Automatic Construction of Fuzzy Controllers for Evolutionary Multiobjective Optimization Algorithms

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

Abstract

In this paper, we present techniques for designing fuzzy systems for controlling the behavior of multiobjective evolutionary algorithms. The aim of this work is to develop methods for improving the performance and understanding the behavior of multiobjective evolutionary algorithms. Because the output of a multiobjective optimization algorithm is a set rather than a single point, we present a search performance metric based on a *Set Quality* measure. According to this set quality measure, we demonstrate our techniques by developing fuzzy control strategies for a genetic algorithm based IC placement application.

1 Introduction

Many real world design problems involve trading off performance along more than one dimension. 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 [2]. Because more than one solution may meet the application specifications, each of these solutions must be regarded as equal.

One approach to this problem is known as multiobjective optimization. The goal of this approach is not to provide the users with a single solution, but rather a set of solutions that represent the set of best alternatives, or *Pareto Optimal* set. The notion of Pareto optimality is based on the concept of dominance. Let $\vec{f}(x) = (f_1(x), \ldots, f_n(x)$ represent a vector valued objective function and u and v represent two solutions, and assume each of the f_i is to be minimized. u dominates v, written $u <_d v$, if and only if $\forall i: f_i(u) \leq f_i(v)) \wedge (\exists i: f_i(u) < f_i(v)$. Solutions included in the *Pareto Optimal Set* are those that cannot be improved along any dimension without simultaneously being deteriorated along other dimension(s).

Evolutionary algorithms (EAs) have been applied to multiple objective optimization problems. EAs are population based stochastic search strategies that employ mechanisms found in natural genetics[6]. However, there are a number of unsolved problems not addressed in these techniques relating to the quality of the solution sets found and the efficiency of the search process. Non-dominated solutions may be very few and/or clustered in a small region of the cost-space, only providing the user with limited tradeoff information. Hence, the need for developing solution set quality measures and systematic techniques for controlling and improving quality of the final solution set.

This paper extends the hybrid fuzzy system/evolutionary algorithm techniques developed in [9,10]. The previously developed approach uses a fuzzy system to monitor and control the behavior of a genetic algorithm. A schematic diagram of the system is shown in Figure 1. The fuzzy system can include rules that relate indicators measuring population distribution characteristics to control parameters such as population size or mutation rate. The work reported in this paper is a continuation of the work described in [7].

Section 2 presents the Set Quality measure used in our experiments. Section 3 presents a fuzzy evolutionary algorithm architecture and exemplifies how performance can be improved by modifying the algorithm behavior through fuzzy control. Section 4 discusses techniques for acquiring high performance search control strategies.

2 Set Quality Measure

From one or more given sets of solutions the user will ultimately select a single solution as the "best". For exam-

0-7803-3645-3/96 $5.00©1996 IEEE

ple, in the case of IC placement, a single layout will ultimately be selected for production. The basic idea of the proposed 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 wrt. 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 outlined below (a detailed description can be found in [3]). Let $\bar{f}_i(x) \in [0, 1]$ denote the normalized cost of $x \in X$ wrt. the *i'th* dimension. The selection function $s_w(x)$ is then defined as

$$s_w(x) = \sum_{i=1}^{n} w_i \bar{f}_i(x) \quad (1)$$

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

$$q(X) = E(min\{s_w(x) \mid x \in X\}) \quad (2)$$

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.

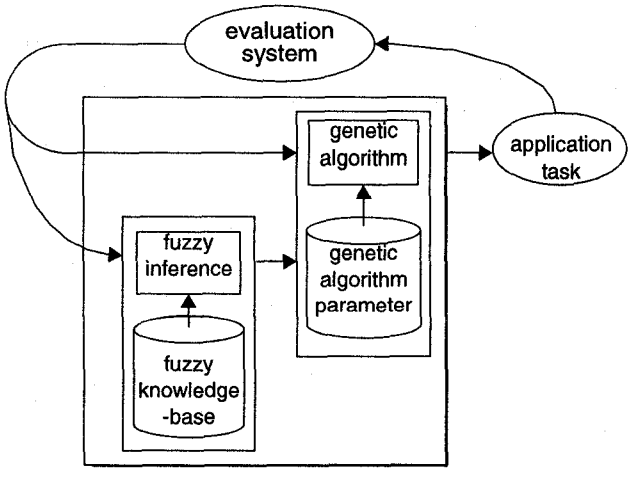

Figure 1. Schematic diagram of the Dynamic Parametric Genetic Algorithm. Genetic algorithm search behavior is monitored and controlled by a fuzzy system.

