



Optimization of a Genetically Engineerable Evolvable Program (GEEP) for Improved Data Understanding

Gary D. Boetticher [UHCL] / Kimberly Kaminsky [UHCL]

Abstract

Historically, Genetic Program (GP) modeling avoids the usage of explicit knowledge. The assumption is that explicit knowledge interferes in the discovery of novel solutions. However, lately researchers have applied explicit domain knowledge in GP modeling. Using explicit knowledge is plausible considering the vast amount of human knowledge, wisdom, and experience embodied in existing equations. Recognizing the importance of infusing domain knowledge into GP modeling, this work defines a new technique called a Genetically Engineerable Evolvable Program (GEEP). This approach architecturally supports the infusion of explicit knowledge from various domains in constructing a GP by uncoupling the domain knowledge from the GP, thus, allowing the freedom to dynamically apply different domain analysis (in the form of equations).

One crucial step within the process is the optimization of chromosome solutions. Reducing solution complexity allows for redundancy checking, promotes optimized representations, and enhances the potential for “human-understandability.” This last benefit is critical for interpreting the significance of an automated solution.

Different experiments are conducted exploring an optimized version of a GEEP against an unoptimized version. Results from the experiments are statistically assessed.

GENETIC PROGRAMMING (GP) SUPPORTS AUTOMATIC programming by genetically breeding a population of computer programs using the principles of Darwinian natural selection and biologically inspired operations. They have demonstrated some of their potential by evolving programs for medical signal filters (Oakley), modeling complex chemical reactions (Bettenhausen), performing optical character recognition (Andre), target identification (Tackett), and routing of analog electrical circuits (Koza 2000).

Historically, incorporating explicit knowledge within a GP is not an accepted practice. Gero and Kasakov claim that genetic programs are “knowledge lean machines; they make no use of knowledge in their execution” where this approach supposedly gives GP “robustness and breadth of applicability.” Angeline discourages the use of explicit knowledge and argues that it should be used as “a last resort.” He claims that “including explicit knowledge for a particular problem removes the opportunity for contradictory knowledge to emerge.” Thus, the historical approach forces all genetic programs to reinvent many existing algorithms.

Considering the decades worth of legacy data, the decades

GEEP—Dr. Gary D. Boetticher, Asst. Professor of Science and Computer Engineering, leads a research team in the field of genetic engineering with evolvable programs.

worth of software repositories, and the current rate of data propagation (2 exabytes per year), the GP modeling process would benefit immensely from the use of explicit knowledge. Lately, GP modeling has recognized the importance of explicit knowledge in GP modeling. The research literature contains examples of GP modeling using financial domain knowledge in GP modeling (Dempster, Frick, Mahfoud). These successes demonstrate the importance of explicit domain knowledge with GP modeling. However, they tend to bundle the domain knowledge within the GP model. As a consequence, it is difficult to create dynamic GP models.

An alternative approach is to provide explicit (domain) knowledge which is not bundled to the genetic programs, producing “knowledge-wise machines.” This new technique is referred to as a Genetically Engineerable Evolvable Program (GEEP). Unbundling the explicit domain knowledge from the GP model enables the examination of the relationship between domain knowledge and GP model formation. Hence, the three long-term objectives of this research comprise domain knowledge utility; reusability of domain knowledge; and discovery of new knowledge.

Domain knowledge utility. Given the vast amount of domain knowledge in the form of equations, formulas, and algorithms, there is a major opportunity to incorporate this information into a GP. This would enable the GEEP to produce better models than traditional GP models.

Reusability of domain knowledge. Incorporating domain knowledge from multiple domains makes it possible to perform ontological assessments of knowledge. By projecting domain knowledge into a new domain, it is possible to discover new relationships and ontologically bridge two or more domains.

Discovery of new knowledge. Rather than teach a GP concept we already know, another long-term goal is to either refine existing knowledge or discover new formulas. Theoretically, this could lead to major discoveries in many disciplines.

