
Genetic Programming and Evolvable Machines, 2, 165–191, 2001

What Makes a Problem GP-Hard? Analysis of a Tunably Difficult Problem in Genetic Programming

JASON M. DAIDA, ROBERT R. BERTRAM, STEPHEN A. STANHOPE,
JONATHAN C. KHOO, SHAHBAZ A. CHAUDHARY, AND OMER A. CHAUDHRI
The University of Michigan, Artificial Intelligence Laboratory and Space Physics Research Laboratory, 2455 Hayward Avenue, Ann Arbor, MI 48109-2143 USA

JOHN A. POLITO, II
Consilient, Inc., 1815 4th Street, Suite B, Berkeley, CA 94710 USA

Received August 2, 2000; Revised January 30, 2001

Abstract. This paper addresses the issue of what makes a problem genetic programming (GP)-hard by considering the binomial-3 problem. In the process, we discuss the efficacy of the metaphor of an adaptive fitness landscape to explain what is GP-hard. We indicate that, at least for this problem, the metaphor is misleading.

Keywords: problem difficulty, test problems, fitness landscapes, GP theory

1. Introduction

What makes a problem GP-hard? Unlike other areas in evolutionary computation, genetic programming (GP) has but a nascent body of theoretical work that has addressed this subject. Guidance for understanding what makes a problem difficult for GP has come from work and ideas in areas like genetic algorithms (GA). For example, one could take a cue from previous works in GA and posit that what makes for a GP-hard problem is what makes for a GA-hard problem—a rugged fitness landscape (a deceptive fitness landscape, a flat fitness landscape, etc.). As of this writing, however, GP theory has furnished only a few principles to guide practitioners about whether a problem is difficult (or easy). The ability to score the difficulty of a problem in advance of actually trying to solve it with GP has proven troublesome, if only because investigators have yet to identify all of the essential ingredients in creating a difficult problem for GP.

In place of theory, then, conventional wisdom in GP has suggested that what makes a problem difficult in GP is a problem's domain. For that reason, many empirical papers that address GP theory feature several different problems from several different domains. In recent years, researchers have moved toward an informal consensus in adopting several of these problem domains as being suitable for investigations in theory.

We have found in our investigations that perhaps for GP, neither prevailing notions of fitness landscapes nor intrinsic properties of a problem's domain have sufficient explanatory power to account for what makes a problem GP-hard. To accomplish this, we have investigated a tunably difficult problem that features the following: a statistically invariant combinatorial search space, a fixed fitness function, a fixed set of GP operators, a fixed function set, and a fixed terminal set.

1.1. Previous work

There are but a few theoretical works that address problem difficulty in GP at all. The first work to do so appeared in Koza [1], Chapter 8. In this work, Koza provided a semi-empirical formula that estimated the number of trials needed to solve a problem with a specified success probability. O'Reilly [2] attempted to extend fitness landscape analysis in GA research (i.e., [3, 4, 5]) to GP. Langdon and Poli [6] provided an alternative by proposing to sample a solution space (either exhaustively or using Monte Carlo methods) and applying this technique to a particular problem (i.e., artificial ant on the Santa Fe trail [1, 7].

A closely related issue involves GP test problems that are tunably difficult. As of this writing, the GP community has not had a well-recognized suite of test problems (along the lines of the De Jong [8] or Ackley [9] test suites in GA research). There have been several promising candidates, however, Koza [1] provided the first set of tunably difficult problems that have included the Boolean multiplexers and the Boolean parity functions (i.e., both even and odd parity). In his second book, Koza [10] included polynomials (a sextic and a quintic), Boolean symmetry (5- and 6-symmetry), Fourier sine series (3- and 4-terms), the lawnmower problem and the bumble-bee problem. Punch et al. [11] introduced a tunably difficult royal tree problem, which they have designed along the lines of the royal road problem [12] in GA. Gathercole and Ross [13] have proposed the MAX test suite, which they have developed along the lines of the ones–max problem [9] in GA research. Foster and his colleagues have offered the Maximum Clique problem for GP [14]. O'Reilly [15] developed another tunable problem along the lines of the GA Royal Road function. O'Reilly and Goldberg [16, 17] have also developed two other problems called ORDER and MAJORITY that have also been patterned after the ones–max problem.

For the most part, however, researchers have turned to examples from Koza's books [1, 10] in place of a recognized suite of test problems. Typical suites have included multiplexer, lawnmower, symbolic regression, and artificial ant (in [18]; Boolean parity, symbolic regression, artificial ant (in [19]); Boolean parity, sunspot, and intertwined spiral (in [20]). General domain themes have been to include a problem from each of the following categories; Boolean, symbolic regression, and finite-state machine.

