Recent Developments in Evolutionary and Genetic Algorithms: Theory and Applications

N. Chaiyaratana and A. M. S. Zalzala

University of Sheffield, UK

Email: rrg@sheffield.ac.uk

Abstract

This paper provides a review on current developments in genetic algorithms. The discussion includes theoretical aspects of genetic algorithms and genetic algorithm applications. Theoretical topics under review include genetic algorithm techniques, genetic operator technique, niching techniques, genetic drift, method of benchmarking genetic algorithm performances, measurement of difficulty level of a test-bed function, population genetics and developmental mechanism in genetic algorithms. Examples of genetic algorithm application in this review are pattern recognition, robotics, artificial life, expert system, electronic circuit design, cellular automata, and biological applications. While the paper covers many works on the theory and application of genetic algorithms, not much details are reported on genetic programming, parallel genetic algorithms, in addition to more advanced techniques e.g. micro-genetic algorithms and multiobjective optimisation.

1. Introduction

Genetic Algorithms have been intensively studied during the past three decades. Amounts of applications have benefited from the utilisation of genetic algorithms. Theoretical approaches on genetic algorithms have also helped researchers to understand the mechanisms of genetic algorithms. This paper is produced in an attempt to provide a brief description of current developments in genetic algorithms. The discussion can be divided roughly into two main parts: theoretical aspects of genetic algorithms and genetic algorithm applications.

2. Theoretical Aspects of Genetic Algorithms

2.1 Genetic Algorithm Techniques.

Two genetic algorithm techniques are discussed in this paper. They are coevolution and breeder genetic algorithm.

In co-operative coevolution genetic algorithm (CCGA), the population contains a number of species or sub-populations. Each species represents a variable or a part of the problem which is needed to be optimised. The combination of all species will leads to a complete solution which can be used as an index to fitness value. Potter and De Jong (1994) have demonstrated the use of this technique in multivariable function optimisation problems. Every function used in this case is a multimodal function. Each species is representing a single variable in the optimisation problem. The fitness of a species member is obtained by combining it with the current best individuals of the remaining species. CCGA outperform standard genetic algorithm in every case of function optimisation except in the case of function with high interdependencies between the function variables. Potter and De Jong (1994) have slightly modified CCGA to accommodate this problem. Rather than only combine an individual with the current best individuals from other species, each individual is also randomly combined with other individuals from the remaining species during fitness evaluation. The one between these two cases that gives a higher fitness value will be used as the offspring's fitness. CCGA has been used in a number of applications. For example, Potter et al. (1995) and De Jong and Potter (1995) have used CCGA to evolve sequential decision rules in a genetic algorithm based system called SAMUEL (Strategy Acquisition Method Using Empirical Learning). Different SAMUEL modules are used to represent different set of decision rules. Each SAMUEL module will be represented as a species in the population. These sets of rules are used to evolve behaviour of a robot. Another application which utilises CCGA is the design of a cascade neural network (Potter and De Jong, 1995). Since this type of network is self-growing in nature, CCGA has to introduce a new species to the population each time the network grows. A new set of connection weights which is introduced to the network is coded as a new species.

Breeder Genetic Algorithm (BGA) is first introduced by Mühlenbein and Schlierkamp-Voosen (1993). The major difference between simple genetic algorithm and BGA is the method of selection. Generally, truncation selection is used in BGA. In truncation selection, T % best individuals of the population are selected as parents. Chromosome in BGA can be coded using either binary or floating-point representation. A number of standard recombination (crossover) and mutation methods for BGA are discussed in details in Mühlenbein and Schlierkamp-Voosen (1993), Mühlenbein and Schlierkamp-Voosen (1995), Schlierkamp-Voosen and Mühlenbein (1994) and Schlierkamp-Voosen and Mühlenbein (1996). A comparative study on different recombination method used in BGA can be found in Voigt et al. (1995). Voigt et al. (1996) have shown that fitness distribution of the population in any generation of