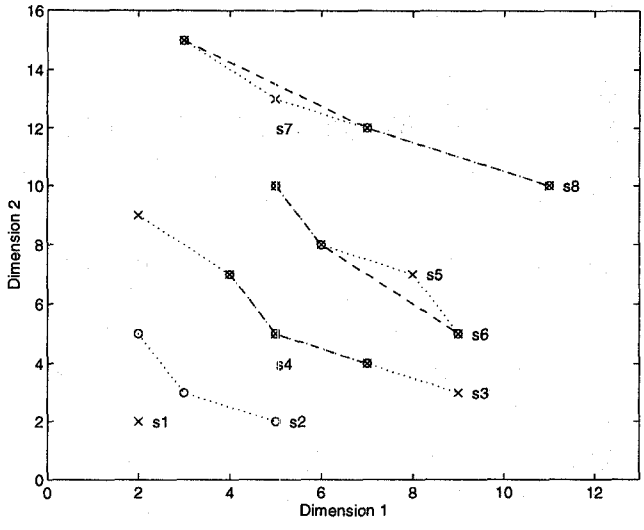

Figure 2. Each set is indicated by either circles connected by a dotted line or crosses connected by a dashed line. s4 is a subset of s3, s6 a subset of s5 and s8 a subset of s7.

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

$$\hat{q}(X) = \frac{1}{N} \sum_{k=1}^{N} min\{s_{w(k)}(x) \mid x \in X\} \quad (3)$$

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

Figure 2 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.

Table 1: Estimated set qualities for sets s1 through s8, using (3) 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

3 Hybrid Fuzzy-Genetic Algorithms

In this section we demonstrate how an evolutionary algorithm can be controlled by a fuzzy system to improve the final solution set quality.

3.1 Architecture and Case Study

In the experiments relating to this work, we have designed a fuzzy controller with the following inputs: set quality, set quality change, and time left (See Figure 3). The inputs and outputs chosen can be substituted with other indicators and control outputs. For example, other output parameters controlling crossover probabilities or selection operators can be controlled using this structure. Further research is warranted in this area.

The change in set quality can be measured as follows:

$$\Delta q = \left(\frac{q(X^t) - q(X^{t-1})}{q(X^{t-1})} \right). \quad (4)$$

where X^t is the population at time t.

A time-based input was included after considering how tasks are usually defined; both a problem specification and deadline are given. Humans faced with a time constraint will usually alter their strategy by not how much time has elapsed, but by how much time remains.

Outputs of our system are the change in population size, and mutation rate. Each of the outputs was coded as a multiplier to the current value. In addition, following the parameter update, hard limiting was enforced to bound population size to [10, 200] and mutation rate to [0.00001, 0.01]. An additional constraint on the population size was that it could never shrink to below the size of the non-dominated set.

Rules used in our experimental controller are:

IF time left is small THEN increase mutation rate

IF Δq is close to zero THEN decrease mutation rate

IF time left is small THEN decrease population size

(These manually constructed rules are fixed and are presented for illustration purposes only.) The system used triangular membership functions which were combined using the MIN operator. The membership functions were initially set to equally cover the input ranges and then tuned by trial and error. The output of each rule was a singleton value. The final values were aggregated using WEIGHTED-SUM defuzzification.

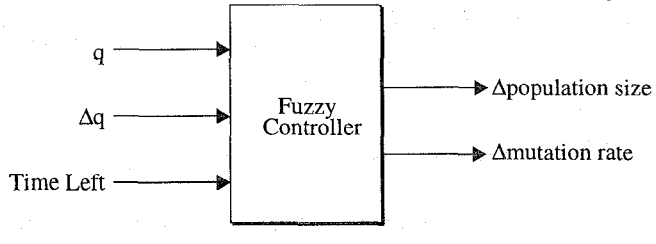

Figure 3. Inputs and outputs of the fuzzy controller used to modulate genetic algorithm parameters.

Initial values for the mutation rate and population size were set according to a meta-level optimization described in Section 4.1: population size 153 and mutation rate 0.001315.

3.2 Experiments and Results

The techniques presented in this paper 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 [2].

A class of integrated circuits (ICs) consist of a number of rectangular 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 cannot overlap. In practice, area, delay, aspect ration, and routing congestion are among the key objectives. A sample placement is shown in Figure 4.

The results applying random walk, a static genetic algorithm, an optimized static genetic algorithm, and a dynamic genetic algorithm to the placement problem for a simple circuit are given in Table 2 below.

The random walk algorithm generates random solutions and updates and stores the set of non-dominated solutions. To equalize the search space size, the random walk generated random genetic strings which are passed through the same decoding and evaluation routines that the genetic algorithms use. The static genetic algorithm uses static parameter settings.

