Genetic Algorithms and Genetic Programming: Combining Strengths in One Evolutionary Strategy*

M.-R. Akbarzadeh-T., Center for Autonomous Control Engineering, University of New Mexico†
E. Tunstel, Jet Propulsion Laboratory, California Institute of Technology
M. Jamshidi, Center for Autonomous Control Engineering, University of New Mexico

Abstract

Genetic Algorithms (GA) and Genetic Programs (GP) are two of the most widely used evolution strategies for parameter optimization of complex systems. GAs have shown a great deal of success where the representation space is a string of binary or real-valued numbers. At the same time, GP has demonstrated success with symbolic representation spaces and where structure among symbols is explored. This paper discusses weaknesses and strengths of GA and GP in search of a combined and more *evolved* optimization algorithm. This combination is especially attractive for problem domains with non-homogeneous parameters. In particular, a fuzzy logic membership function is represented by numerical strings, whereas rule-sets are represented by symbols and structural connectives. Two examples are provided which exhibit how GA and GP are each best utilized in optimizing robot performance in manipulating hazardous waste. The first example involves optimization of a fuzzy controller for a *flexible robot* using GA and the second example illustrates usage of GP in optimizing an intelligent navigation algorithm for a *mobile robot*. A novel strategy for combining GA and GP is presented.

1 Introduction

Robots used for hazardous material handling commonly operate in unstructured and noisy environments. These robots need to exhibit robustness, stability and fine manipulability which is often difficult to control by a human tele-operator and must be dealt with locally by an intelligent controller. Consequently, new and alternative methods to increase autonomy and intelligence of a robot controller are desirable. Combinations of fuzzy logic-genetic algorithms and fuzzy logic-genetic programs have already shown the ability to learn from past experiences and to develop a robust knowledge-base without human intervention. In this paper, we will discuss two examples of such *evolutionary* optimizations of fuzzy controllers as applied to a *flexible robot* and a *mobile robot*. We will then propose a novel method of combining their strengths in order to determine an improved and more *evolved* knowledge-base.

Genetic algorithms are optimization search routines modeled after nature's evolutionary process. Numerous simulations of evolutionary fuzzy controllers have established the utility of GAs in optimizing knowledge-bases for fuzzy controllers [1]. Most of these applications differ in their approach to design an interpretation function between the phenotype and the genotype, i.e. the coding of fuzzy rule sets and membership functions, definition of fitness functions and handling of evolutionary operations. Here, the GA encoding is an aggregate string of n-bit binary numbers. In Section 2, we demonstrate one instance where GAs are used to optimize parameters of membership functions of a fuzzy controller for a flexible link robot arm.

In Section 3, GP [2] is applied to the problem of learning/discovering rules for use in a fuzzy rule-based controller for autonomous mobile robot navigation. The GP paradigm computationally simulates the Darwinian evolution process using a different representation than the GA. It applies fitness-based selection and genetic operators to a population of parse trees (individuals). Each parse tree represents a computer program of a given programming language, and is a candidate solution to a particular problem. Programs are structured as hierarchical compositions of functions and arguments with various sizes and shapes. These individuals participate in an evolutionary process wherein the population evolves over time in response to selective pressure induced by the relative fitnesses of the individuals in a particular problem environment. For the purpose of evolving fuzzy rule-bases, the search space is contained in the set of all possible rule-sets that can be composed recursively from specified linguistic variables, fuzzy

*This work was supported in part by Waste Education and Research Consortium grant under award # WERC/NMSU/DOE Amd 35 and by NASA under contract # NCCW-0087.