Genetic Algorithms in Engineering Systems: Innovations and Applications, 2-4 September 1997, Conference Publication No. 446, © IEE, 1997

BGA using floating-point chromosome coding and uniform fuzzy gene pool recombination can be described by a gamma distribution. A gamma distribution appears to fit the simulation results better than a normal distribution in this case. It is shown that this conclusion is more obvious in the case of optimisation with small number of variables. BGA has been used to design an optimal Sigma-Pi neural network (Zhang and Mühlenbein, 1994). Sigma-Pi network has the structure similar to a multilayer perceptron in the sense that some of its units use weighted sum relations (Sigma). The difference is that a Sigma-Pi network also contains units which use weighted product relations (Pi).

2.2 Genetic Operator Technique.

Spears (1995) has introduced the use of combination between uniform crossover and two-point crossover in the same population. Each individual, coded in binary form, has an extra bit attached to its chromosome. This extra bit is used to indicate which kind of crossover the individual will use when it mates. This extra bit will not change by mutation operation. The tag bit of the offspring is depended upon the tag bit of its parents. If the parents have the same tag bits, the tag bit of the offspring will be the same as its parents. On the other hand, if the parents have different tag bits, the type of crossover and the tag bit of the offspring will be defined by using unbiased coin tossing method.

2.3 Niching Techniques.

Two niching techniques - simple sub-population scheme and deterministic crowding are included in this review.

In simple sub-population scheme, the population is divided into sub-populations, each individual in each sub-population can only perform mating with other individuals from the same sub-population (Spears, 1994). Each individual will be tagged or labelled to indicate which sub-population it belongs. In this case, evolution strategy used is sharing. The optimal solutions are treated as resources. Overcrowding on one particular optimal solution implies that the resource is overused. In this case the perceived fitness of that solution will decrease. On the other hand if a few individuals are concentrated on one solution, that resource is underused. The perceived fitness of that solution will increase. This is a modification to the internal structure of genetic algorithm. It is a change in the perception of genetic algorithm to the objective function, not the objective function itself.

Crowding method is a method for maintaining sub-populations in genetic algorithm at different niches in multimodal fitness landscape. In crowding method, a small set of parents is selected from the population. Each child will replace an individual from this set which is most, similar to itself. In this method, stochastic replacement errors prevent the algorithm from locating all niches. Mahfoud (1994a) has introduced a new variant of crowding method called deterministic crowding. In deterministic crowding, the children compete against their parents for inclusion in the new generation. Unlike general crowding method, all parents have to participate in the competition. In deterministic crowding, similarity between children and parents can be measured using either genotypic or phenotypic distance. Detail comparison between deterministic crowding and sharing, parallel hill-climbing and sequential niching can be found in Mahfoud (1995).

2.4 Genetic Drift.

Genetic drift is an important phenomenon in genetic algorithm search. Once the algorithm is converged, the size of original gene pool is reduced to the size of found solution(s) gene pool. This leads to genetic drift.

Mahfoud (1994b) has performed theoretical calculation and simulation in order to test his theory about genetic drift in sharing method. Sharing method is generally used for maintaining stable sub-populations when multiple niches are the aimed solutions. However, some sub-populations may disappear before all solutions are found. An expected time of sub-population loss is a desirable measurement. This will indicate the generation number which some fine tunings are needed to be done on the population in order to prevent premature genetic drift. Mahfoud (1994b) have shown that the expected time of sub-population loss can be expressed in the following equation:

$$\mu_L = \frac{1}{c} e^{n/c} \qquad (1)$$

where μ_L is the average expected time of subpopulation loss,

c is the number of sub-populations or classes

and n is the size population.

The simulation results by Mahfoud (1994b) follow this equation with 95% confidence level.

