Structure Fitness Sharing (SFS) for Evolutionary Design by Genetic Programming

Jianjun Hu[+], Kisung Seo[+], Shaobo Li*, Zhun Fan[+], Ronald C. Rosenberg[Ψ], Erik D. Goodman[+]

hujianju@cse.msu.edu
[+]Genetic Algorithms Research and Applications Group (GARAGe)
Michigan State University
East Lansing, MI, 48824

*CAD Institute
Guizhou University of Technology
Guiyang, 550003, China

[Ψ]Department of Mechanical Engineering
Michigan State University
East Lansing, MI, 48824

Abstract

Balanced structure and parameter search is critical to evolutionary design with genetic programming (GP). Structure Fitness Sharing (SFS), based on a structure labeling technique, is proposed to maintain the structural diversity of a population and combat premature convergence of structures. SFS achieves balanced structure and parameter search by applying fitness sharing to each unique structure in a population, to prevent takeover by the best structure and thereby maintain the diversity of both structures and parameters simultaneously. SFS does not require definition of a distance metric among structures, and is thus more universal and efficient than other fitness sharing methods for GP. The effectiveness of SFS is demonstrated on a real-world bond-graph-based analog circuit synthesis problem.

1 INTRODUCTION

Genetic programming has been applied successfully to a rich variety of problems such as machine code evolution (Nordin, 1997), quantum algorithm design (Spector, 1999), cellular automaton rule discovery, and soccer-playing program evolution (Andre 1999). GP has been particularly effectively used as an efficient Darwinian Invention Machine that enabled Koza *et al.* to achieve human-competitive results in analog circuit design and in the transmembrane segment identification problem (Koza 1999). Indeed, one of GP's most significant features is the ability to simultaneously evolve both a structure and its parameters, opening up promising applications in many real-world engineering design problems and in neural network design. In all of these problems, the objective is to search for an open-ended structure, together with its related parameters, to achieve several desired goals. Genetic programming – especially evolutionary design by genetic programming – is recognized as making a high demand on computational resources (Koza, 1999). To some extent, this demand can be traced to the premature convergence problem, especially convergence of the structures in a GP population; it can be ameliorated using diversity-maintenance techniques for the population. Based on an analysis of the weak causality of GP, the new concept of Structure Fitness Sharing (SFS), based on a structure labeling technique, is proposed to achieve balanced structure and parameter search by maintaining the diversity of both structures and parameters at all times. This method does not require definition of a distance metric, and is thus very efficient compared to other fitness sharing methods. Its effectiveness is demonstrated on a real-world bond-graph-based analog circuit synthesis problem using GP.

2 THE DIVERSITY PROBLEM IN EVOLUTIONARY DESIGN BY GP

2.1 CATEGORIES OF EVOLUTIONARY DESIGN PROBLEMS

Most evolutionary design problems can be classified into one of three types:

TYPE I: Fixed structure with fixed number of parameters.

These problems are essentially parameter optimization problems – the task is to optimize the parameters of a given structure. Genetic algorithms, simulated annealing, evolutionary programming, evolution strategies, and even gradient-based optimization techniques are often used here.

TYPE II: Variable structure with no parameters.

This type includes problems such as algorithm design, program induction and logic design, in which only structure search is needed. These problems are well suited for GP, which intrinsically manipulates the program structure, often represented as a tree. Of course, some of these problems can be solved with

genetic algorithms, simulated annealing, and other techniques, by using a somewhat indirect representation of the structure.

TYPE III: Variable structure with variable number of parameters (of course, there can also be variable structure with a fixed number of parameters, but that is not the typical case – number of parameters usually varies with the size of the structure).

Many of the most interesting evolutionary design problems belong to this third category, in which a structure is sought within a topologically open-ended space, but the fitness of a structure can often only be evaluated after parameters are assigned to key variables associated with the structures evolved. Since the structure is varied during the search process, the number of parameters and their semantics change frequently. Such problems include analog circuit design (Koza, 1999), mechanical system design (Fonseca, 1993), and neural network design (Oliker, 1992). Although a GA with a variable-length representation can be used here, GP, with its outstanding capability to search simultaneously for a good structure and for appropriate parameters, distinguishes itself as the most important tool for this kind of open-ended design problem.