1.2. About this paper

This paper describes the binomial-3 problem and presents its statistical portrait as the problem is tuned from relatively easy to relatively difficult. We show that under certain conditions, the problem scales logarithmically in difficulty, where difficulty is measured in terms of the hit-score metric. We present our analysis of this problem and describe the process by which this problem can be tuned. In doing so, the analysis challenges current views about what makes a problem difficult.

The conventional view for thinking about what makes any problem difficult for any EC method has been the metaphor of an adaptive landscape in evolutionary biology. The adaptive landscape, as posed in Wright [21], has suggested to EC practitioners an optimization of fitness in a multi-dimensional, multi-modal search space. A common idea in EC research has been that the adaptive landscape is primarily an external consideration, an environment, and that EC individuals "walk" on this landscape. This interpretation of the metaphor of an adaptive landscape is not without precedent in evolutionary biology. After all, Wright's illustration of adaptive landscapes looked like topographic maps (which Simpson commented upon in his seminal work [22, 49]). Noted neo-Darwinist Dobzhansky took Wright's figure of speech one step further and mapped Wright's abstraction of "hills" and "valleys" to *real* mountains and valleys (i.e., the San Bernadino Mountains, CA, USA) [23; 24, p. 294]. In essence, one can tell merely by looking at an environment how difficult that environment will be for adaptation—a task similar to assessing the Antarctic or the Amazonia for suitability of life. Likewise under the conventional view, one can tell how difficult a problem would be for GP *by inspection*. Perhaps, one could even rate GP problems in a way that is reminiscent of how rock climbers rate the difficulty of their climbs—with a single metric, regardless of who is doing the actual climb.

In his thesis, Jones [5, p. 46] correctly noted that this common idea of adaptive landscapes is fraught with pitfalls for EC. Instead, Jones proposed a one-operator/one-landscape view of fitness for GA. In this view, landscapes are directed graphs, the configuration and the traversal of which are determined by a particular operator (e.g., mutation). In a sense, Jones' proposal for a rigorous definition of a fitness landscape is that of constrained externality. In particular, problem difficulty is still primarily an external phenomenon. Problem difficulty is also a constrained phenomenon as well, since the determination of which topological environment a GA individual must traverse is determined by a GA's operators. By framing the fitness landscape as such, Jones and Forrest [25] were able to propose a metric of problem difficulty that was largely independent of a GA. Although there exist counterexamples to Jones and Forrest's metric [26], for the most they successfully provided a rule-of-thumb measure that predicts *a priori* problem difficulty, at least for GA.

However, GA behaviors are not necessarily precursors to ones that occur under GP (see [27]). Despite Jones and Forrest's measure, single metrics that describe the potential difficulty of a problem under GP has proven elusive to find. Instead, conventional wisdom in the field of genetic and evolutionary computation has asserted that epistasis is responsible for the difficulties that one encounters. For that reason,

it is a worthwhile to digress momentarily to consider epistasis and the role that it may play in determining problem difficulty.

Like many concepts in the field of genetic and evolutionary computation, *epistasis*—or "gene interaction"—is a term that has been borrowed from genetics. Unfortunately, like many of these borrowed terms, the borrowing has been imprecise and loose. (See [28]) for a discussion on how borrowed terms can overconstrain theoretical development in genetic programming.) Many practitioners in GP have come to think of epistasis as meaning one thing without ever realizing that they are evoking disparate meanings of the term.

Wade [29] describes that in genetics, epistasis has two distinct definitions. In molecular and biochemical genetics, epistasis involves a biochemical pathway, i.e., one gene is considered "epistatic" to another if the function of its product is conditional to another gene that operates on the same biochemical pathway. In statistical and quantitative genetics, epistasis is a population concept that describes a nonlinear relationship between phenotypic variations and their underlying genotypes. Epistasis accounts for the phenotypic variation among individuals that cannot be accounted for by an additive treatment of single loci.

The two definitions are *not* interchangeable. Biochemical epistasis can occur without ever resulting in population epistasis. Population epistasis *requires* genetic variation; biochemical epistasis does not. Population epistasis cannot occur between genetically identical individuals; biochemical epistasis can, regardless of whether individuals are genetically identical. Population epistasis is intimately associated with the fitness landscapes through Wright [21]; biochemical epistasis is not beholden to *any* notion of landscape. Population epistasis has direct implications for individuals to evolve in a changing environment; in biochemical epistasis, the role of an external environment is moot.

Not surprisingly, some of the earliest work in genetic and evolutionary computation defer to Wright's usage of epistasis. For example, even though Fraser [30] never really uses the term *epistasis*, he clearly employs it in the population sense of the word (i.e., as *inter-locus interactions*). Likewise Holland (1975), also uses the term in the population sense. To some degree, Jones and Forrest [25] do as well. Later works in GA, as in [31, 32] also presuppose this view. In GA, a bit string that is devoid of any environmental context is not meaningful—there is no sense that bits would interact with other bits until a fitness function is defined.

