
| Optimized Genetic Algorithm | 0.297 | 0.015 | 0.264 | 0.012 |
| Dynamic Genetic Algorithm | 0.299 | 0.023 | 0.261 | 0.017 |

According to these results, using a genetic algorithm for IC placement improves both the final value and offline measure relative to random search. The optimized static genetic algorithm showed improvements in both the mean final value and mean offline measures over the genetic algorithm with manually set parameter values. One of the most striking results is that the optimized genetic algorithm had much lower standard deviations than the other algorithms.

## SUMMARY, CONCLUSIONS, AND EXTENSIONS

In this chapter, we presented fuzzy/genetic multiobjective techniques for intelligent systems design tools and components. The fuzzy system design method proposed in this chapter integrates structure and parameter identification while optimizing multiple objectives. Although the results presented demonstrate complexity and a single performance metric trade-off, the technique can be extended to include additional objectives.

The development of good performance indicators is an essential step toward improving the performance of multiobjective search algorithms and understanding their behavior. In this chapter, we formulated a search performance metric aimed at characterizing multiobjective search performance. Although the measure captured intuitive notions of final set quality, the measure can be augmented further using by adding other knowledge based structures. The measure should be enhanced to consider the cognitive load on the user. That is, decision making is not supported if the user is given too many solutions. Interactive multiobjective optimization techniques developed in the context of linear optimization can be applied to the techniques pre-

sented in this chapter.

Results comparing genetic algorithms using empirically determined parameter settings and genetic algorithms with optimized parameter settings show that performance can be significantly improved by the meta-level optimization approach presented in this chapter. The key points for making the approach practical, however, are 1) formulating the appropriate search performance measure and 2) dealing with various forms of noise inherent to meta-level optimization.

## EXERCISES

1. Messy Genetic Algorithms allow for genetic representations of different lengths. Adapt the fuzzy system representation and genetic operators presented in this chapter to accommodate messy genetic algorithms.

2. In the genetic algorithm presented in this chapter, the primary goal of the evaluation step is to produce a partial ordering of the population. Adapt the ordering scheme in this chapter to one that incorporates goals and feasibility regions.

3. What effect do the operator probabilities have on the final set characteristics? How could you incorporate human interaction to drive the characteristics of the final set?

4. How could you accommodate additional constraints into the multiobjective optimization technique proposed in this chapter?

5. The solution set quality presented in this chapter does not incorporate application specific knowledge. For an application in your field, formulate a linguistic based set quality measure.

## ACKNOWLEDGMENTS

This research is supported in part by NASA Grant NCC-2-275, ONR Grant N00014-96-1-0556, and BISC program of UC Berkeley. The author would also like to thank Riad Hartani of NSERC in Canada.

## REFERENCES AND RELATED PUBLICATIONS

[1] Asanovic, A., Beck, J., "T0 Engineering Data, Revision 0.14," Technical Report, International Computer Science Institute, Berkeley, CA, 1994.

[2] Cohoon, J.P., Hedge, S.U., Martin, W.N., Richards, D., "Distributed Genetic Algorithms for the Floorplan Design Problem," *IEEE Transactions on Computer-Aided Design*, Vol. 10, pp. 484-492, April 1991.

[3] DeJong, K.A. (1975) An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Dissertation, University of Michigan, University Microfilms No. 68-7556.

[4] Esbensen, H., Kuh, E.S., "An MCM/IC Timing-Driven Placement Algorithm Featuring Explicit Design Space Exploration," *Proc. of the 1996 IEEE Multi-Chip Module Conference*, pp. 170-

175, 1996.

[5] Esbensen H., Kuh, E.S., "Design Space Exploration Using the Genetic Algorithm," *Proc. of the IEEE International Symposium on Circuits and Systems*, 1996 (to appear).

[6] Eschenauer, H., Koski, J., and Osyczka, A. (1990) editors, "Multicriteria design optimization : procedures and applications," New York, Springer-Verlag.

[7] Fonseca, C.M. and Fleming, P.J. (1993) "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion, and Generalization," in *Proc. of the Fifth International Conference on Genetic Algorithms*, ed. S. Forest, San Mateo, CA: Morgan Kaufmann.

[8] Glorennec, P.Y. (1993) "Algotihmes d'Apprentisage pour Réseaux Neuroflous," Internal report, IRISA.

[9] Grefenstette, J.J. (1986) "Optimization of Control Parameters for Genetic Algorithms", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 16, No. 1.

[10] Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA.

[11] Ihara, J. (1980) "Group Method of Data Handling Towards a Modelling of Complex Systems," *Systems and Control*, Vol. 24, pp. 158-168.

[12] Lee, M. A. (1995) "On Genetic Representation of High Dimensional Fuzzy Systems," *Proc. of NAFIPS'95*, College Park, MD, IEEE Computer Science Press, pp. 752-757.

[13] Lee, M. A. and Takagi, H. (1993) "Integrating Design Stages of Fuzzy Systems using Genetic Algorithms," *Proc. IEEE Int. Conf. on Fuzzy Systems* (FUZZ-IEEE '93), San Francisco, CA, pp.612-617.

[14] Lee, M.A. and Takagi, H. (1993) "Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques," in *Proc. of the Fifth International Conference on Genetic Algorithms*, ed. S. Forest, San Mateo, CA: Morgan Kaufmann.

[15] Ralescu, A. and Hartani, R. (1995) "Some Investigations in Fuzzy and Linguistic Modelling," IEEE/IFES Conference on Fuzzy Systems, Yokohama, Japan.

[16] Shafer, J.D. (1985) "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," in *Proc. of the First International Conference on Genetic Algorithms*, ed. J.J. Grefenstette, Erlbaum.

[17] Takagi, T. and Sugeno, M. (1985) "Fuzzy Identification of Systems and Its Applications to Modelling and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.5, No.3, pp. 116-132.

[18] Whitley, D., "The Genitor Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best," *Proc. of the Third International Conference on Genetic Algorithms*, pp. 116-121, 1989.

[19] Wong, D.F., Liu, C.L., "A new algorithm for floorplan design," *Proc. of the 23rd Design Automation Conference*, pp. 101-107, 1986.