2.2 PREMATURE CONVERGENCE AND DIVERSITY IN THREE TYPES OF EVOLUTONARY DESIGN PROBLEMS

TYPE I problems are often described as parameter optimization problems, readily addressable by GA. Premature convergence in GA has been well studied. Common diversity maintenance techniques include crowding (DeJong, 1975), deterministic crowding (Mahfoud, 1992), and fitness sharing (Goldberg, 1989). The fitness derating method (Beasley, 1993), a multi-objective method, employs fitness sharing in a popular and effective way.

The premature convergence problem when GP is applied to TYPE II problems has also been well studied. Most of the resulting methods are derived from GA, but with some specific consideration of the GP context. In multi-objective genetic programming, Rodriguez (Rodriguez-Vazquez, 1997) uses the MOGA approach with fitness sharing being performed in the fitness space, and extends it to genetic programming. Though easier to implement, it remains an open question whether diversity of fitness values is generally a true indicator of the diversity of a population – a measure which should actually be based on the parameter space. DeJong *et al.* (DeJong, 2001) use the multi-objective method to explicitly promote diversity by adding a diversity objective. In their method, a distance measure defined as follows is used in the diversity objective. The distance between two corresponding nodes is zero if they are identical and one if they are not. The distance between two trees is the sum of the distances of the corresponding nodes – i.e., distances between nodes that overlap when the two trees are overlaid, starting from the root. The distance between two trees is normalized by dividing by the size of the smaller of the two trees. The diversity objective is defined as the average squared distance to other members of the population. An improved version of the above distance metric between two trees is proposed in Ekart and Nemeth (2000) and used to do fitness sharing in GP. Their method includes the following three steps:

1) The two GP trees to be compared are brought to the same tree-structure (only the contents of the nodes remain different).
2) The distance between each pair of symbols situated at the same position in the two trees is computed.
3) The distances computed in the previous step are combined in a weighted sum to form the distance of the two trees.

The major improvement of this method is that it differentiates the types of nodes when calculating the distance between two nodes. It first divides the GP functions and terminals into several subsets. For nodes with types belonging to the same subset, it calculates the relative distance. For nodes with types belonging to different subsets, it uses a defined function to make sure that the distance between nodes from different subsets is larger than that between nodes of the same subset. It also considers the fact that a difference at some node closer to the root could be more significant than a difference at some node farther from the root, using a multiplier K to distinguish them. Edit distance and phenotypic distance for fitness sharing for GP are also tested in their experiment. The former gets slightly better accuracy but with relatively high computational cost. The latter doesn't provide much improvement over the original GP without fitness sharing.

Implicit fitness sharing (McKay, 2000) has also been applied to GP. Instead of calculating the distance between the structures of GP trees, it is a kind of phenotypic (behavior-based) fitness sharing method. The fitness is "shared" based on the number of other individuals who have similar behaviors, capabilities or functions. Implicit fitness sharing provides selection pressure for each individual to make different predictions from those made by other individuals.

Population diversity of TYPE III problems in GP has not been investigated thoroughly. These problems are characterized by the need for simultaneous optimization of topology and parameters. In a GP population, structure diversity is needed to enable efficient topology exploration, which is the main objective, in most case, for discovery of innovative designs. At the same time, the goodness (or fitness) of a structure can only be evaluated after sufficient parameter exploration within the same structure. Thus, the parameter diversity of each structure also needs to be maintained. As a result, in the context of variable structure and parameter design by GP, the

population diversity has some significant differences from that of a GA, in the following respects:

- Number of peaks

  When applying fitness sharing in GA, two assumptions are made. One is that the number of peaks is known or can be estimated. The second is that the peaks are almost evenly distributed. In many problems of GA, a relatively limited number of peaks is expected to enable efficient use of fitness sharing. However, in TYPE III problems, each structure may have a huge number of peaks with respect to its parameter space, while in the structure space, each structure is a distinct peak, since the structure space is not a continuous space, but rather a highly nonlinear discrete space.

- Continuity of search space

  In GA, many problems can be considered as defined in an approximately continuous space, although sometimes certain aspects have distinctly discrete behavior. However, in TYPE III problems, GP deals with a highly discrete structure space that also has a huge continuous space (of parameter values), since for each structure, the search for appropriate parameters can be regarded as an instance of GA search.