On the other hand, in GP, an individual that is devoid of any environmental context can still be evaluated. One can execute (though not score) a GP individual without any knowledge of its fitness function. In many instances, one can even anticipate node-to-node interactions, again without any knowledge of that individual's fitness function. For that reason, researchers in GP have correctly used the term epistasis to describe these node-to-node interactions. However, and this is key, the use of epistasis is in the biochemical sense of the term, *which is in conflict with prior usage of epistasis* in the field of genetic and evolutionary computation.

In this paper, we show that problem difficulty can largely be driven by factors that have usually been considered internal to an EC algorithm. In the binomial-3 problem, the "outside" is not the necessary component that determines problem

difficulty. A fitness function does not need to correspond to a "rugged" environment for a GP to encounter difficulty. Instead, the source of difficulty stems from "internal" conflicts involving content, context, and the emergent strategies that arise to quell them. It is in the process of solving the problem and not the problem itself that difficulty ensues. Perhaps difficulty for GP, then, is not so much pictured as a photograph from Ansel Adam's series "Sierra Nevada: The John Muir Trail" [33]—a portfolio of the soaring peaks and the deep valleys of the Sierra Nevada. Perhaps at least for some cases in GP, a more appropriate picture of difficulty would be Edvard Munch's painting "The Scream"—an oil depicting an internally tortured soul on what would otherwise be a fairly mundane landscape. In other words, we would claim that the metaphor of population epistasis, as has been traditionally used in the genetic and evolutionary computation community, is not the appropriate framing to understand problem difficulty in GP. We would, however, reinforce the use of the metaphor of biochemical epistasis, and claim that node–node interactions play a significant role in determining problem difficulty that is distinct and separate from that of a fitness landscape.

2. Experiment description

This section describes our experiment and includes a description of the binomial-3 problem.

2.1. Binomial-3 problem description

The binomial-3 problem is an instance taken from symbolic regression and involves solving for the function $f(x) = 1 + 3x + 3x^2 + x^3$. The term "binomial" refers to the sequence of coefficients in this polynomial; the "3" refers to the order of this polynomial.

We define the binomial-3 problem as follows. Fitness cases are 50 equidistant points generated from the equation $f(x) = 1 + 3x + 3x^2 + x^3$ over the interval $[-1, 0)$. Raw fitness score is the sum of absolute error. A hit is defined as being within 0.01 in ordinate of a fitness case for a total of 50 hits. The stop criterion is when an individual in a population first scores 50 hits. Adjusted fitness is the reciprocal of the quantity one plus raw fitness score.

A function set is a subset of $\{+, -, \times, \div\}$, which corresponds to the arithmetic operators of addition, subtraction, multiplication, and protected division. We define protected division as the operator that returns one if the denominator is exactly zero. Typical function sets include $\{+, -, \times, \div\}$, which we presume for this paper. Other sets may include other permutations such as $\{+, \times\}$ or $\{-, \times\}$.

A terminal set is a subset of $\{X, \mathsf{R}\}$, where X is the symbolic variable and R is the set of ephemeral random constants (ERCs). We presume that the ERCs are uniformly distributed over a specified interval of the form $[-a_{\mathsf{R}}, a_{\mathsf{R}}]$, where a_{R} is a real number that specifies the range for ERCs. We require that each ERC is generated but once at population initialization and *is not changed in value during the*

course of a GP run. Typical terminal sets include either $\{X\}$ (a binomial-3 problem without ERCs) or $\{X, \mathsf{R}\}$ (a binomial-3 problem with ERCs). For example, a small population of two individuals consisting of two terminals apiece could have as a terminal set $\{X, -0.1, 0.3, 0.8\}$. It would not be unusual to have a terminal set consisting of X and several thousand terminal constants for a population size in the range of several hundred individuals.

Tuning is achieved by varying the value associated with a_{R}. We defer until Section 4 the discussion of how a_{R} affects problem difficulty *without* changing the combinatorial search space.

2.2. *Binomial-3 problem background*

The binomial-3 problem shares many properties that are common to other problems in GP. It requires symbol manipulation. It allows for *nocs* (i.e., non-coding segments, also known as *introns* or unexpressed code). It affords GP to choose from multiple approaches to solve for the same problem. Of these properties, the latter two warrant further explanation.

The problem allows for several types of nocs, some of which involve the 3-tuple structure $(-X, X)$. Multiplication of this structure to any other results in a value of zero. Division by this structure to any other results in a value of 1. We note that other types can be derived or are similar to these basic two.