†Center for Autonomous Control Engineering, EECE Building, University of New Mexico, Albuquerque, New Mexico, 87131, Email: akbazar@eece.unm.edu

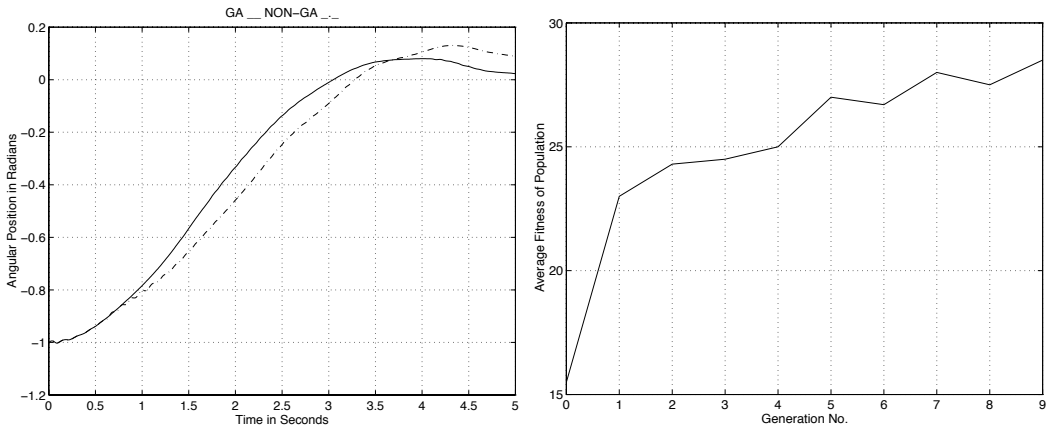

Figure 1: GA Simulation (a) Comparison of Time Responses (b) Plot of Average Fitness

sets, and fuzzy logic connectives which make up the generic fuzzy if-then rule. That is, the same fuzzy linguistic terms and operators that comprise the genes and chromosomes persists in the phenotype.

The proposed *co-evolution* of the two evolutionary strategies is motivated by the ability of GA to manipulate numbers (parameters of membership functions) and utility of GP for manipulating structures/rules (symbolically). The *co-evolution* strategy allows for parameters of membership functions and rule-bases to be evolved separately and in parallel by GA and GP. Details of the proposed method are discussed in Section 4.

2 Genetic Algorithms

An interpretation function is the transformation function between the representation (genotype) space and the evaluation (phenotype) space [3]. GA uses this transformation to encode several real or binary parameters in the domain of the problem to a single string of real or binary numbers in the GA domain. The coding in GA can be performed in various ways. One common method is as follows, the real variable a is first mapped to an n-bit signed binary string where the highest bit represents the sign. This way, the variable a can take on 2^n different values. Then the binary number is aggregated with other n-bit binary numbers to construct the genotype representation.

Once all optimization parameters are encoded in a string, GA creates a set of randomly chosen individuals to constitute the initial population. Future generations are determined by applying genetic operators such as *mutation*, *crossover*, and *reproduction* in search of improved solutions. Definitions and utility of each of these operators can be found in [4].

2.1 Application of GA to Control of a Flexible Robot

Due to the constraints in work environment, robots used in hazardous material handling are often slim when compared to their lengths and their expected payload. For example, the Department of Energy's underground storage tanks are 80 feet in diameter and 35 feet in depth, while the access hole where the robot must enter is only 12 inches [5]. In other instances, the robot is expected to lift heavy loads which result in bending its links. As a result, link flexibility becomes a significant factor in modeling and control of robot manipulators in hazardous waste handling applications. Ignoring link flexibility can result in poor performance and instability.

In this example, GA's parameter set is chosen to optimize parameters related to the input membership parameters. Other parameters in the knowledge-base are not allowed to vary. This will reduce simulation time and will still demonstrate the potential utility of GA. The following fitness function was used to evaluate various individuals within a population of potential solutions,

$$fitness = \int_{t_i}^{t_f} \frac{1}{e^2 + \gamma^2 + 1} dt, \tag{1}$$

where e represents the error in angular position and γ represents overshoot. The above fitness function was chosen to contain important parameters in a system's time response: rise time, steady state error, oscillations and overshoot. Consequently, the fitness function is inversely proportional to the error and overshoot in the system. Consequently,

a fitter individual is an individual with a lower overshoot and a lower overall error (shorter rise time) in its time response.