- Constraints

  In GA, only parameter constraints exist. However, in TYPE III problems, GP must deal with both structure constraints and parameter constraints.

The demand for structure diversity as well as parameter diversity makes the existing fitness sharing methods inefficient for Type III problems. For fitness-space-based fitness sharing (Rodriguez-Vazquez, 1997) and the implicit fitness sharing (McKay, 2000) methods, significant parameter diversity is lost since they do not promote coexistence of individuals with the same structure but with different parameters in order to enable efficient parameter search. Fitness sharing with the distance metric, as in (Ekart, 2000; KeJong, 2001), is also inefficient in this case. First, the computational cost is still demanding, since in TYPE III problems, a complex structure and its parameters often require a big tree – perhaps 1000 - 2000 nodes in most of our experiments – especially when parameters are normally represented by a numeric subtree such as Koza uses (Koza, 1999). Second, but more importantly, the underlying assumption of the above distance metrics is that structural dissimilarity measured between two GP trees meaningfully reflects the dissimilarity in function between the two structures. However, as the structure space represented by a GP tree is a highly non-linear space, in most cases, a change of a single (non-parameter) node changes the behavior of the GP tree dramatically. This phenomenon can be traced to the weak causality of GP (Rosca, 1995), which means that small alterations in the underlying structure of a GP tree cause big changes in the behavior of the GP tree. So measuring a sort of "Hamming" distance between the structures of two GP trees to predict the difference of the behavior/function is not well founded, and thus inefficient. This makes a useful definition of a sharing radius hard to determine. It seems that distance metrics in the structure space and the parameter space and the association of a set of parameters with the structure to which they apply must be faithfully captured in order to most effectively maintain both structure diversity and parameter diversity and thereby to achieve efficient search. Therefore, given the inherent difficulty of structure/function mapping, perhaps it is counterproductive to use any structural similarity measure beyond the most basic and completely faithful one – the identity mapping: two structures are either identical, or they are not. That is the structural distance measure used here. While it is possible to define a broader relationship that still captures identity of function (for example, if swapping of the order of two children of a node has no effect on the function computed), such definitions depend on the semantics of the functions, and were not implemented here.

3 BALANCED STRUCTURE AND PARAMETER SEARCH IN EVOLUTIONARY DESIGN BY GP

In design problems involving both variable structure and variable parameters, search must be balanced between the structure and parameters. On one hand, each structure needs sufficient exploration of its parameters to develop its potential to some extent, which means that a reasonable number of individuals of the same structure must probably be kept in the population. On the other hand, no structure should dominate the population, or it would prevent sufficient future exploration of the structure space.

Premature convergence of structures in evolutionary design by GP can be caused by neglecting the different roles of structure and parameter search. In standard GP, nodes at which to perform crossover and mutation are selected randomly from the entire set of nodes, treating those specifying structure modifications identically with those specifying numerical modifications (provided that numerical subtrees are used to define the parameters of components, as is often done). This means that a new circuit structure is often discarded by the selection process if its fitness is low with its initial set of parameters. The result is that often, structures of moderate quality with slightly better parameters proliferate and dominate the population, while inherently better structures with bad parameters are discarded. This is called the *premature structural convergence* problem. This phenomenon arises from the fact that ”promising” structures are often discarded just because their current parameters are not adjusted well enough to demonstrate their potential. Ideally, a structure should be discarded only when it is demonstrated to be bad after a sufficient effort to adjust its

parameters. In addition, since there are often many more numeric nodes than structure modifying nodes, premature structural convergence is accentuated, since there is a lower probability of choosing a structure modifying node that will generate a new structure than of choosing a node that changes only parameters.