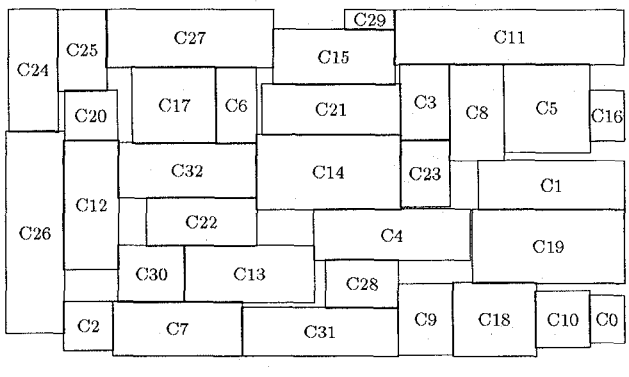

Figure 4. A layout of 33 blocks named C0 through C32.

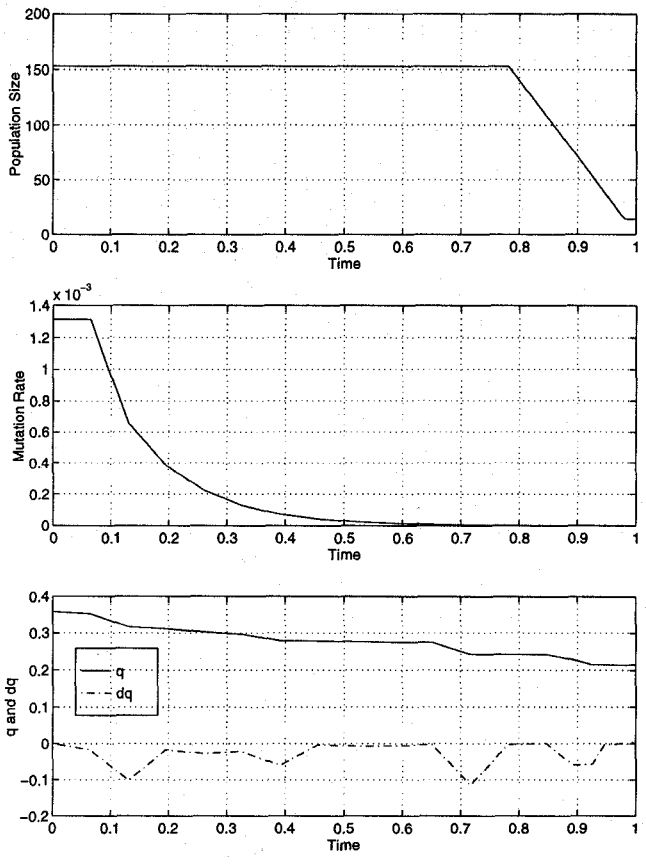

Figure 5. Dynamic population size and mutation rates are shown in the first two plots, respectively. The bottom plot shows how q and dq, set quality and Δq change over time.

The optimized static genetic algorithm used a different set of static parameters for population size, mutation rate, and bias. These parameter values were obtained by optimizing the algorithm using a technique presented in Section 4.1.

The dynamic genetic algorithm uses the controller described in Section 3. An example run produced the dynamic population sizing and mutation rate shown in Figure 5.

Table 2: Final q Performance Measures

Algorithm	$\bar{q}$	q_σ
Random Walk	0.376	0.031
Genetic Algorithm	0.273	0.030
Optimized Genetic Algorithm	0.264	0.012
Dynamic Genetic Algorithm	0.258	0.024

Entries of Table 2 were compiled from ten independent runs. For each algorithm, the first and second columns report the average and standard deviation of the final values, respectively.

According to these results, using a genetic algorithm improves the final set quality relative to random search. Optimizing the static parameter settings for the genetic algorithm shows further improvement in both the average final value and its standard deviation. The standard deviation for this optimized static genetic algorithm is reduced to less than half the value attained by the other algorithms. The dynamic genetic algorithm further improves the average final set quality, however, the standard deviation increases.

4 Acquiring Search Control Strategies

In this section, we present techniques for acquiring and refining the dynamic search control strategies. The first method is based on a meta-level optimization of the fuzzy rule-base. The second method aims at acquiring knowledge by monitoring user interaction patterns.

4.1 Meta-level Optimization of the Fuzzy Rule-Base

One approach to acquire the fuzzy rule-based control strategy using a meta-level genetic algorithm technique proposed in [9].

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 maximize a search performance measure through dynamic control. For example, we used the final set quality after a fixed amount of CPU time as the performance measure to be maximized. A diagram showing the relationship between the meta-level search and evolutionary search algorithm is shown in Figure 6.