One of the challenges in creating such a powerful tool is how to optimize the GEEP as it traverses the solution space. As domain knowledge is incorporated into the GEEP paradigm, the GEEP will produce very sophisticated solutions in the form of equations. It is critical to reduce the complexity of these equations into its simplest form for several reasons.

A GEEP, like other genetic programs, creates a set of solutions in the form of equations. Traditionally, these equations are called chromosomes, in the genetic algorithm/genetic programming discipline. One of the goals in genetic programming is to explore a majority, if not all, of the solution space. Ideally, this is accomplished by generating a diverse set of chromosomes. If there is a lack of diversity, then the genetic program will converge to a relatively few solutions.

By not reducing a chromosome to its simplest form, the problem is that it is very easy to produce equivalent solutions

where the only difference is that one equation is reduced, and the other is not. This leads to a false perspective regarding the amount of diversity within a genetic program.

A second reason for reducing an equation into its simplest form is human readability. One advantage of using a genetic program over another machine learner, such as a neural network, is that the solution is presented in human readable form. Obfuscating the solution deters the reader from drawing any useful knowledge from the solution.

This paper explores the application of optimization techniques to a GEEP through a series of experiments. Section two describes related research in terms of the infusion of domain knowledge into a genetic program along with the optimization of a genetic program. Section three provides an overview of the GEEP paradigm. Optimization techniques are discussed in section 4. A series of experiments are performed in section 5 in order to assess the optimization techniques. The results of these experiments are statistically validated. Section 6 discusses the results. Sections 7 and 8 describe conclusions and future directions.

Related Research

Current research related to domain knowledge and GP models can be characterized two ways. In the first case, domain knowledge is not included in the GP. Instead, domain knowledge is extracted from a GP in the form of a collapsed chromosome solution. Several examples include Automatically Defined Functions, Genetic Library Builder, and Hierarchical Genetic Programming. In the second case, domain knowledge is statically infused into a GP.

Automatically Defined Functions

In Automatically Defined Functions (ADF), the architecture of the GP solution is hierarchically arranged during design time. ADFs are sub-trees that may be treated as functions within the GP. The notion of ADFs was extended to include architecture-altering operations (e.g., recursion, looping) (Koza 1999).

ADFs incorporate *programming domain knowledge* into the GP paradigm but fail to include *user domain knowledge* in the GP modeling process.

Genetic Library Builder

The Genetic Library Builder, (GLiB), employs a novel approach to the modularization process. In GLiB, mutation plays a major role in defining new modules. The first mutation operator is *compression*. Compression randomly selects a subtree from the parent to become the body of a newly defined module that is added to the genetic library. The second operator is *expand*. Expand searches the parent for all modules at the top level and selects one randomly. The module is replaced with its definition, which is stored in the genetic library. Once a subtree has been compressed, no further manipulation of the module's content by crossover or mutation is allowed. Since only modules at the highest level of the tree are expanded, modules further down in the hierarchy are protected from further expansion and manipulation. The idea is to keep productive modules intact throughout the

reproductive process. At the same time, less productive modules will disappear along with the individuals in the population that contain them.

GLiB claims more domain independence than ADFs because the hierarchy in ADFs is static. Thus, while the definition of each individual ADF is emergent, the number of parameters to each module is static. Angeline concludes that this property of ADFs makes them more suitable for problems with more explicit knowledge and GLiB more suitable as a general Genetic Programming algorithm.

This process is limited to the abstraction of implicit knowledge. The abstraction process is arbitrary, lacking the capability of applying semantic reasoning to the process and performing intelligent abstraction.

Hierarchical Genetic Programming

Hierarchical Genetic Programming (HGP) locates useful building blocks of code so that they may be transformed into new functions. HGP differs from both GLiB and ADF in that it does not rely on random selection for the code evolution. Instead, it depends on heuristics to select the code for modularization and then evolves a hierarchy of the functions. Unlike ADFs and GLiB, which assume that poor performing modules will be automatically weeded out when whole programs are evaluated, HGP attempts to evaluate each code fragment individually before introducing it into the population. At the end of each evolutionary epoch, part of the population is replaced with random individuals built using the extended function set. Rosca notes that all techniques employing modularization increase the diversity of the population, which increases the efficiency of the Genetic Program.