In order to address this problem, structure and parameter search must be controlled explicitly. In our work, a probabilistic method is first devised to decide whether GP does a structure modification (crossover or mutation on a structure modifying node) or does parameter modification (crossover or mutation on a parameter modifying node). Since structure changes have a more fundamental effect than the parameter changes on the performance of the system, we introduce a bias toward more parameter modifications than structure modifications by controlling the probability of selecting these types of nodes for crossover and mutation sites. The following example probabilities are defined to facilitate keeping the structure and its function stable and to allow parameters to be adjusted well enough to demonstrate the potential of a structure.

$$p(\textit{structure modification}) = 0.1$$
$$p(\textit{parameter modification}) = 0.9$$

We also use explicit control of the node selection process to achieve balanced parameter evolution for all parameters in a structure. During the parameter modification stage, we first establish a list of all variables whose values need to be established during evolution, then we randomly select a variable as the current variable to be changed. We then select a node in the numeric subtree of this variable and do a crossover or mutation operation. In this way, each variable has an equal opportunity to be changed during evolution. This improvement speeds the evolution of balanced numeric subtrees. All variables tend to have numeric subtrees with similar depths.

Even with the methods above, premature structural convergence still often occurs as structures with well-fitted parameters quickly dominate the whole population. The Structure Fitness Sharing (SFS) method is proposed to control the reproduction of high-fitness structures. Our assumptions are that fitness sharing can profitably be based on the number of individuals with the same structure, and that distance between the structures of two GP trees with distinct structures is not generally an adequate predictor of the differences between their behaviors. Thus, any "counting of positions where the trees differ" distance metric is not well founded. Instead, a simple labeling technique is used to distinguish structures.

4 STRUCTURE FITNESSS SHARING (SFS)

Structure Fitness Sharing is the application of fitness sharing to structures in GP. In contrast to the GA fitness sharing using a distance measure to identify peaks, in SFS, fitness sharing is based on the tree structures, treating each tree structure in GP as a peak in the space of parameters and structures.

In SFS, each structure is uniquely labeled, whenever it is first created. So long as GP operations on an individual do not change its structure, but only its parameters, the structure label of this individual is not changed. Parameter crossover and mutation, or replication of the individual, simply increase the number of individuals with this structure (label) in the population. If structure modifications are conducted on an individual that change the structure – for example, we change a Rep_C (a GP function node replacing a resistor or inductor of the circuit with a capacitor) to a Rep_I (a GP function replacing a resistor or capacitor of the circuit with an inductor) node – then a new structure label (structureID) is created and is attached to this new individual. Our assumption is that the possibility that any particular structure-altering operation produces exactly the same structure possessed by other individuals in the current population is relatively low, so it is not necessary (or worthwhile) to check a new structure against all other existing structures to see if it is identical with one of them (and so could use its label), although a hashing technique might make this relatively easy to do. Furthermore, even if some newly created individual shares the same structure with another individual but is labeled with a different structure label, the algorithm is not strongly affected, so long as it occurs infrequently.

In standard GP, individuals with certain structures will prosper while others will disappear because of their low fitnesses. If this process is allowed to continue without control, some good structures (usually one) tend to dominate the population and premature convergence occurs. To maintain diversity of structures, fitness sharing is applied to individuals of each structure. SFS decreases the fitness of the individual as follows: SFS penalizes only those structures having too many individuals, according to the following fitness adjustment rule used for the experiments in this paper:

N_s : Number of structures to be searched simultaneously

N_{esp} : Expected number of search points (individuals) for each structure in the whole population

$N^{s_i}_{Ind_i \in s_i}$: Number of individuals with structure s_i (of which individual ind_i is one)

For each individual Ind_i if $N^{s_i}_{Ind_i \in s_i} > 0.8 * N_{esp}$ then

$$k = \left(\frac{N^{S_i}}{\frac{Ind_i \in S_i}{N_{esp}}} \right)^{-\alpha} \quad \text{where } \alpha = 1.5$$

If $k > 1$ *then* $k = 1.0$

$$f_{adj} = f_{old} \ * \ k \qquad (1)$$

With this method, each structure has a chance to do parameter search. Premature convergence of structures is limited, and we can still devote more effort to high-fitness structure search.

4.1 LABELING TECHNIQUE IN SFS

A labeling technique is used in SFS to distinguish different structures efficiently. A similar technique has been used by Spears (1994), in which tag bits are used to identify different subpopulations. Spears's result suggests that in crowding and fitness sharing in GA, we only need to decide whether or not two individuals have the same label. The added precision of the distance metric for maintaining the diverse state of a subpopulation is often unnecessary. In SFS, the label is used only to decide whether or not two individuals have the same label (*i.e.,* structure). We use simple integer numbers as labels rather than more complicated tag bits. Ryan (1994, 1995) also uses similar labelling ideas to decide which race an individual belongs to in his racial GA (RGA).