4.2 User Interaction Modeling

The meta-level optimization proposed in the previous subsection requires a significant amount of computation. Learning the rule base requires simulating many controllers on several circuits. However, from a practical point of view, the search techniques to date may not be efficient enough to deliver a solution in an acceptable amount of time. Therefore, it may be advantageous to leverage from the perceptual capabilities of a human user to assist in the development of the algorithm control strategy. In this section, we present an approach that attempts to build a user model through observation.

The tool described in [2] has a graphical user interface, allowing the user to interactively view and control the opti-

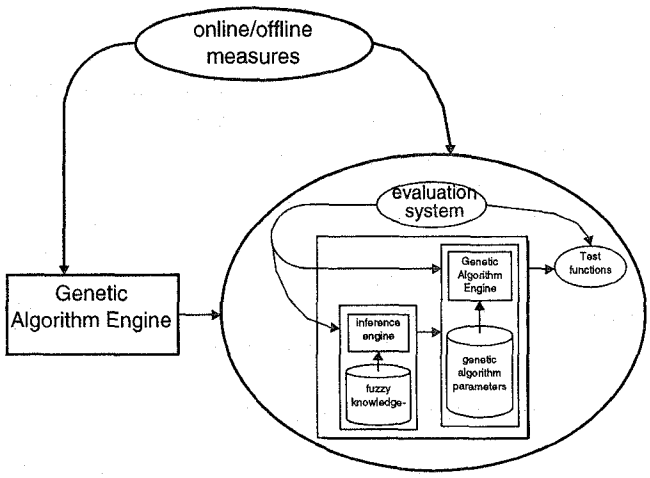

Figure 6. Meta-level technique for evolving hybrid fuzzy-genetic multiobjective evolutionary algorithms.

mization process. The current state of the process is continuously displayed to the user and as the optimization progresses and knowledge of obtainable cost trade-offs is gained, the user can perform various actions aiming at obtaining the best possible solution set relative to his/hers preferences.

The current state of the optimization is visualized in the form of graphs showing the cost trade-offs of the solutions obtained so far. Up to four independent graphs can be shown simultaneously, and each graph can be 2- or 3-dimensional with any of the four cost criteria on any axis.

Given this information, the user can perform three types of actions:

1. Redefine the notion of a "good" solution. This is done by adjusting the values of a 3-dimensional goal vector, defining a goal value for each of the three criteria optimized. A preference relation between solutions which incorporates the goal vector, is the key component of the rank-based selection mechanism used in [2]. Hence, by adjusting the goal vector, the user can directly affect the sampling of the search space and can focus the search, or "zoom in", on the region of interest.
2. A simple hillclimbing algorithm can be executed on a selected solution. A desired direction in the cost space of the hillclimbing process can be specified. For example, the user can specify that the hillclimber should attempt to improve as much as possible wrt. the second optimization criterion, allowing a cost increase wrt. the first and third criterion of at most 5 and 10%, respectively.
3. The control parameters of the GA, such as population size and mutation rate can be adjusted.

To evaluate whether search performance can be improved by interactively controlling the optimization process, the following experiments have been performed. For each of five example circuits, the algorithm was executed 10 times non-interactively, using a 1 CPU-hour time limit per run. A single interactive run was then performed for each circuit, defining the time limit to be 1 hour, wall-clock time, i.e., including the time spent interacting with the tool.As a baseline comparison, a random walk was also executed 10 times per circuit, using a 5 CPU-hour time limit per run.

The obtained set qualities were evaluated using the measure described in Section 2 and results are shown in Table 3.

Table 3: Each entry for the non-interactive executions and the random walk (RW) is the average value obtained from 10 runs followed by the standard deviation in brackets. The number of blocks for each circuit is indicative of the search space size.

Circuit	no. of blocks	set quality		
		interactive	non-interactive	RW
xeroxF	10	0.572	0.741(0.073)	0.888(0.045)
hpF	11	0.605	0.638(0.033)	0.822(0.029)
spertF	20	0.096	1.886(0.640)	2.178(0.010)
ami33F	33	0.690	0.759(0.058)	1.152(0.048)
ami49F	49	0.641	0.676(0.093)	1.197(0.052)

As can be seen from the table, the output sets obtained non-interactively in 1 hour are always significantly better than those obtained by the random walk in 5 hours. But more interestingly, all of the five sample executions in interactive mode yield better results than the average non-interactive execution. Furthermore, the number of circuit evaluations performed in interactive mode averages only about 78% of that of the non-interactive mode because of the idling processor during user-interaction. Hence, user-interaction significantly improves the efficiency of the search process, yielding a better result quality while performing less computational work.