The problem affords GP to choose from multiple approaches. For example, equivalent solutions include $(1 + x)^3$, $(1 + x)(1 + 2x + x^2)$, $(x - -1)^3$ and $(x + 1) \div (1 \div (1 + (x \div 0.5) + (x \div (1 \div x)))$. In addition to these equivalent approaches, there exists a number of approximate approaches (e.g., rational polynomials that fit all 50 points, but not necessarily anywhere else). There are several ways to generate numerical coefficients as well. For example, the coefficient 2 can be generated by using an ERC that (approximately) equals this value. It can be generated with the value 0.5 and taking the reciprocal of that value. It can also be generated through distribution, e.g., $(x + x)$. We surmise that the total number of ways to solve the binomial-3 problem to be on the order of a few thousand (i.e. see [34]).

The choice of coefficients, form, and order of the target function $f(x)$ for the binomial-3 problem was purposeful and deliberate. The use of $f(x) = (1 + x)^3$ has allowed for an extended mathematical treatment [34].

The binomial-3 problem does not share an antecedent with a related test problem in GA research, but its domain has an extended history in GP. One of the earliest, intuitive applications of GP has involved data modeling under the moniker of symbolic regression. In [1], symbolic regression has been synonymous with function identification, which involves finding a mathematical model that fits a given data set. Closely linked problems have included sequence induction, Boolean concept learning, empirical discovery, and forecasting. Typically, practitioners use GP and symbolic regression in several ways: as a benchmark problem to test GP systems, as a software demonstration or tutorial, and as a means of generating mathematical models for real-world domains. The latter area includes applications in control systems, bioengineering, biochemistry, image compression, and finance.

In spite of these works, we recognize that from a purely practical standpoint, there exist modifications to standard GP that may be better suited for data modeling. This seems to have been particularly true in the generation of parameter constants, which standard GP does awkwardly with ERCs. Recent developments in GP indicate methods that appear to generate constants with greater efficacy than as with using ERCs (e.g., [35, 36, 37]).

Our interest in using ERCs stems from their worth in illustrating fundamental processes in GP dynamics. ERC values can serve as tracers that allow tracking of individual nodes, if each ERC value is unique and generated just once. ERCs can also be used to address building block issues, as we have done in [34].

2.3. Experiment procedure

We used a patched version of lilgp [38] to generate our data. Most of the modifications were done for bug fixes, as well as to add other features for use in other experiments (e.g., strong typing and population initialization). The patches came from three sources: Luke, Andersen, and Daida. Luke's patches consist of memory leak fixes, multi-threading bug fixes. His enhancements also include provisions for strong-typing (which we did not use) and population initialization. Andersen's fixes included patches to Luke's population initialization routine, so that population initialization could include integer-valued ERCs. Our patches include modifications to the population initialization routine, so that population initialization could include real-valued ERCs. We also replaced the random number generator (RNG) in lilgp (Knuth subtractive RNG) with the Mersenne Twister [39, 40]. The Mersenne Twister has excellent mathematical properties that make this RNG a reasonable candidate for theoretical work in GP. (See [41–43] for issues concerning RNGs.) We configured lilgp to run as a single thread, to mitigate against possible artifacts introduced by parallelizing an RNG. We note that lilgp supports the use of ERCs and that ERCs in lilgp are generated once at population initialization. For all practical purposes, all ERC values generated at population initialization are unique, with every ERC value having just one instance in an initial population.

Most of the GP parameters were identical to those mentioned in [1, chapter 7]: population size = 500; crossover rate = 0.9; replication rate = 0.1; population initialization with ramped half-and-half; initialization depth of 2–6 levels; and fitness-proportionate selection. Other parameter values were maximum generations = 200 and maximum tree depth = 26 (Note: these last two parameters differ from those presented in [1], which specified a maximum number of generations = 51 and a maximum depth = 17. Part of the reason we extended these parameters was to delay possible effects that occur when GP processes individuals at these limits.)

The main experiment involved varying the tuning parameter a_{R}. We used seven values of a_{R} : 0.1, 1, 2, 3, 10, 100, 1000. We also ran one control with no ERCs. Eight data sets were collected in all: Control (No ERCs), Tenth (ERC: $[-0.1, 0.1]$), Unity (ERC: $[-1, 1]$), Two (ERC: $[-2, 2]$), Three (ERC: $[-3, 3]$), Ten (ERC: $[-10, 10]$), Hundred (ERC: $[-100, 100]$), and Thousand (ERC: $[-1000, 1000]$). Each data set consisted of 600 trials for a total of 4,800 runs for the main part.

We did another, albeit limited, experiment that involved fixing the tuning parameter a_R, but varying the population size. In this experiment, we collected two data sets with $a_R = 1$, but with population sizes of 50 and 5000, respectively. All other parameters were set as in the main experiment. Each data set consisted of 600 trials for a total of 1200 runs for the main part.

We note that the total amount of computation represented by both experiments is specified as 1,086,000,000 GP individual evaluations, which is equivalent to about 21,720 trials of typical size in the GP community (e.g., 50 generations, population size 1,000). This figure approaches the amount of computation indicated in [18]. All trials were run on Sun Ultra workstations.