Effective module selection depends upon the infusion of explicit *user domain knowledge* in the form of a set of rules.

As mentioned in the introduction, the research literature contains examples of GP modeling using domain knowledge in GP modeling. For instance, the financial domain (Dempster, Frick, Mahfoud) has produced three instances of successful GP modeling. These successes demonstrate the importance of explicit domain knowledge with GP modeling. However, their approach uses static domain in that they bundle it within the GP model. As a consequence, it is difficult to create dynamic GP models which can span multiple domains.

Thus, what is desired is an approach which allows a GP to build a model from a vast array of components spanning multiple domains.

Creating a powerful tool capable of building sophisticated models will generate complex answers. It will be important to be able to optimize the solutions produced by GP.

Lucier applies two types of optimization techniques to the GP application. The first is specific to the domain under investigation (image analysis). The second technique, which applies to GP operators, flattens "and/or" trees by focusing upon the precedence order of the *and* and *or* operators.

Genetically Engineerable Evolvable Program (GEEP)

Genetic Program represent solutions, called chromosomes, as tree structures of variable length. Normally these tree structures consist of a set of simple mathematical operators

and corresponding operands.

Some of the more advanced GP models may include additional functions and/or procedures in the tree structure. However, these GP models are designed for a specific domain and the functions/procedures are “hard wired” into the GP.

The GEEP approach supports the use of functions and/or procedures. Figure 1 illustrates the usage of domain specific knowledge, in this case trigonometry, in developing a GEEP-based solution.

This proposed research holds the promise of producing a more sophisticated, yet accurate, solution in less time by exploiting what is already known.

What distinguishes the GEEP approach from tradition GP modeling is the ability to seamlessly incorporate domain knowledge, in the form of procedures and functions, from various domains into the software. This capability allows a GEEP to leverage off existing domain knowledge and build more sophisticated models. As this research matures, it will spawn further research regarding domain analysis within GP. It may be possible to have a user direct which domain knowledge to include/exclude in the modeling process. Figure 2 depicts one future architecture scenario for the GEEP model.

There may be a vast repository of knowledge, collected from multiple domains, available for the GEEP. The GEEP will be capable of building very sophisticated solutions. However, because of the abundant number of functions/procedures available to the GEEP, it is necessary to optimize the solution building process. It seems plausible that optimizing the process generates faster and better solutions. Furthermore, it seems reasonable that optimizing the solution produces more readable models. This readability leads to humans learning more about their environment from an empirical perspective.

The Optimization Process

Optimization assumes two forms. In the first case it, refers to the finding of a concise representation for a given chromosome (equation). In the second case, it refers to eliminating redundancy within a population.

Chromosome (Equation) Optimization

Chromosome optimization seeks to find the most concise representation for a given chromosome. Below is a description of the various optimization techniques applied during the experiments.

Operand ordering. For those operators that are commutative, it is possible to swap the order of the operands. For example:

$$y \text{ operator } x \rightarrow x \text{ operator } y$$

Equation reduction. There are many opportunities for reducing an equation. Some examples applied during the experiments include:

$$\begin{aligned} x/x &\rightarrow 1 \\ x - x &\rightarrow 0 \\ x + x &\rightarrow 2 * x \end{aligned}$$