More details about the control architecture can be found in [6]. Here, a novel method is used which incorporates initial a-priori expert knowledge in creating initial populations, hence, increaing the average fitness of initial population and reducing convergence time. Figure 1.a shows the time response of the GA-optimized fuzzy controller compared to that of a non-GA fuzzy controller. A constant population size of 40 individuals was used. After only ten generations, the rise time is improved by 0.34 seconds (an 11% improvement), and the overshoot is reduced by .07 radians (a 54% improvement). Figure 1.b shows the average fitness of each generation. Mutation rate for generating the initial population was set at 0.3. The resulting initial population has an average fitness of 15.8, which converges to 28 after only 10 generations. Mutation rate for creating the following generations was set at 0.033. Probability of crossover was set to 0.6.

3 Genetic Programming

In GP, programs are structured as hierarchical compositions of functions (in a set F) and terminals (function arguments in a set T). These individuals participate in a probabilistic evolutionary process wherein the population evolves over time in response to selective pressure induced by the relative fitnesses of the individuals for solving the problem. The set, F, consists of components of the generic *if-then* rule and common fuzzy logic connectives, i.e. functions for antecedents, consequents, fuzzy intersection, rule inference, and fuzzy union [7]. The set, T, is made up of the input and output linguistic variables and the corresponding membership functions associated with the problem. A rule-base that could potentially evolve from F and T can be expressed as a tree data structure with symbolic elements of F occupying internal nodes, and symbolic elements of T as leaf nodes of the tree. This tree structure of symbolic elements is the main feature which distinguishes GP from GA which uses the numerical string representation. All rule-bases in the initial population are randomly created, but descendant populations are created primarily by reproduction and crossover operations tailored for manipulating rule-base tree structures [2]. GP cycles through the current population performing fitness evaluation and application of genetic operators to create a new population. The cycle repeats on a generation by generation basis until satisfaction of termination criteria (e.g. lack of improvement, maximum generation reached, etc). The GP result is the best-fit rule-base that appeared in any generation. The same fuzzy linguistic terms and operators that comprise the genes and chromosome persist in the phenotype. Thus, the use of GP allows direct manipulation of the actual linguistic rule representation of fuzzy rule-based systems. Furthermore, the dynamic variability of the representation allows for rule-bases of various sizes and different numbers of rules. This enhances population diversity which is important for the success of the GP system.

3.1 Application of GP to Mobile Robot Navigation

Mobile robots are useful for waste clean up and management tasks in environments where human presence is a substantial risk. In some cases teleoperated mobile robots are sufficient. However, in extremely hazardous environments a safer alternative might be autonomous mobile robots which could be used for waste site survey, inspection and, eventually, clean up and storage. In the following example, we show that GP can be used to design fuzzy coordination rule-bases for such robots.

We focus here on autonomous point-to-point navigation using simple behaviors — `go-to-xy` and `avoid-collision`. Each are fuzzy rule-bases which recommend wheel speeds given robot position, goal location, and current sensory data about range to nearby obstacles.

GP was used to evolve the behavior coordination rules based on a previously reported scheme [7]. Population sizes of 10–20 rule-bases were run for a number of generations ranging from 10–15. In GP, genetic diversity remains high even for very small populations due to the tree structure of individuals [2]. The simulated robot, modeled after a differentially steered $60cm$ mobile platform, senses range to obstacles using ultrasonic sensors. Pose information $(x\ y\ \theta)$ is computed from optical wheel encoders and its maximum speed was limited to $0.5m/s$. The simulated environment is a hypothetical indoor environment, e.g. a warehouse storing hazardous waste. For each rule-base, fitness was computed as an average error-based score for goal-directed navigation performance in simple obstacle situations. The mean performance of GP over five runs is shown, in the left half of Figure 2, as the progression of the population average fitness during the first ten generations. A trend towards higher fitness is evident. Evolved coordination behaviors were tested in the simulated environment (Figure 2) for sensor-based navigation from initial pose $(1\ 11 - \frac{\pi}{2})$ to a goal located at $(12, 1)$. The right half of the figure shows a navigation run successfully coordinated