3. Results

Table 1 summarizes the best-of-trial results of the experiment. The best possible score is 600. Throughout the course of this paper, we used perfect, upper decile, and upper quartile hit-score measures of problem difficulty.

The inclusion of ERCs as a whole increased problem difficulty. Without ERCs, the binomial-3 was an easy problem to solve, with five out of six trials resulting in a perfect score. We note that for the most part, the general trend is that if $a_R \geq 1$ and a_R increasing, the problem becomes increasingly more difficult to solve. (That trend does not hold for $0 \leq a_R < 1$). Figure 1 plots the results for $a_R \geq 1$ in Table 1, with hit scores normalized to 100%. The regression coefficient is -0.997.

Table 1 and Figure 1 represent just three slices of the best-of-trial distribution associated with each data set. Subsequently, Figure 2 shows the full distribution of hit scores per data set. The distributions are generally unimodal.

Figure 3 summarizes the results from the following data sets: Control, Tenth, Unity, Two, Three, Ten, Hundred, and Thousand. Each plot shows 600 points, with each point corresponding to a best-of-trial individual. Rows are arranged by data set.

In creating the plots for the second and third columns, we added a small amount of uniform random noise to both (x, y) coordinates of each point. We did this for

Table 1. The total number of trials (out of 600 trials) that scored perfectly in the upper decile and in the upper quartile

a_R	Perfect	1 Decile	1 Quartile
None	502	515	546
0.1	14	42	130
1	219	329	463
2	144	285	433
3	105	239	390
10	57	145	312
100	9	32	104
1,000	3	4	5

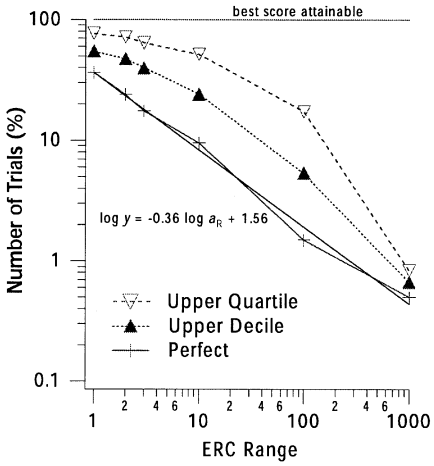


Figure 1. Tuning of hit score vs a_R. This log–log plot shows the relationship between the tuning parameter a_R and the hit score. The problem becomes progressively more difficult as a_R increases.

visualization only. The quantities corresponding to node count, depth, and generation are integer values—because of this, a single dot could correspond to many data points. The noise was added to displace points visually away from each other. That technique was not repeated for the first column, if only because adjusted fitness is a real-, not integer-valued quantity.

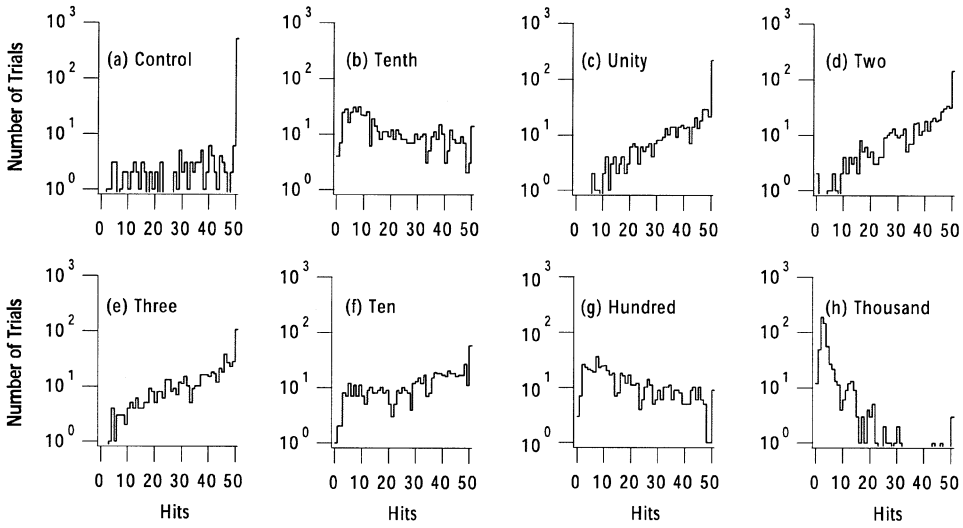


Figure 2. Distribution of hit scores vs a_R. Since the distributions are unimodal, the abbreviated summaries presented in Table 1 and Figure 1 are indicative of their corresponding distributions.