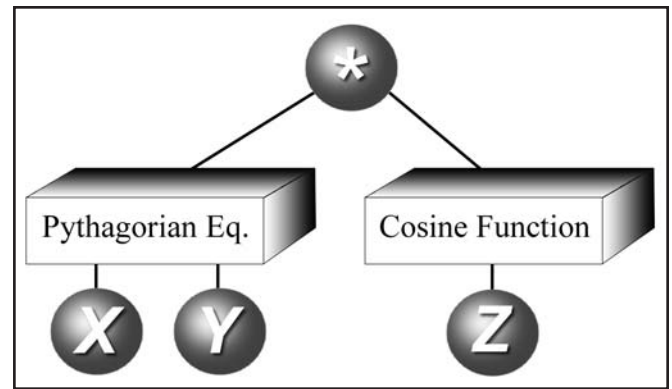


Figure 1. Using Domain Specific Knowledge To Build Models

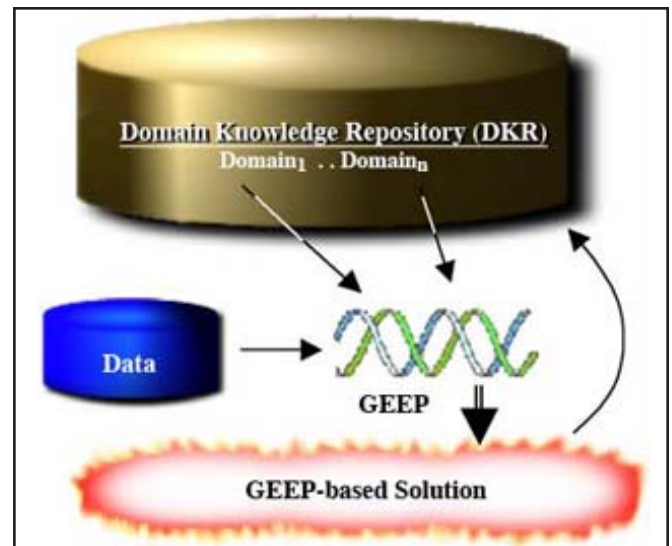


Figure 2. GEEP Solution = DKR + GP + Data

Applying these optimization techniques allows the GEEP to detect equivalent chromosome representations.

Population Optimization

After chromosome optimization, the next step sorts the chromosomes and discards duplicates. New chromosomes are generated and added to the population. The stipulation is that new chromosomes differ from current population members.

These optimization techniques incur overhead costs in terms of performance. One question is how effective these optimization techniques are in terms of producing a better answer in less time (in terms of generations). The answer may be determined by conducting several experiments.

Experiments

Validation of the optimization techniques is based upon three different experiments. All three experiments compare an unoptimized GEEP against an optimized GEEP. The experiments seek to assess the optimization techniques as applied to a progression of increasingly complex data. All data sets are independent of each other and all are based upon a polynomial equation.

All experiments used the same parameter settings. The only difference between experiments is the equations used to generate the data. The parameter settings are outlined in Table 1.

The experiments used all permutations of The Chromosome Length, No. of Generations, and Population, for a total of 27 runs.

The Mean Absolute Relative Error (MARE) served as the fitness function. It is defined as follows:

$$MARE = \sum \frac{|estimated_i - actual_i|}{actual_i}$$

The operator set included the basic math operators along with: $\log_{10}(x)$, e^x , *absolute value*, *sine*, and *cosine*.

Experiment 1

The first experiment seeks to model the following equation:

$$v^2 + w*x - y/z$$

This experiment is repeated 27 times in order to provide a sufficient amount of data for statistical analysis. Table 2 shows the results from these experiments. Column 1 represents the maximum number of generations the GEEP could train before quitting. Columns 2 and 4 show the actual number of generations the GEEP trained, non-optimized and optimized respectively. Columns 3 and 5 show the fitness values for non-optimized and optimized models. The fitness values range from 0 to 1000.

After sorting each column of fitness values, values are plotted as depicted in Fig. 3. Examining Fig. 3 suggests that optimizing the learning process produces better results.

Performing a *t*-test determines whether this difference is statistically significant. The statistical results are displayed in Table 3.

The one-tail *t*-test produces a value of 0.0554. This implies that there is no statistically significant difference between the means.

The actual numbers of generations needed to solve the problem are also examined for statistical significant difference. Table 4 shows these results.

The *t*-test result of 0.06039 indicates that there is no statistical significance in the number of generations needed to build a model.