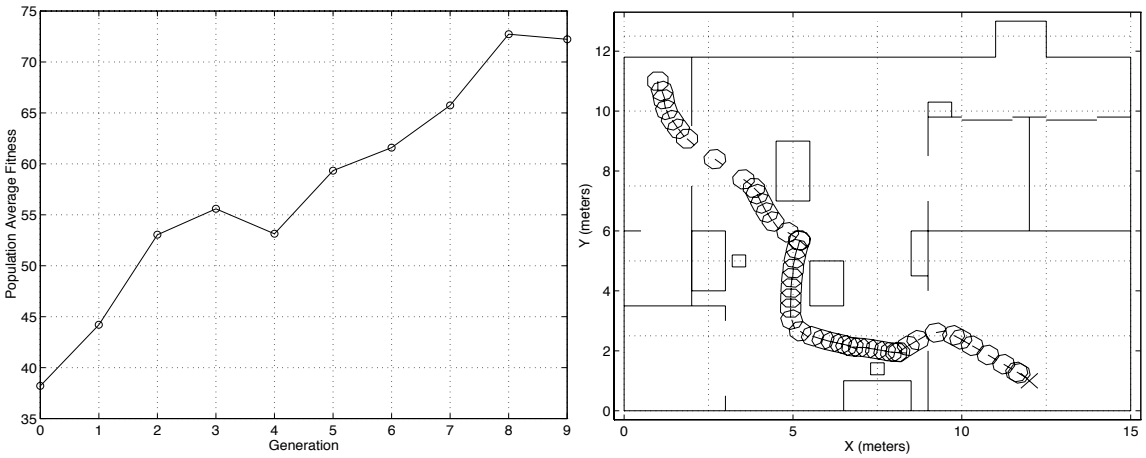

Figure 2: Performance of GP-evolved coordination.

by a GP-evolved behavior with a fitness of 71 out of a possible 100, and with 12 fuzzy rules. The best behavior discovered by GP over the five runs had a fitness of 86.5 and 11 rules.

4 GA & GP Co-Evolution

In this paper, *coevolution* is used to describe a search for a highly fit coupling between membership functions and rule-bases evolved by GA and GP, respectively. In this search, couples formed by individuals from the two coevolving populations have a joint fitness value and essentially make up one individual (a set of membership functions and a set of rules jointly constitute a knowledge base). In the conventional usage of "coevolution", only one evolutionary strategy is utilized and members of both sets are typically made up of the same domain of variables (i.e. binary or real numbers, tree structures, etc). Here, the two sets can be made up of several diverse domains, i.e. real numbers and structures, to implement coevolution as the evolution of GA and GP.

A simple way to co-evolve GA & GP is by assigning i^{th} individual in k^{th} population of GA (set of sets of membership functions, we will refer to this as set M from now on) to i^{th} individual in the k^{th} population of GP (set of sets of rules, we will refer to this as set R from now on). There are two problems in this approach. One is that an "inherently good" rule set may not yield a high fitness value if the corresponding membership functions are not "good". For example, a set of membership functions may be "inherently good" if it spans the universe of discourse, etc. And a rule set may be "inherently good" if it is a complete rule-base, rules are consistent, etc. If an "inherently good" rule set is coupled with a "bad" set of membership functions, we may never find the right match between rules and membership functions. Second is that while both the membership functions and rules affect the system performance, how does one determine a fitness value for each separately? If a coupled set of membership functions and rule-set performs good/bad, which is responsible? And how much?