Asoh and Mühlenbein (1994b) have shown that for a genetic algorithm with no selection and mutation, using uniform crossover and each gene has only two alleles (e.g. A and a), the time at which the allele that has less distribution in the population disappears or the mean convergence time τ can be approximated by

$$\tau = 1.4N(0.5\log_e n + 1.0)^{1.1} \text{ for } p_A = 1/2$$
$$\tau = 1.0N(0.7\log_e n + 1.0)^{1.1} \text{ for } p_A = 3/4$$
$$\tau = 0.7N(0.8\log_e n + 1.0)^{1.2} \text{ for } p_A = 7/8 \qquad (2)$$

where N is the size of population,

n is the number of loci on chromosome

and p_A is the probability of existence of allele A in the initial population.

2.5 Method of Benchmarking Genetic Algorithm Performances.

Horn et al. (1994) have introduced a new way of determining performance of an optimisation algorithm. By considering the number of iterations required by an algorithm to cover a path length - distance in which an initial solution has to travel from the starting point to the goal, performances from different algorithms can be compared. For a problem of l-bit long and step size of one bit, path length can be formed with the length of

$$|P_l| = 3*2^{\lfloor (l-1)/2 \rfloor} - 1 \quad (3)$$

where $|P_l|$ is the path length

and l is the bit length of the problem.

Path length increases in proportion to $(\sqrt{2})^l$ and thus grows exponentially in l. Horn et al. (1994) have compared performances of genetic algorithm with no mutation, steepest ascent hill-climbing and next ascent hill-climbing by utilising this method. Genetic algorithm is found to be the algorithm which covers the path length with the minimum number of iterations.

2.6 Measurement of Difficulty Level of a Test-bed Function.

Kargupta (1995) has utilised the idea of signal-to-noise in genetic algorithm analysis. The higher the noise level, the harder for genetic algorithm to solve the problem. In this case, noise in genetic algorithm can be defined as a combination between the variance within partition and the covariance between different partitions. The term "partition" in this case is referring to a group of schemata which has the same fix-bit patterns. For example, partition f** in 3-bit string will contain schemata 1** and 0**. This partitioning leads to two types of dependency between partitions: intra-partition dependency and inter-partition dependency. When a partition contains more than one fix position, schemata which belong to that partition will also belong to lower order partitions. For example, partition ff* contains schemata 00*, 01*, 10* and 11*. In the same time, these schemata will also be part of partition f** and *f*. This kind of dependency is called intra-partition dependency. On the other hand, certain strings will belong to more than one partition. For example, string 11 will belong to partition f* and *f. This leads to inter-partition dependency. The covariance between different partitions will depend on the levels of inter-partition and intra-partition dependency.

Jones and Forrest (1995) have introduced a technique which is based on a measurement of correlation between the fitness of an individual and its distance to the global optimum. Hence this technique is called fitness distance correlation (FDC) method. The distance between the position of an individual and the global optimum in the search space can be measured in terms of Hamming distance. For a given set of n individuals with their fitness value f_i and their Hamming distance d_i, the correlation coefficient r is given by

$$r = \frac{C_{FD}}{S_F S_D} \quad (4)$$

where $C_{FD} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \overline{f})(d_i - \overline{d})$ is the covariance of fitness and distance

and $S_F, S_D, \overline{f}, \overline{d}$ are the standard deviations and means of the fitness set and distance set, respectively.

The use of FDC in conjunction with a scatter plot between fitness and distance can be used to measure the level of GA difficulty of the test-bed function. For a maximisation problem, the fitness value is expected to be increasing as the distance to optimum solution is reduced. This leads to the coefficient r of -1.0 for an ideal fitness function. On the other hand, for a minimisation, genetic algorithm should work well if the fitness value is decreasing as the distance to optimum solution is reduced. The coefficient r of 1.0 is expected for an ideal fitness function. By measuring a deviation from the ideal value of coefficient r, the level of difficulty can be determined.

2.7 Population Genetics.

In population genetics, the relationship between response to selection and selection deferential is given by

$$R(t) = b_t S(t) \quad (5)$$

where $R(t) = M(t+1) - M(t)$,

$S(t) = M_s(t+1) - M(t)$,

$M(t)$ is the mean fitness value of population in generation t,

$M_s(t)$ is the mean fitness value of selected parents in generation t,

and b_t is the realised heritability.