Experiment 2

The next experiment models a polynomial equation which contains trigonometry functions. This experiment models the following equation:

$$e^X * (\sin(X) + \cos(Y))$$

This experiment is repeated 27 times in order to provide a sufficient amount of data for statistical analysis. Table 5 shows the results from these experiments. Column 1 represents the maximum number of generations the GEEP could

Table 1. Experiment Parameter Settings

GEEP Parameter	Setting
Selection Process	Roulette
Crossover rate	75%
Mutation rate	0%
Elitism	1 Chromosome
Chromosome length	200, 600, 100
No. of Generations	50, 1000, 1950
Population Size	200, 600, 1000
Number of Data Samples	200

Table 2. Results from Experiment 1

Maximum No. of Generations	Non-Optimized		Optimized	
	Actual No. of Generations	Fitness Values	Actual No. of Generations	Fitness Values
200	200	371	200	372
200	200	482	200	372
200	200	334	200	372
200	200	364	200	375
200	200	365	200	372
200	200	471	200	314
200	200	183	200	375
200	200	471	200	442
200	200	439	200	757
600	600	757	600	367
600	600	300	600	380
600	600	91	600	338
600	600	373	600	385
600	600	375	600	379
600	600	487	600	480
600	600	757	600	430
600	600	300	342	1000
600	600	375	344	1000
1000	1000	251	1000	777
1000	1000	757	1000	535
1000	1000	103	1000	377
1000	1000	186	750	1000
1000	1000	377	1000	633
1000	933	1000	1000	453
1000	1000	375	1000	377
1000	1000	757	1000	757
1000	1000	251	1000	720

train before quitting. Columns 2 and 4 show the actual number of generations for an experiment, non-optimized and optimized. Columns 3 and 5 show the corresponding fitness values for non-optimized and optimized models. Fitness values range from 0 to 1000.

After sorting each column of fitness values, the non-optimized are plotted against the optimized, as in Fig. 4. The bold line in Fig. 4, which represents the optimized results, suggests that optimizing techniques produce better results.

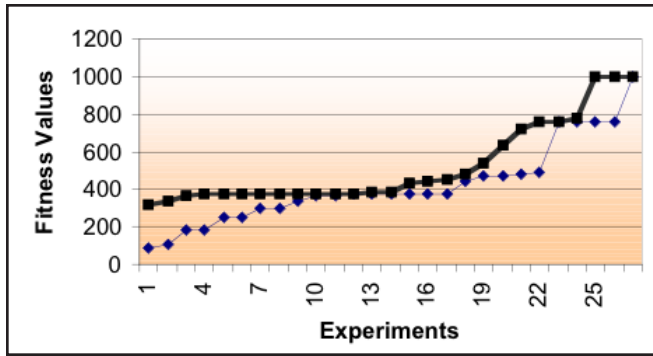


Figure 3. Visual Comparison of Non-Optimized and Optimized Results

Table 3. Experiment 1 *t*-test Results on Fitness Values

	Non-Optimized Fitness	Optimized Fitness
Mean	420.4444	523.6667
Variance	47547.49	48835.77
Observations	27	27
Pearson Correlation	-0.09545	
Hypothesized Mean Diff.	0	
df	26	
<i>t</i> Stat	-1.65067	
P(T <= <i>t</i>) one-tail	0.055418	
<i>t</i> Critical one-tail	1.705616	

Table 4. Experiment 1 *t*-test Results on Actual Number of Generations

	Non-Optimized Actual No. of Generations	Optimized Actual No. of Generations
Mean	597.5185	571.7037
Variance	108874	109730.1
Observations	27	27
Pearson Correlation	0.968145	
Hypothesized Mean Diff.	0	
df	26	
<i>t</i> Stat	1.607249	
P(T <= <i>t</i>) one-tail	0.060039	
<i>t</i> Critical one-tail	1.705616	

Performing a *t*-test determines whether this difference is statistically significant. These statistical results are displayed in Table 6.

The one-tail *t*-test produces a value of 0.004894. This implies that there is a statistically significant difference between the two means.

Next, the actual numbers of generations needed to solve the problem are examined for statistical significant difference. Table 7 shows these results.