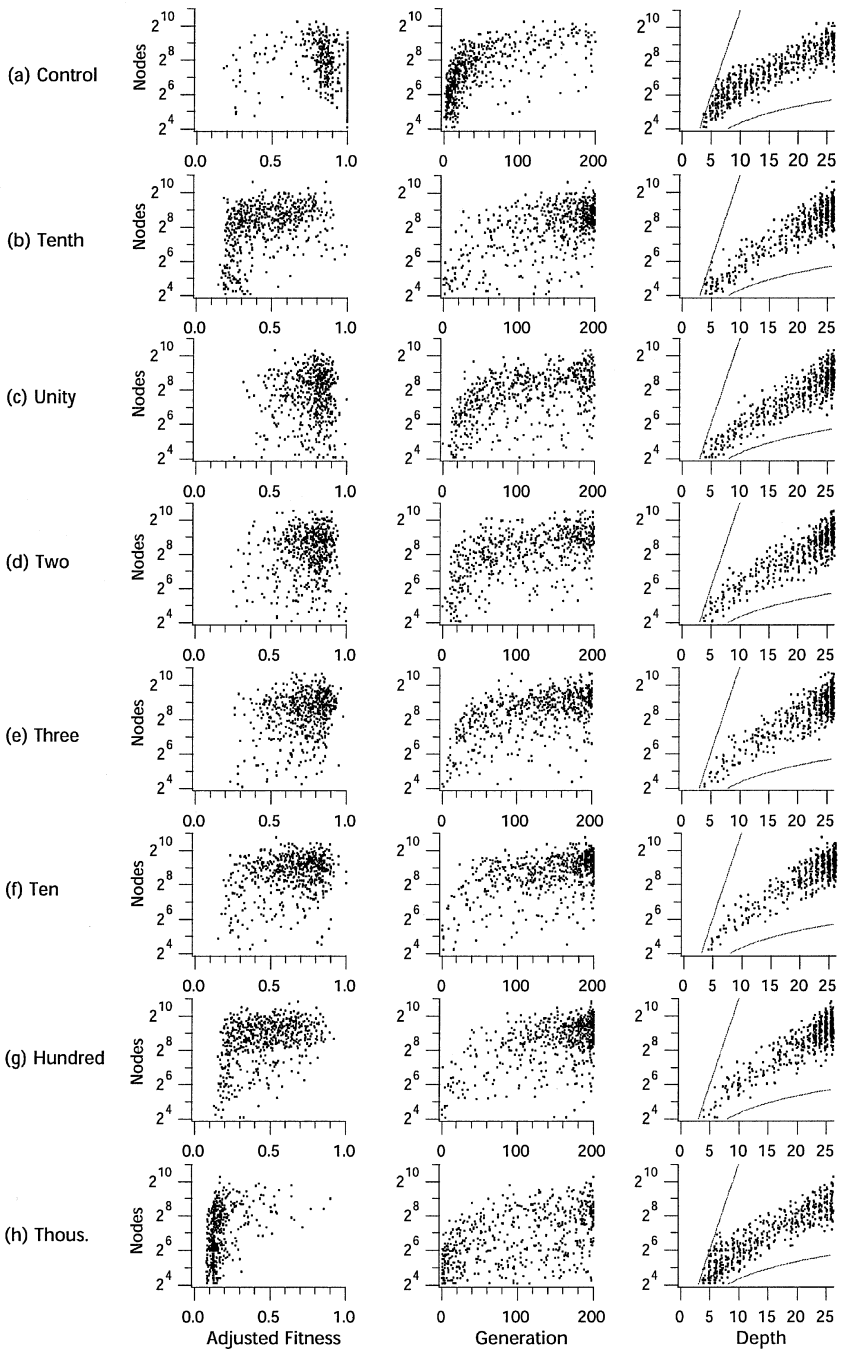


Figure 3. Best-of-trial results. Definite patterns in individual size and shape are correlated to a_R. Each row summarizes a data set, where each data set consisted of 600 trials. This figure shows the effect of increasing a_R on the size and shape of best-of-trial individuals.

The first column of Figure 3 shows the effect of ERC range concerning node count versus adjusted fitness. From Unity to Thousand data sets, the cluster of points appears to progress from right to left (higher to lower fitness). The results from the Tenth data set appear similar to the results from the Hundred data set. The vertical line of data points in Control corresponds to those best-of-trial individuals that had perfect adjusted fitness scores.

The second column of Figure 3 shows the effect of ERC range concerning node count versus the generation in which the best-of-trial individual was identified. Note that the individuals that occur near generation zero are generally concise and have likely required less computational effort to generate than those solutions near generation 200. From Unity to Hundred, the cluster of points appears to progress toward the right. That overall pattern breaks down for Thousand. The pattern for Tenth is similar to that for Hundred.