It is important in population genetics to be able to estimate the value of realised heritability. Asoh and Mühlenbein (1994a) have shown a method of estimating the value of b_t. It is shown that the value of b_t can be computed from genetic variance of the population. Variance of the fitness function f can be decomposed into

$$Var_p = V_1 + V_2 + \ldots + V_{n-1} + V_n \quad (6)$$

where n is the number of loci on chromosome. Details of each term can be found in Asoh and Mühlenbein (1994a). With the use of random mating and uniform crossover, the covariance can be estimated by

$$Cov_{po} = \sum_{k=1}^{n} \frac{1}{2^k} V_k \quad (7)$$

This estimated value of covariance is approximately equal to the value of realised heritability.

2.8 Developmental Mechanism in Genetic Algorithms.

Hart et al. (1994) have stressed the importance of developmental mechanisms in genetic algorithms. The

term "development" is used in the context of process by which genotypes are transformed into phenotypes. Developmental mechanisms in genetic algorithms contain two major parts: maturation and learning. Maturation is the process by which a genotype is mapped into a phenotype and learning refers to local search such as hill-climbing. A number of advantages are gained from the use of maturation and local search. Hart et al. (1994) have explained these advantages in three different issues: fitness transformations, time complexity and evolutionary bias.

3. Genetic Algorithm Applications

3.1 Pattern Recognition Applications.

In any pattern recognition application, a minimum feature set which yields a maximum classification accuracy is desirable. Imam and Vafaie (1994) and Vafaie and Imam (1994) have performed a comparative study between the use of genetic algorithm and important score method for reducing the number of feature used in pattern recognition application via the use of AQ15 learning system. Genetic algorithm is used to search the space of all possible subsets of the complete set which includes all features. The feature set is coded as a binary string which each bit represents presence or absence of a particular feature. The classification accuracy will be used as fitness. Genetic algorithm gives a better classification accuracy than important score method with the expenses of a higher number of iterations required and a higher number of features required in the reduced feature set. The same technique in using genetic algorithm to find an optimal feature set is also applied to a hybrid GA-ID3 system by Bala et al. (1995). Genetic algorithm can also be used to select a new feature set which combines original features and new features constructed by applying arithmetic operations (such as +, -, *, /) to the original features (Vafaie and De Jong, 1995).

3.2 Robotics and Artificial Life Applications.

Menczer and Belew (1994) have used steady-state genetic algorithm to evolve sensory characteristics of artificial organism in an environment with controlled complexity. The environment model used is called a latent energy environment (LEE). Feed-forward neural networks are used to simulate organisms. Two types of sensors are interested in this study: contact and ambient. Contact sensors are presented in the organism which is required to learn avoidance tasks. Reinforcement learning is used to train motor actions which are the outputs from neural network. Ambient sensors are presented in organism which approaching task is required. Back-propagation learning is used to train sensory prediction outputs of the network. Any changes in motor characteristic of this type of organism can only be achieved via evolution. Steady-state genetic algorithm is used in this study as follows. Each individual, represented by neural network, must acquire energy from atoms in the environment beyond a fixed threshold before it can asexually reproduce. If the energy level within an individual is lower than a threshold, that individual will die. Chromosome of each individual contains two parts, one in floating-point format, the other in binary format. The connection weights in neural network are coded into the floating-point section of chromosome. Mutation is done by randomly added uniformly distributed noise to the chromosome. The types of atoms the sensors sensed are coded into the binary part. Bit-flip mutation is used in this section.

Grefenstette and Schultz (1994) have used genetic algorithm to evolve rule sets in SAMUEL system. The evolving rule sets contain the rules for collision avoidance and energy resources finding. The SAMUEL system consists of two modules: execution system module and off-line system module. Execution system module contains the actual robot and environment. Off-line system module consists of robot simulation module and genetic algorithm module for rule sets evaluation. The initial population is a heterogeneous population which contains a variety of rule sets and their variants which are automatically generated. After the off-line learning is accomplished, the rule sets are tested on the actual system. Ramsey and Grefenstette (1994) have modified this learning system to include real-time modification to the robot model. This learning strategy is called cased-based anytime learning.