Table 5. Experiment 2 Results

Maximum No. of Generations	Non-Optimized		Optimized	
	Actual No. of Generations	Fitness Values	Actual No. of Generations	Fitness Values
200	200	263	200	253
200	200	10	47	1000
200	200	8	200	276
200	200	49	200	312
200	200	94	200	686
200	200	70	29	1000
200	200	74	200	182
200	200	75	200	297
200	200	356	200	153
600	600	57	600	230
600	600	13	600	776
600	600	54	600	252
600	600	111	600	344
600	600	149	600	556
600	7	1000	600	522
600	600	285	600	230
600	600	595	7	1000
600	600	801	600	418
1000	1000	9	1000	258
1000	1000	91	1000	366
1000	1000	84	1000	120
1000	1000	70	1000	444
1000	1000	634	25	1000
1000	21	1000	1000	362
1000	1000	298	1000	192
1000	8	1000	188	1000
1000	1000	214	1000	876

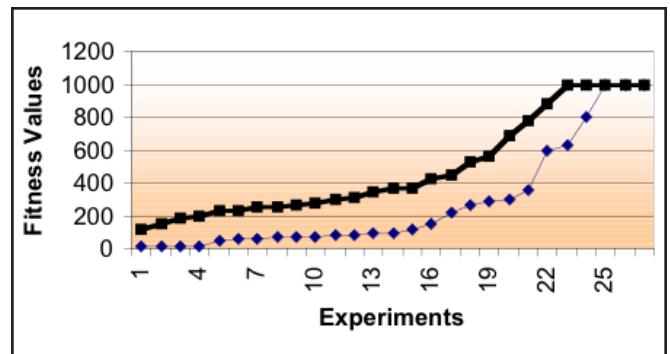


Figure 4. Non-Optimized and Optimized Results

The single tail *t*-test value of 0.466984 suggests that there is no difference between the non-optimized and optimized in terms of number of generations used to train.

Experiment 3

The third experiment uses a polynomial equation with nested functions. This experiment modeled the following equation:

$$e^{(\sin(\log(\text{Abs}(\cos(X))))))}$$

Table 6. Experiment 2 *t*-test results

	Non-Optimized Fitness	Optimized Fitness
Mean	276.4444	485.3704
Variance	109019.5	95599.55
Observations	27	27
Pearson Correlation	0.259424	
Hypothesized Mean Diff.	0	
df	26	
<i>t</i> Stat	-2.78774	
P(T <= <i>t</i>) one-tail	0.004894	
<i>t</i> Critical one-tail	1.705616	

Table 7. Experiment 2 *t*-test Results on Actual Number of Generations

	Non-Optimized Actual No. of Generations	Optimized Actual No. of Generations
Mean	505.037	499.8519
Variance	128995	132810.3
Observations	27	27
Pearson Correlation	0.603896	
Hypothesized Mean Diff.	0	
df	26	
<i>t</i> Stat	0.08366	
P(T <= <i>t</i>) one-tail	0.466984	
<i>t</i> Critical one-tail	1.705616	

This experiment is repeated 27 times in order to provide a sufficient amount of data for statistical analysis. Table 8 shows the results from these experiments. Column 1 represents the maximum number of generations the GEEP could train before quitting. Columns 2 and 4 show the actual number of generations for an experiment, non-optimized and optimized. Columns 3 and 5 show the corresponding fitness values for non-optimized and optimized models. The fitness values range from 0 to 1000.

After sorting each column of fitness values, the non-optimized are plotted against the optimized, as in Fig. 5. The bold line in Fig. 5, which represents the optimized results, suggests that optimizing techniques produce better results.

Performing a *t*-test determines whether this difference is statistically significant. These statistical results are displayed in Table 9.

The one-tail *t*-test produces a value of 5.84E-09. This clearly indicates that there is a statistically significant difference between the two means.

Next, the actual numbers of generations needed to solve the problem is examined for statistical significant difference. Table 10 shows these results.