4.2 HASH TABLE TECHNIQUE IN SFS

In order to keep track of all individuals with each particular label, SFS uses a hash table is used -- this speeds up the access to the structure by each individual when we do crossover, mutation, and reproduction. Each time we create a new structure, we create an entry in the hashtable with the structureID (next integer) as the key. The size of the hash table is controlled to accommodate at most 500 structures in our experiments. Whenever the number of structure entries in the hashtable exceeds 500, those structure entries with no individuals in the current population or with a low fitness of its best individuals and with old ages (generations since their labels were created) are removed from the hashtable. The corresponding individuals of these structures are removed from the current population at the same time.

4.3 THE STRUCTURE FITNESS SHARING ALGORITHM IN GP

The following is the outline of the algorithm of SFS as applied to GP:

Step 1: Initialize the population with randomly generated individuals. Initialize the structure hash table.

Step 2: Assign each individual a unique label. Here a label is just an unassigned integer number incremented by one at each assignment.

Step 3: Loop over generations

3.1 Select the parents for a genetic operation according to their standard fitness

3.2: If current operation is an operation that changes the structure from that of the parent(s), (including crossover and mutation at structure operator nodes of GP trees)

Create a new label for each new structure created and add the new structure item to the structure hash table.

3.3: If the current operation is a parameter modification (mutating the parameter nodes or crossing over at a parameter node) or only replication of an existing individual, do not create a new label. New individuals inherit the labels from their parents. Update information about the structure items in the hash table, including the best fitness of this structure, number of individuals, age, etc.

3.4: If the maximum number of structures in the hash table is reached, first purge those structures that have no individuals in the current population. If there are still too many structures, then delete those structures whose best individuals have lower fitness and high age (>10 generation, in our case) and delete their individuals in the population and replace them with new individuals formed by crossover or mutation until the maximum number of structures in hash table is kept.

3.5: Adjust the fitness of each individual according to equation (1).

Step 4: If stopping criterion is satisfied, stop; else go to step 3.

5 EXPERIMENTS

5.1 PROBLEM DEFINITION

GP with the SFS technique has been applied to an analog circuit synthesis problem that was previously approached using GP without SFS (Rosenberg, 2001). In this problem, an analog circuit is represented by a bond graph model (Fan, 2001) and is composed of inductors (I), resistors (R), capacitors (C), transformers (TF), gyrators (GY), and sources of effort (SE). The developmental evolution method similar to (Koza, 1999) is used to evolve both the structure and parameters of a bond graph representation of a circuit that achieves the user-specified behavior. With this method, a set of structure modifying operators (e.g. Rep_C, Rep_I) are provided to operate on the embryo

bond graph. A set of numeric operators (such as +, -, *, /) are provided to modify the parameters of the structure.

In this paper, the design objective is to evolve an analog circuit with response properties characterized by a pre-specified set of eigenvalues of the system equation. By increasing the number of eigenvalues specified, we can define a series of synthesis problems of increasing difficulty, in which premature convergence problems become more and more significant when traditional GP methods are used. This eigenvalue assignment problem has received a great deal of attention in control system design. Control over eigenvalues in designing systems, in order to avoid instability and to provide particular response characteristics, is often an important and practical problem.

Circuit synthesis by GP is a well-studied problem that generally demands large computational power to achieve good results. Since both the topology and the parameters of a circuit affect its performance, it is easy to get stuck in the evolution process.

5.2 EXPERIMENTAL SETUP

In the example that follows, a set of target eigenvalues was given and a bond graph model with those eigenvalues was generated. Table 1 shows the three sets of 6, 8, and 10 target eigenvalues used as targets for example genetic programming runs:

We applied single population GP with and without SFS and multi-population GP with and without SFS to all three problem instances. Each experiment was repeated 10 times with different random seeds.