Jakobi (1995) has introduced a new encoding scheme in genetic algorithm. During evolution process, a multi-cellular organism is created and is finally transformed into a recurrent network. This neural network can be used as a robot controller. The neural network robot controller can be used to control a robot to perform corridor following task and object avoidance task. Within each cell there is a genomic regulatory network (GRN). GRN is composed of a number of units, each unit contains a single string genome. One genome is responsible for the production of one protein. Protein which is produced by one unit regulates other genes in the different units. Proteins within each cell are divided into different classes which effect the gross behaviour of the cell. Signal proteins diffuse out of one cell and into another, resulting in an interaction between cells. Initially a single cell is placed in a controlled environment which contains a number of predefined extra-cellular signal protein sources. This leads to cellular developments including cell division and cell movement. Interaction between cells will eventually lead to cell differentiation. Once a cell is differentiate, a number of densities are grown out of each cell. When a dendrite from one cell contacts another cell, a synaptic connection is established. After every cell has been fully developed, thresholds and weights are assigned to each

cell and dendrite, respectively. This leads to the formation of a recurrent neural network.

3.3 Expert System Applications.

Roache et al. (1995) have explained how to use a genetic algorithm to validate an expert system. The objective of testing an expert system is to find input combinations which will cause the expert system to give inappropriate responses. Changes can then be made to the expert system. It is exhaustive to test an expert system with all possible input combinations. Genetic algorithm can be used to generate test inputs to the expert system. This results in an optimal number of test cases which yield a good coverage of all possible input combinations to the expert system. In this case, an expert system is used to control electricity output of a power station. Plant parameters are used to code chromosomes. The fitness function used is based on the plant heat rate. Plant heat rate is the number of British Thermal Units (BTUs) required to produce one Kilowatt-hour of electricity. A combination of plant inputs and environment inputs which makes the expert system to respond inappropriately and increases the plant heat rate will result in high fitness.

3.4 Electronic and Electrical Applications.

Koza et al. (1996d) have introduced a method incorporating genetic programming in electronic circuit design. Both topology and components' value in a circuit will be automatically determined by a pool of evolving programs. These evolving programs will undergo a genetic programming evolution which includes reproduction, crossover and mutation. In order to apply genetic programming to a circuit design, electrical circuits must be mapped to program trees. Each program tree contains two main types of function: connection-modifying functions which modify the topology of the circuit and component-creating functions which insert electronic components into locations within the topology of the circuit. The examples of circuit designed by using genetic programming are crossover (woofer and tweeter) filter (Koza et al., 1996d), low pass filter (Koza et al., 1996b), double band-pass filter (Koza et al., 1996c), amplifier circuit (Koza et al., 1996a) and food-forging controller for simulating behaviour of a lizard (Koza et al., 1996e).

3.5 Cellular Automata Applications.

Cellular automata are an abstract way of analysing the simultaneous execution of local rules. A cellular space is a uniform array of cells arranged in some forms of topology and dimension. For a cellular automaton (CA), each cell in the cellular space contains an identical automaton. The next state of each automaton is defined by a function of its current state and the current state of other automata in a predefined neighbourhood. Andre et al. (1996a,b) have used genetic programming with automatically defined functions to produce state-transition rule of linear cellular automaton for solving majority classification problem. The cellular space consists of 149 automaton in linear arrangement. Each automaton has either state 0 or state 1 at any given time. The next state of each automaton depends on its own current state and the states of its six neighbour automata, three to the left and three to the right. In this case, the program tree contains result-producing branch and automatically defined functions. The resulting state-transition rule can solve the majority classification problem with a higher accuracy than the Gacs-Kurdyumov-Levin (GKL) rule and other known human-written rules.