The single tail *t*-test result of 0.0020 suggests that there is a statistically significant difference between the means. Thus,

Table 8. Experiment 3 Results

Maximum No. of Generations	Non-Optimized		Optimized	
	Actual No. of Generations	Fitness Values	Actual No. of Generations	Fitness Values
200	200	263	200	253
200	200	10	47	1000
200	200	391	200	537
200	200	391	200	710
200	200	391	200	537
200	200	391	200	537
200	200	391	100	1000
200	200	391	200	537
200	1	1000	20	1000
200	200	391	200	740
200	200	391	200	537
600	600	391	600	711
600	600	391	220	1000
600	19	1000	1	1000
600	600	391	150	1000
600	600	537	600	711
600	600	391	600	711
600	600	391	600	537
600	600	391	103	1000
600	600	391	38	1000
1000	1000	391	1000	711
1000	1000	391	1000	537
1000	1000	391	1000	711
1000	1000	391	18	1000
1000	1000	537	1000	782
1000	1000	537	1000	776
1000	1000	391	48	1000
1000	1000	391	327	1000
1000	1000	391	69	1000

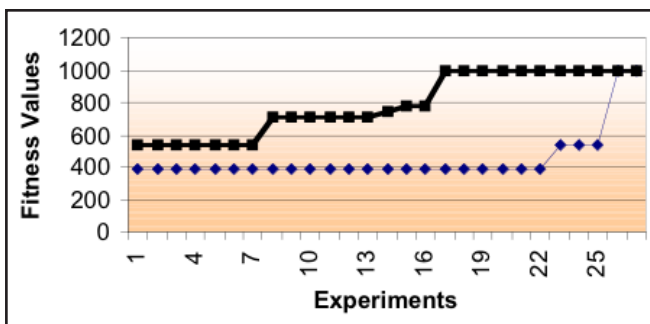


Figure 5. Non-Optimized and Optimized Results

for the runs in experiment 3, the optimization techniques did solve the equation faster.

Discussion

Two out of the three experiments produced statistically significantly better results and one out of three experiments solved the problem quicker. It is reasonable to expect that applying optimization techniques produces better and quicker results.

Analysis of the optimization techniques is revealing. As

Table 9. Experiment 3 *t*-Test Results

	Non-Optimized Fitness	Optimized Fitness
Mean	452.3333	789.7037
Variance	27082.38	37443.29
Observations	27	27
Pearson Correlation	0.291714	
Hypothesized Mean Diff.	0	
df	26	
<i>t</i> Stat	-8.17826	
P(T <= <i>t</i>) one-tail	5.84E-09	
<i>t</i> Critical one-tail	1.705616	

Table 10. Experiment 3 *t*-Test Results on Actual Number of Generations

	Non-Optimized Actual No. of Generations	Optimized Actual No. of Generations
Mean	571.1111	366.4444
Variance	130531.9	127.002.7
Observations	27	27
Pearson Correlation	0.555038	
Hypothesized Mean Diff.	0	
df	26	
<i>t</i> Stat	3.141409	
P(T <= <i>t</i>) one-tail	0.002082	
<i>t</i> Critical one-tail	1.705616	

Table 11. Assessment of Optimization Techniques

	Average Direct Duplicate Chromosomes	Average Indirect Duplicate Chromosomes
Experiment 1	23.55%	0.16%
Experiment 2	25.69%	0.93%
Experiment 3	34.67%	0.51%

displayed in Table 11, the average number of chromosome duplicates removed from a population ranged from 23.55 to 34.67 percent per generation. Perhaps the lack of a mutation rate contributes to this high redundancy.

A second observation regarding the results in Table 11 is the relatively low *Average Indirect Duplicate Chromosomes*. These numbers are derived from performing equation rewrites as a means of identifying equivalent equations. About five grammar rules were assessed for the experiments. Expanding the grammar to include over 100 rules will increase the number of equivalent chromosomes dramatically.

Conclusions

This work describes two optimization processes, one focused on chromosome optimization; the second, on population optimization. As the experiments become increasingly com-

plex, the optimization techniques become more important in producing better answers quicker. Applying optimization strategies is worth the overhead because better models are produced in less time.