It is, therefore, desirable to construct a method which allows for

- Crossover among individuals at the joining of individuals in M and individuals in R. In other words, we wish to allow i^{th} individual in set M to join j^{th} individual in set R. This allows good rule sets to join good membership functions, consequently, a fast convergence to a high fitness is expected.
- A fitness function which can separately evaluate a set of membership functions and a rule-set.

Let's define a "couple" as a joined membership function set and rule-set. A "partner" is an individual which can be either a set of membership functions or a rule-set. Now, co-evolution can be performed in two ways (or a combination of them).

- One-to-one bonding. Allows exchange of partners in the following fashion: Higher fit individuals of set M join with higher fit individuals of set R. The strength of the bonding between the two partners is determined by their joint fitness, W_{ij}, relative to the problem. At the beginning of each generation, the old generation is evaluated and regrouped. If a couple are doing well, they are allowed to survive. If a couple are not doing well, they will separate and may form a new bond. See Figure 3.

Figure 3: Demonstrating One-to-One and One-to-Many Bonding

- One-to-many bonding. One individual from set M can have several partners from set R and vice versa. The higher fit a couple are, the higher is their bonding. Higher fit individuals tend to marry more individuals of the other type. If a certain bond between two individuals is increased, other bonds involving those individuals decrease. If a couple is doing poorly, their bond is weakened and eventually broken. Couples with higher bonds survive in the next generation. See Figure 3.

Similarly, fitness can be evaluated in two ways (or a combination of them):

- One-to-One (with history): Fitness of an individual is an integral function of its performance with its previous partners.
- One-to-many (without history): Fitness of an individual is a summation of its current performance with its present partners.

New individuals are initially randomly joined with other individuals, then the process is continued. A new individual in M, which is created randomly or as a result of mutation [6], initially bonds with several individuals from set R. Consequently a fitness value is created for it. Lower fit individuals will be removed from the pool.

The above coevolution, therefore, has three fitness variables. First is the fitness of the couple (i.e. the bonding between the two), second is the fitness of the individuals in set M, and third is the fitness of the individuals in set R. At the end, the couple with the highest joint fitness (bond) will be selected.

5 Summary and Conclusions

Genetic algorithms have shown a great ability to manipulate numerical stings, whereas genetic programs have shown the ability to manipulate linguistic variables and structures directly from the problem and allow for populations of rule-bases of various sizes. In some applications, such as in knowledge base enhancement, the application domain is non-homogeneous, i.e. it is made up of both numerical strings as well as structures and linguistic variables. This paper briefly describes these characteristics and subsequently proposes a method to combine GA and GP. With this optimization method, the algorithm is expected to converge to near optimal solutions faster and the chances of achieving a globally optimal solution is increased.

References

[1] A. Homaifar & E. McCormick, "Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms," *Transactions on Fuzzy Systems*, Vol. 3, No. 2, pp.129, 1995.

[2] J. R. Koza, *Genetic Programming: On the programming of Computers by means of natural selection*, MIT Press, Cambridge, MA, 1992.

[3] L. C. Jain, "Evolutionary Computing in NN Design", Prepublished copy.

[4] D. E. Goldberg,"Genetic Algorithms in Search, Optimization and Machine Learning," *Addison-Wesley*, MA, NY, 1989.

[5] J.T. Fedemma, "Digital Filter Control of Remotely Operated Flexible Robotic Structures," *Proceedings of the 1993 American Control Conference*, 1993.

[6] M.-R. Akbarzadeh-T., M. Jamshidi, & Y.T. Kim, "Evolutionary Fuzzy Control of a Flexible Link," *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'97*, Albuquerque, New Mexico, 1997.

[7] E. Tunstel and T. Lippincott, "Genetic Programming of Fuzzy Coordination Behaviors for Mobile Robots", *International Symposium on Soft Computing for Industry, 2nd World Automation Congress*, Montpelier, France, pp. 647–652, 1996.