Das et al. (1994) have also studied the behaviour of cellular automata via the use of linear cellular automaton. Unlike the work by Andre et al. (1996a,b), Das et al. (1994) use genetic algorithm to evolve the state-transition rules. Chromosome of each individual represents the output bits from all rules in a rule set in lexicographic order of neighbourhood configuration. Since in this case, each rule output depends on the states of seven cells, each individual will have chromosome of length 128 (2^7). Das et al. (1994) have applied this technique on majority classification task. Although their results are very promising, the resulting rule has a lower classification accuracy than that of GKL rule. Das et al. (1995) have utilised the same technique on synchronisation task. Further analysis of majority classification task and synchronisation task using cellular automata and genetic algorithm can be found in Hordijk et al. (1996).

3.6 Applications in Biology and Medicine.

Hightower et al. (1995) have used genetic algorithm to model an evolution in an immune system. Chromosome of each individual represents libraries of genetic material in the immune system. These genetic materials are used to construct antibodies which are responsible for recognising antigens. Unlike other applications using genetic algorithm, phenotype of an individual which is represented by antibodies produced is not a complete mapping from every gene in the chromosome. Chromosome is divided into four libraries of genetic material. Each library contains eight elements. An antibody is produced by combing one element from every library. This study has demonstrated that genetic algorithm is capable of improving fitness of the population even only partial information about each individual is given to the algorithm during each generation. Hightower et al. (1996) have extended this study to include Baldwin effect in evolving immune system.

Parsons et al. (1995) have introduced a method involving the use of genetic algorithm in gene

sequencing. The simulation is based on an actual sequencing method called short-gun sequencing method. Firstly, DNA is replicated many times and then individual strands of the double helix are broken randomly into fragments. These fragments (from both strands) are then used in pair-wise relationship computation process. Each pair of fragments is compared and their similarities are determined, resulting in an overlap strength. All possible orientations and alignments are tried to maximise the overlap strength. The overlap strength will be the key to fitness value used in genetic algorithm. All fragments are then totally ordered. Different possible permutations of fragment order will be represented by individuals in genetic algorithms. Once the ordering process starts, a number of fragments will form a continuous layout called contig. Initially, there will be many short length contigs presented in the process. Toward the end of the process, contigs should increase in their length and reduce their number. Finally, the end sequence or the consensus sequence should contain only one contig.

4. Conclusions

This paper summarises a number of current developments in genetic algorithms. It includes both theoretical aspects of genetic algorithms and some potential applications which incorporate the use of genetic algorithms.

References

Andre, D., Bennett III, F. H. and Koza, J. R. (1996a). Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference* (pp. 3-11). Cambridge, MA: MIT Press.

Andre, D., Bennett III, F. H. and Koza, J. R. (1996b). Evolution of intricate long-distance communication signals in cellular automata using genetic programming. *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA: MIT Press.

Asoh, H. and Mühlenbein, H. (1994a). Estimating the heritability by decomposing the genetic variance. In Y. Davidor, H. P. Schwefel and R. Männer (Eds.), *Lecture Notes in Computer Science 866 - Parallel Problem Solving from Nature - PPSNIII, International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature* (pp. 98-107). Berlin, Germany: Springer-Verlag.

Asoh, H. and Mühlenbein, H. (1994b). On the mean convergence time of evolutionary algorithms without selection and mutation. In Y. Davidor, H. P. Schwefel and R. Männer (Eds.), *Lecture Notes in Computer Science 866 - Parallel Problem Solving from Nature - PPSNIII, International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature* (pp. 88-97). Berlin, Germany: Springer-Verlag.

Bala, J., Huang, J., Vafaie, H., De Jong, K. A. and Wechsler, H. (1995). Hybrid learning using genetic algorithms and decision trees for pattern classification. *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Quebec, Canada.