The third column of Figure 3 shows the effect of ERC range concerning node count versus the depth of the best-of-trial individuals. The lines indicate the upper and lower bounds for the numbers of nodes that can be present in a tree for a certain depth. From Unity to Hundred, the cluster of points appears to progress toward the right. That overall pattern breaks down for Thousand, which appears more like Control. The pattern for Tenth is similar to that for Hundred.

Figure 4 summarizes the results from the experiment that involved fixing the tuning parameter a_R, but changing the parameter for population size. One could consider the limited experiment as population variations on Unity. Each plot shows 600 points, with each point corresponding to a best-of-trial individual. Rows are arranged by population size. The layout and method of visualization for this data is similar to that of Figure 3.

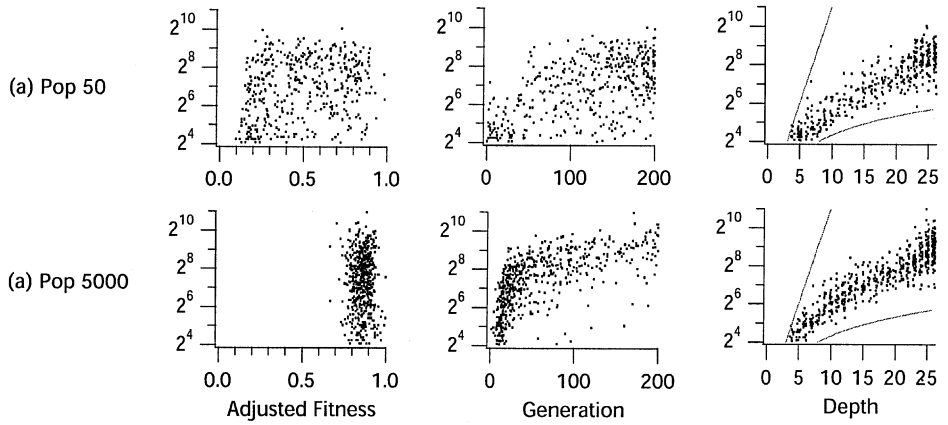


Figure 4. Effect of population. Although there are variations in pattern, the attractors that are associated with $a_R = 1$ do not change in position. Each row summarizes a data set, where each data set consisted of 600 trials. This figure shows the effect of increasing population size ($a_R = 1$) on the size and shape of best-of-trial individuals.

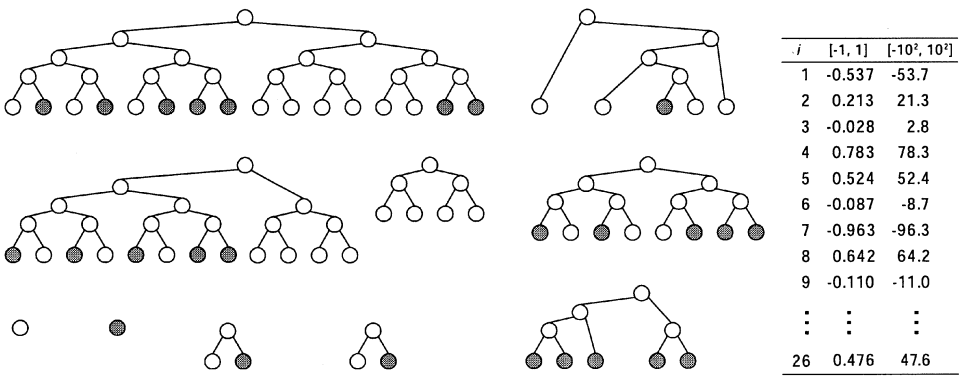
4. Discussion

4.1. Tuning

The experiment demonstrates clearly that the problem difficulty can be tuned by means of varying $a_{\mathtt{R}}$. As shown in Table 1 and Figure 1, the hit scores for perfect, upper decile, and upper quartile were monotonically decreasing for increasing $a_{\mathtt{R}}$ for $a_{\mathtt{R}} \geq 1$. The hit scores for perfect are well described with a log–log regression fit. That the findings are representative of the data as a whole is supported by the distributions of Figure 2, which indicated that perfect, upper decile, and upper quartile are representative for the binomial-3 problem.

The crux of this paper addresses why the binomial-3 problem is tunable in this way. A reasonable notion associated with increasing $a_{\mathtt{R}}$ is that GP needs to sort through an increasing number of ERCs. Consequently, the problem becomes more difficult because there are that many more ERCs from which to choose. We show otherwise by examining our claim that the combinatorial search space remains statistically invariant even though $a_{\mathtt{R}}$ varies.