Future Directions

As the GEEP paradigm expands, optimizing equations and populations will play a more significant role. Future optimization/modeling activities include:

- *Add other optimization techniques.* At times, the GEEP would settle in a local minima. Some techniques could be added to break out of these situations.
- *Explore other domains.* Future analysis could include more sophisticated operators from specific data domains.
- *Expand the equation optimization grammar.* There is extensive opportunity to expand the equation optimization grammar.

List of Works Cited

Andre, D. "Learning and Upgrading Rules for an OCR System Using Genetic Programming," *Proc.*, 1994 IEEE World Congress on Computational Intelligence, Orlando, FL, June 27-29, 1994.

Angeline, P. J. "Genetic Programming and Emergent Intelligence," in *Advances in Genetic Programming*. Ed. K. E. Kinnear, Jr. Boston: MIT Press, 1994. 4: 75-98.

Bettenhausen, K. D., S. Gehlen, P. Marenbach, and H. Tolle. "BioX++—New Results and Conceptions Concerning the Intelligent Control of Biotechnological Processes," in *6th International Conference on Computer Applications in Biotechnology*. Ed. A. Munack and K. Schügerl. N.Y.: Elsevier Science, 1995. 324-27.

Dempster, M. A. H. and C. M. Jones. "A Real-Time Adaptive Trading System Using Genetic Programming," *Quantitative Finance*. Vol. 1. London: IOP Publishing, Ltd., 2001. 397- 413.

Frick, A., R. Herrmann, M. Kreidler, and A. Narr. "Genetic-Based Trading Rules - A New Tool to Beat the Market With — First Empirical Results —," in *Aktuarielle Ansätze für Finanz-Risiken, Proc., 6th International AFIR Colloquium*, Nürnberg, Oct. 1-3, 1996, Ed. P. Albrecht. Verlag Versicherungswirtschaft e.V. Karlsruhe. Vol. I/II: 997-1018.

Gero, J. S. and V. Kazakov. "A Genetic Engineering Extension to Genetic Algorithms," *Evolutionary Systems* 9.1 (2001): 71-92.

Lucier, B. J., S. Mamillapalli, and J. Palsberg. "Program Optimization for Faster Genetic Programming," *Genetic Programming 1998: Proc., Third Annual Conference*, Madison, WI, July, 1998.

Koza, J. R., F. H. Bennett III, D. Andre, and M. A. Keane. *Genetic Programming III*. Morgan Kaufmann, 1999.

Koza, J. R., F. H. Bennett III, D. Andre, and M. A. Keane. "Automatic Design of Analog Electrical Circuits Using Genetic Programming," in *Intelligent Data Analysis in Science*. Ed. H. Cartwright. Oxford: Oxford University Press, 2000. 8: 172-202.

Mahfoud, S. and G. Mani. "Financial Forecasting Using Genetic Algorithms," *Applied Artificial Intelligence* 10 (1996): 543-65.



GEEP RESEARCHER—Dr. Boetticher meets with Kimberly Kaminsky. Ms. Kaminsky brought to the Genetically Engineerable Evolvable Program (GEEP) her background in business administration from San Diego State University where she earned her baccalaureate degree. In 2004, she earned an M.S. degree in computer science at the University of Houston-Clear Lake. She is currently enrolled in the doctoral program in Management Information Systems [MIS] in the Bauer College of Business at the University of Houston.

Miller, G. A. "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," *Psychological Review* 63 (1956): 81-97.

Oakley, H. "Two Scientific Applications of Genetic Programming: Stack Filters and Non-Linear Equation Fitting to Chaotic Data," in *Advances in Genetic Programming*. Ed. K. E. Kinneer, Jr. Boston: MIT Press, 1994. 17: 369-89.

Tackett, W. A. "Genetic Programming for Feature Discovery and Image Discrimination," in *Proc., 5th International Conference on Genetic Algorithms, ICGA-93*. Ed. Stephanie Forrest. University of Illinois at Urbana-Champaign, July 17-21, 1993. Morgan Kaufmann. 303-09.