Das, R., Mitchell, M. and Crutchfield, J. P. (1994). A genetic algorithm discovers particle-based computation in cellular automata. In Y. Davidor, H. P. Schwefel and R. Männer (Eds.), *Lecture Notes in Computer Science 866 - Parallel Problem Solving from Nature - PPSNIII, International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature* (pp. 344-353). Berlin, Germany: Springer-Verlag.

Das, R., Crutchfield, J. P., Mitchell, M. and Hanson, J. E. (1995). Evolving globally synchronized cellular automata. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

De Jong, K. A. and Potter M. A. (1995). Evolving complex structures via cooperative coevolution. *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, San Diego, CA.

Grefenstette, J. J. and Schultz, A. (1994). An evolutionary approach to learning in robots. *Machine Learning Workshop on Robot Learning*, New Brunswick, NJ.

Hart, W. E. (1994). The role of development in genetic algorithms. In D. Whitley and M. Vose (Eds.), *Foundations of Genetic Algorithms 3*. Morgan Kaufmann.

Hightower, R. R., Forrest, S. and Parelson, A. S. (1995). The evolution of emergent organization in immune system gene libraries. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 344-350). San Francisco, CA: Morgan Kaufmann.

Hightower, R. R., Forrest, S. and Parelson, A. S. (1996). The Baldwin effect in the immune system: learning by somatic hypermutation. In R. K. Belew and M. Mitchell (Eds.), *Adaptive Individuals in Evolving Populations* (pp. 159-167). Reading, MA: Addison-Wesley.

Hordijk, W., Crutchfield, J. P. and Mitchell, M. (1996). Embedded-particle computation in evolved cellular automata. *Physics and Computation'96*.

Horn, J. Goldberg, D. E. and Deb, K. (1994). Long path problems. In Y. Davidor, H. P. Schwefel and R. Männer (Eds.), *Lecture Notes in Computer Science 866 - Parallel Problem Solving from Nature -*

PPSNIII, International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature (pp. 149-158). Berlin, Germany: Springer-Verlag.

Imam, I. F. and Vafaie, H. (1994). An empirical comparison between global and greedy-like search for feature selection. *Proceedings of the Florida AI Research Symposium, FLAIRS-94*, FL, 66-70.

Jakobi, N. (1995). Harnessing morphogenesis. *International Conference on Information Processing in Cells and Tissues*, Liverpool, UK.

Jones T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 184-192). San Francisco, CA: Morgan Kaufmann.

Kargupta, H. (1995). Signal-to-noise, crosstalk and long range problem difficulty in genetic algorithms. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 193-200). San Francisco, CA: Morgan Kaufmann.

Koza, J. R., Andre, D., Bennett III, F. H. and Keane, M. A. (1996a). Evolution of a low-distortion, low-bias 60 decibel op amp with good frequency generalization using genetic programming. In J. R. Koza (Ed.), *Late Breaking Papers at the Genetic Programming 1996 Conference* (pp. 94-100). Stanford, CA: Stanford University Bookstore.

Koza, J. R., Andre, D., Bennett III, F. H. and Keane, M. A. (1996b). Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference* (pp. 132-140). Cambridge, MA: MIT Press.

Koza, J. R., Bennett III, F. H., Andre, D. and Keane, M. A. (1996c). Automated WYWIWYG design for both topology and component values of electrical circuits using genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference* (pp. 123-131). Cambridge, MA: MIT Press.

Koza, J. R., Bennett III, F. H., Andre, D. and Keane, M. A. (1996d). Four problems for which a computer program evolved by genetic programming is competitive with human performance. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 1-10.

Koza, J. R., Bennett III, F. H., Andre, D. and Keane, M. A. (1996e). Toward evolution of electronic animals using genetic programming. *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA: MIT Press.

Mahfoud, S. W. (1994a). Crossover interactions among niches. *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Orlando, FL, 188-193.

Mahfoud, S. W. (1994b). Genetic drift in sharing method. *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Orlando, FL, 67-72.

Mahfoud, S. W. (1995). A comparison of parallel and sequential niching methods. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 136-143). San Francisco, CA: Morgan Kaufmann.