The user's actions implicitly embody search control strategies and in the experiments described above, these strategies yielded high performance. This information can be used in two ways: either the knowledge can be modeled using various rule based techniques, i.e. a fuzzy system, or this knowledge can be used to identify the most significant actions. Although in theory the second use is covered by the first, in practice, robust general techniques for rule-extraction still do not exist.

To explore the complimentary use of both the information gained from the interactive use and the meta-level technique described in Section 4.1, we focus on the hill climber invoca

tion strategy. Further investigation, using the interactive tool, revealed that use of the hillclimber, referred to as action 2, accounted for most of the improvement reported in Table 2.

A manually constructed hillclimber invocation strategy yield high performance, however not at levels obtained by interactive use of the tool using all three types of actions. This simple strategy invoked the hillclimber to perform 200 steps at regular intervals. This disparity in performance between the interactive runs and the simple strategy can be attributed to deficiencies in the simple hillclimber invocation strategy in either choosing the right solution to perform hillclimbing on or even when to invoke hillclimbing.

We proceeded to use the meta-level optimization approach to improve the hillclimber invocation strategy. In these experiments, we used a fuzzy system to model the strategy and a simple genetic algorithm to find a high performance strategy. Inputs to the fuzzy system were current set quality, change in set quality, and time left. Output of the system was a binary decision variable. In future experiments, the output will represent the number of hillclimbing steps. The performance of the hybrid search algorithm was measured by examining the final set quality value after a fixed amount of time (1800 cpu seconds). The search is actually performed three times and the average of the independent runs is used by the meta-level search algorithm to determine performance. Table 4 reports average set quality values and standard deviations for a random walk, simple hillclimber invocation strategy, and an automatically constructed fuzzy invocation strategy. (Note that xeroxT is not identical to xeroxF in Table 3).

Table 4: Comparison of random walk, a simple hillclimber invocation strategy, and an automatically constructed fuzzy hillclimber invocation strategy. Entries represent averages over independent 10 runs on circuit xeroxT.

Algorithm	$\bar{q}$	q_σ
Random Walk	0.1838	0.0048
Simple Hillclimbing	0.1563	0.0173
Fuzzy Hillclimbing	0.1553	0.0203

5 Summary

In this paper, we presented a fuzzy/genetic multiobjective optimization algorithm and its application to component placement. Unlike single objective optimization algorithms, the output is a set of solutions rather than a single solution. In our proposed scheme, a fuzzy system is used to embed genetic algorithm control strategies to effect the quality of the output set.

Within this framework, we presented two techniques for acquiring search algorithm control strategies: meta-level optimization and user modeling techniques. We have obtained high performance search control strategies using these techniques both separately and cooperatively. When used in combination with the meta-level optimization, the user modeling is a key element because it can lead to a significant reduction in the space of control strategies, thereby increasing the feasibility of meta-level optimization.

Acknowledgments

This research is supported in part by NASA Grant NCC-2-275, ONR Grant N00014-96-1-0556, LLNL Grant LLL-B-291525, and BISC program of UC Berkeley. The authors would also like to thank Prof. David Wessel and the Center for New Music and Audio Technologies at UC Berkeley for use of computing resources. Dr. Esbensen is supported mainly by the Danish Technical Research Council.

References and Related Publications

[1] 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.

[2] Esbensen, H., Kuh, E.S., "EXPLORER: An Interactive Floorplanner for Design Space Exploration," *Proc. of the European Design Automation Conference*, 1996 (to appear).

[3] Esbensen H., Kuh, E.S., "Design Space Exploration Using the Genetic Algorithm," *Proc. of the IEEE International Symposium on Circuits and Systems*, 1996, Vol. IV, pp. 500-503.

[4] 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.

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

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

[7] Lee, M. A., Esbensen, H., and Lemaitre, L. (1995) "The Design of Hybrid Fuzzy/Evolutionary Multiobjective Optimization Algorithms," Proc. of the 1995 IEEE/Nagoya University WWW on Fuzzy Logic and Neural Networks / Evolutionary Computation, pp. 118-125.

[8] Lee, M. A., and Esbensen, H., (1996) "Multiobjective Optimization using Adaptive Fuzzy/Evolutionary Algorithms," Proc. of the 1996 International Society for Computers and their Applications (ISCA'96), San Francisco, CA, pp. 67-70.

[9] 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.

[10] 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.

Reinforcement Learning in Fuzzy Dynamic Programming

Hamid Berenji
Intelligent Inference Systems Corporation
Computational Sciences Division
Moffett Field, CA 94035

0-7803-3645-3/96 $5.00