Table 1. Target Eigenvalues

Problem 1:	6-eigenvalue problem
$-0.1 \pm 5.0j, -1.0 \pm 2.0j, -2.0 \pm 1.0j$	
Problem 2:	8-eigenvalue problem
$-0.1 \pm 5.0j, -1.0 \pm 2.0j, -2.0 \pm 1.0j, -3.0 \pm 0.7j$	
Problem 3:	10-eigenvalue problem
$-0.1 \pm 5.0j, -1.0 \pm 2.0j, -2.0 \pm 1.0j, -3.0 \pm 0.7j$ $-4.0 \pm 0.4j$	

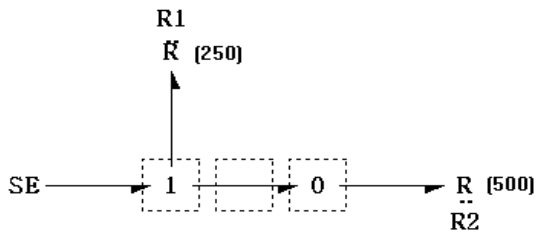


Figure 1. The Embryo Bond Graph Model

The embryo model used is shown in Figure 1. It represents an embryo bond graph with three initial modifiable sites (represented as dotted boxes). In each case, the fixed components of the embryo are sufficient to allow definition of the system input and output, yielding a system for which the eigenvalues can be evaluated, including appropriate impedances. The construction steps specified in the GP tree are executed beginning from this embryo. The numbers in parentheses represent the parameter values of the elements.

Three circuits of increasing complexity are to be synthesized, with eigenvalue sets as specified above. The GP parameter tableau for the single population method is shown in Table 2 below.

These problems exhibit a very high degree of epistasis, as a change in the placement of any pair of eigenvalues has a

Table 2. Parameter Settings for GP

Parameters of Single Population GP	Popsize: 1000 init.method = half_and_half init.depth = 3-6 max_nodes = 1000 max_depth = 15 crossover rate = 0.9 mutation rate = 0.1 max_generation = 1000
Additional Parameters of Multi-Population GP	Number of subpopulations = 10; Size of subpop = 100 migration interval = 10 generations migration strategy: ring topology, migrate 10 best individuals to the next subpopulation in the ring to replace its 10 worst individuals
SFS Parameters	N_s : 50 N_{esp} : 20 = popsize/ N_s

strong effect on the location of the remaining eigenvalues. Eigenvalue placement is very different from "one-max" or additively decomposable optimization problems, and these problems become increasingly difficult with the problem order. The performance of each of the three GP approaches is reported in Figure 2, in which the four GP methods are indicated by

OneGP: single population GP, no SFS

MulGP: multi-population GP, no SFS

ONE.SFS: single population GP with SFS

MULPOP.SFS: multi-population GP with SFS

To observe the effect of structure fitness sharing, we monitor the number of distinct structures in the experiments with and without SFS techniques. From Fig 2, one can see that Structure Fitness Sharing can significantly improve the performance for single population GP and also does better in multi-population GP,

though the difference is not as significant. The reason may be that multi-population runs already provide an inherent diversity maintenance mechanism. We also find that SFS can help to provide probabilistic control of structure and parameter modifications to maintain a stable number of search structures in the whole population, as illustrated in Fig 3.

6 CONCLUSIONS

In this paper, Structure Fitness Sharing (SFS) is proposed to achieve balanced structure and parameter search in evolutionary design by Genetic Programming. SFS can effectively prevent the dominance of any specific structure and when combined with probabilistic control of structure and parameter modification, SFS can maintain a stable number of structures for simultaneous structure and parameter search. The labeling technique in SFS eliminates the necessity of computing the distance between two individuals, which saves computing effort that we believe is often largely wasted when attempting to measure GP structural similarity. The user parameters of the standard fitness sharing method are also eliminated (e.g. the sharing radius). All that must be done is to define the fitness adjustment scheme: how to penalize the fitness of a structure when the number of individuals with that structure label grows large enough to threaten the diversity of the population. The hash table technique allows SFS to quickly update the structure information about the current population during evolution. More complicated balanced structure parameter search methods can be derived using the concept of structure diversity. For example, the authors intend to incorporate the age concept and the elitism method of multi-objective evolutionary computation into SFS. We also intend to explore use of a separate GA for explicit exploration of the parameter spaces of individual structures, with the expectation of a significant impact on the selection, crossover and mutation dynamics of the overarching simultaneous evolutionary search of structure and parameters.