Menczer, F. and Belew, R. K. (1994). Evolving sensors in environments of controlled complexity. In R. A. Brooks and P. Maes (Eds.), *Artificial Life IV: Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems* (pp. 210-221). Cambridge, MA: MIT Press.

Mühlenbein, H. and Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm: I. continuous parameter optimization. *Evolutionary Computation, 1*(1), 25-49.

Mühlenbein, H. and Schlierkamp-Voosen, D. (1995). Analysis of selection, mutation and recombination in genetic algorithms. In W. Banzhaf and F. H. Eeckman (Eds.), *Lecture Notes in Computer Science 899 - Evolution and Biocomputation, Computational Models of Evolution* (pp. 142-168). Berlin, Germany: Springer-Verlag.

Parsons, R. J., Forrest, S. and Burks, C. (1995). Genetic operators for the DNA fragment-assembly problem. *Machine Learning, 21*(1/2), 11-33.

Potter, M. A. and De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. In Y. Davidor, H. P. Schwefel and R. Männer (Eds.), *Lecture Notes in Computer Science 866 - Parallel Problem Solving from Nature - PPSNIII, International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature* (pp. 249-257). Berlin, Germany: Springer-Verlag.

Potter, M. A. and De Jong, K. A. (1995) Evolving neural networks with collaborative species. *Proceedings of the 1995 Summer Computer Simulation Conference*, Ottawa, Ontario, Canada, 340-345.

Potter, M. A., De Jong, K. A. and Grefenstette, J. J. (1995). A coevolutionary approach to learning sequential decision rules. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 366-372). San Francisco, CA: Morgan Kaufmann.

Ramsey, C. L. and Grefenstette, J. J. (1994). Cased-based anytime learning. In D. W. Aha (Ed.), *Cased-Based Reasoning: Papers from the 1994 Workshop.* Menlo Park, CA: AAAI Press.

Roache, E. A., Hickok, K. A., Loje, K. F., Hunt, M. W. and Grefenstette, J. J. (1995). Genetic algorithms for

expert system validation. *Proceedings of 1995 Western MultiConference*, Las Vegas, NV.

Schlierkamp-Voosen, D. and Mühlenbein, H. (1994). Strategy adaptation by competing subpopulations. In Y. Davidor, H. P. Schwefel and R. Männer (Eds.), *Lecture Notes in Computer Science 866 - Parallel Problem Solving from Nature - PPSNIII, International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature* (pp. 199-208). Berlin, Germany: Springer-Verlag.

Schlierkamp-Voosen, D. and Mühlenbein, H. (1996). Adaptation of population sizes by competing Subpopulations. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 330-335.

Spears, W. M. (1994). Simple subpopulation schemes. *Proceedings of the Third Annual Conference on Evolutionary Programming*, San Diego, CA, 296-307.

Spears, W. M. (1995). Adapting crossover in evolutionary algorithms. *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, San Diego, CA.

Vafaie, H. and De Jong, K. A. (1995). Genetic algorithms as a tool for restructuring feature space representations. *Proceedings of the Seventh International Conference on Tools with AI*, Herndon, VA, 8-11.

Vafaie, H. and Imam, I. F. (1994). Feature selection methods: genetic algorithms vs. greedy-like search. *Proceedings of the International Conference on Fuzzy and Intelligent Control Systems*, Louisville, KY.

Voigt, H.-M., Mühlenbein, H. and Cvetkovic´, D. (1995). Fuzzy recombination for the breeder genetic algorithm. In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms* (pp. 104-111). San Francisco, CA: Morgan Kaufmann.

Voigt, H.-M., Mühlenbein, H. and Schlierkamp-Voosen, D. (1996). The response to selection equation for skew fitness distribution. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 820-825.

Zhang, B.-T. and Mühlenbein, H. (1994). Synthesis of Sigma-Pi neural networks by the breeder genetic programming. *Proceedings of the First IEEE International Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Orlando, FL, 318-323.