Acknowledgment

Thanks to Ahmad R Shahid for his assistance with the hash table technique. This work was supported by the National Science Foundation under contract DMI 0084934.

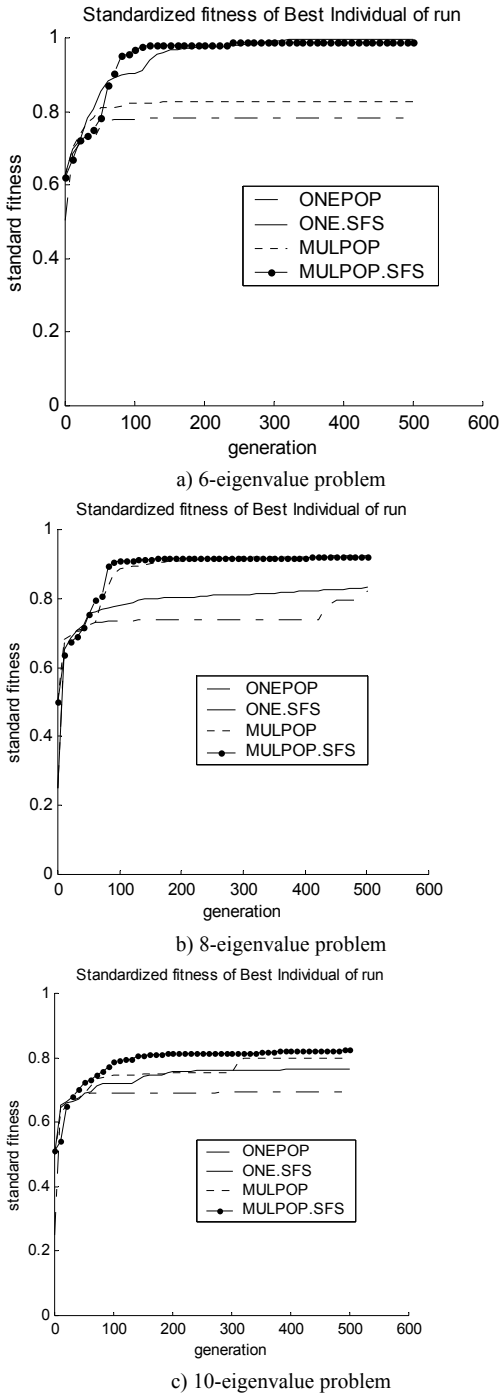


Figure 2. Fitness of Best Individual to Date vs. Generation

References:

P. Nordin, *Evolutionary Program Induction of Binary Machine Code and Its Application*, PhD thesis, University of Dortmund,Germany, 1997.

L. Spector, H. Barnum, and H. J. Bernstein, "Quantum Computing Applications of Genetic Programming", *Advances in Genetic Programming 3*, L. Spector *et al*., eds., MIT Press, Cambridge, Mass., 1999, pp.135-160.

D. Andre and A. Teller, "Evolving Team Darwin United," *RoboCup-98: Robot Soccer World Cup II. Lecture Notes in Computer Science*, M. Asada and H. Kitano, eds., Vol. 164, Springer-Verlag, Berlin, pp.346-352, 1999.

J. R. Koza *et al*., *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann, San Francisco, 1999.

K. Rodriguez-Vazquez, C. M. Fonseca, and P. J. Fleming, "Multiobjective Genetic Programming : A Nonlinear

System Identification Application," *Proc. Genetic Programming '97 Conference*, pp. 207-212, 1997

C. M. Fonseca, and P. J. Fleming, "Genetic Algoritms for Multiobjective Optimisation: Formulation, Discussion, and Generalization," *Proceedings Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Francisco, pp. 416-423, 1993.

S. Oliker, M. Furst, and O. Maimon, "A distributed genetic algorithm for neural network design and training," *Complex Systems*, 6, pp. 459-477, 1992.

K. A. DeJong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD thesis. University of Michigan. Dissertation Abstracts International 36(10), 5410B. (University Microfilms No. 76--9381), 1975.

E. D. DeJong, R. A. Watson, and J. B. Pollack, "Reducing Bloat and Promoting Diversity using Multi-Objective Methods," *Proc. GECCO-2001*, Morgan Kaufmann, San Francisco, pp. 11-18.

S. W. Mahfoud, "Crowding and Preselection Revisited," *Proc. PPSN-92*, Elsevier, 1992.

D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison Wesley, 1989.

D. Beasley, D.R. Bull, and R.R. Martin, "A Sequential Niche Technique for Multimodal Function Optimisation," *Evolutionary Computation*, 1, pp. 101-125. 1993.

A. Ekárt; S. Z. Németh, "A Metric for Genetic Programs and Fitness Sharing," in R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin, T. Fogarty (eds.), *Genetic Programming, Proceedings of EUROGP'2000*, Edinburgh, 15-16 April 2000, LNCS volume 1802, pp. 259-270.

R. I. McKay, "Fitness Sharing in Genetic Programming," *Proc. GECCO-2000*, Las Vegas, NV, Morgan Kaufmann, San Francisco, July, 2000.

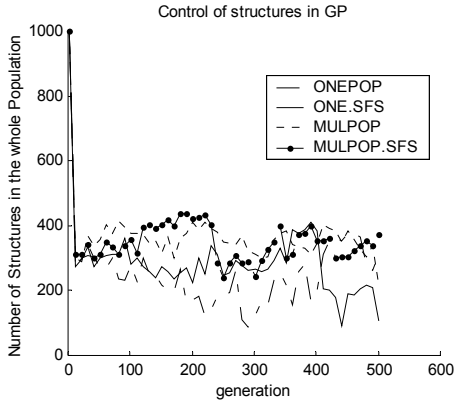
J. P. Rosca and D. H. Ballard, "Causality in Genetic Programming," in L. J. Eshelman, ed., *Proc. Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Francisco, pp. 256-263, 1995.

W. M. Spears, "Simple Subpopulation Schemes," *Proc. Evolutionary Programming Conf.*, pp. 296-307, 1994.

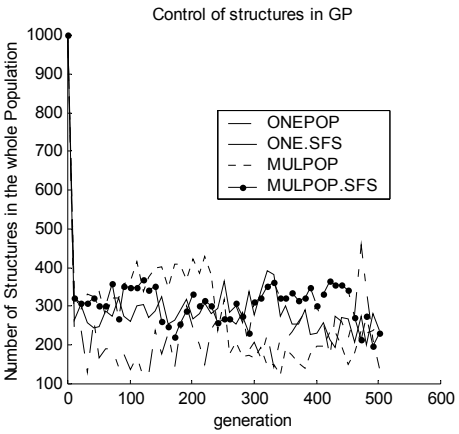
C. Ryan, "Pygmies and Civil Servants, in K. E. Kinnear, Jr., ed., *Advances in Genetic Programming*, pp. 243-263. MIT Press, 1994.

C. Ryan, "Racial Harmony and Function Optimization in Genetic Algorithms - the Races Genetic Algorithm," in *Proc. Of Evolutionary Programming 1995*. pp. 296-307. MIT press.

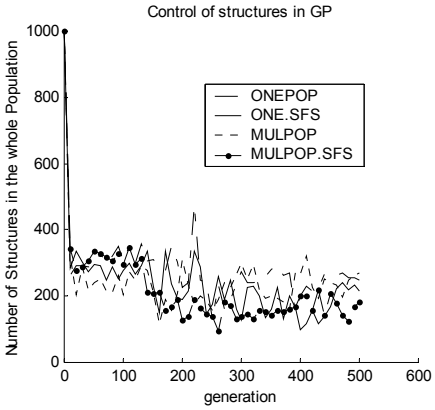
R.C. Rosenberg, E. D. Goodman and K. Seo, "Some Key Issues in Using Bond Graphs and Genetic Programming for Mechatronic System Design", *Proc. ASME International Mechanical Engineering Congress and Exposition*, 2001, November 11-16, New York.



a) 6 eigenvalue problem



b) 8 eigenvalue problem



c) 10 eigenvalue problem

Figure 3. Number of Structures in Population vs. Generation

Z. Fan, J. Hu, K. Seo, E. Goodman, R. Rosenberg, and B. Zhang, "Bond Graph Representation and GP for Automated Analog Filter Design," *Genetic and Evolutionary Computation Conference 2001 Late Breaking Papers*, ISGEC Press, San Francisco, pp. 81-86.