As it turns out, the specification of ERCs as we have alluded to for the binomial-3 problem suggests the following (typical) implementation. Let there be two terminal types X and `r`, where X is a symbolic variable and `r` a terminal of type ERC. From the perspective of the user, this is what is typically specified, as opposed to X and the N different terminals of constants that have a unique value. The terminal `r` serves as a token, a placeholder. Instead of managing N different terminals, GP manages one terminal type, `r`, which references a list of ERC values by means of an index set (e.g., a hash table). In essence, GP operates on X and a set of tokens whose values are determined elsewhere. Consequently, by changing $a_{\mathtt{R}}$, what is changed is not the number of tokens, but the table of lookup values that are assigned to those tokens. Figure 5 shows an example of this in a hypothetical population. The grayed circles represent tokens. The accompanying table shows ERC values that occur for two different ranges of $a_{\mathtt{R}}$. (Note: in actuality, two different random number



i	$[-1, 1]$	$[-10^2, 10^2]$
1	-0.537	-53.7
2	0.213	21.3
3	-0.028	2.8
4	0.783	78.3
5	0.524	52.4
6	-0.087	-8.7
7	-0.963	-96.3
8	0.642	64.2
9	-0.110	-11.0
⋮	⋮	⋮
26	0.476	47.6

Figure 5. Illustration of statistical invariance in combinatorial search space. Although the values change, the number of ERC tokens do not in this hypothetical population. ERC tokens are denoted in gray. ERC values are shown in the accompanying table.

seeds were chosen for each ERC range to generate two independent samples.) The *combinatorial* search space remains statistically invariant even though a_R varies. (For example, the number of ERCs allocated for a population of 500 individuals, regardless of a_R, was roughly $4{,}500 \pm 400$.) In other words, the total number of structural permutations that are possible does not change (although the transition matrix from one set of possible structures to another may change).

4.2. Internal conflict: content and context

If the combinatorial search space remains statistically invariant, and if fitness function, function set, and specifications for crossover and replication remain constant, what causes the problem to vary in difficulty?

In posing a problem like the binomial-3, we have shifted away from linking problem difficulty with problem scalability. Examples of scalable genre include parity and multiplexer problems (which increase in difficulty with increases to the number of inputs). Instead, we have linked problem difficulty with terminal selection, in which the task is to choose the most appropriate set of terminals out of a large set to solve for the problem. Genres like these also have practical implications for real-world applications.

Without knowing the results presented in Section 3, one could reasonably hold the expectation that the binomial-3 problem would actually get *easier* as a_R increases. Intuitively, this would make sense. It is easier to visualize how the value 1 makes more sense in solving for $(x + 1)^3$ than the value 1,000. Clearly, the "obviously wrong" values would be selected against. For example, [1] describes the Biathlon, in which GP addresses two completely unrelated problems in the course of a single run. In each version of the Biathlon, the problem changed from symbolic regression to artificial ant. Only a single function set and a single terminal set was provided. What was claimed was that GP was able to solve for each problem, in spite of a large number of irrelevant functions and terminals.

By positing the hypothesis that the binomial-3 problem would actually get *easier* as a_R increases, one is arguing that *content matters* in what makes a problem GP-hard. While for many GP practitioners this makes reasonable sense, in the larger scope of EC theoretical research on fitness landscapes, *the linkage is not obvious*. Terminal content is a matter that arguably goes beyond the operator and directed graph formalism of fitness landscapes. Furthermore, it would also mean that one intrinsically binds the concept of fitness landscapes to not just the fitness function and parse-tree representation, but to the components used to solve for the fitness function. In other words, one could recreate a fitness landscape, albeit one specific to GP, by either an exhaustive or Monte Carlo sampling of random parse tree programs created from program components.

We would agree that content matters. *However, we would also argue that content alone does not determine problem difficulty*. After all, the binomial-3 problem became *harder* as a_R increased. We posit that *context* also matters and that context is an emergent by-product of GP processing.

To a GP system working with X and N random tokens r, at the outset, all values corresponding to r are equally valid. It is only after a few iterations of GP that any values of r gain any meaning (worth) towards solving the problem. Anything that confounds moving toward a common meaning for a value of r hinders selection, since the worth ascribed is inconsistent. What drives inconsistency is the context of an ERC value in a parse tree. Figure 6 illustrates two common inconsistencies that can arise.

Figure 6(a) shows the inconsistencies that arise when the context of an ERC value switches from a noc to a functional expression. In this example, there are two ERC tokens $\mathtt{r}_1$ and $\mathtt{r}_a$, where $\mathtt{r}_a$ is not expressed in Parent 1 and $\mathtt{r}_1$ exists as a part of Parent 2. In this hypothetical example, $\mathtt{r}_1$ can occur in the next generation as part of either of two possible children. The possible tree fragments are functionally equivalent to $(x + r_1)$ or $(x + 1)$. We assume that $(x + 1)$ is a desired fragment towards the solution of the problem. In either possible child, the meaning of $\mathtt{r}_1$ is conflicted: in one instance $\mathtt{r}_1$ means nothing and in the other instance $\mathtt{r}_1$

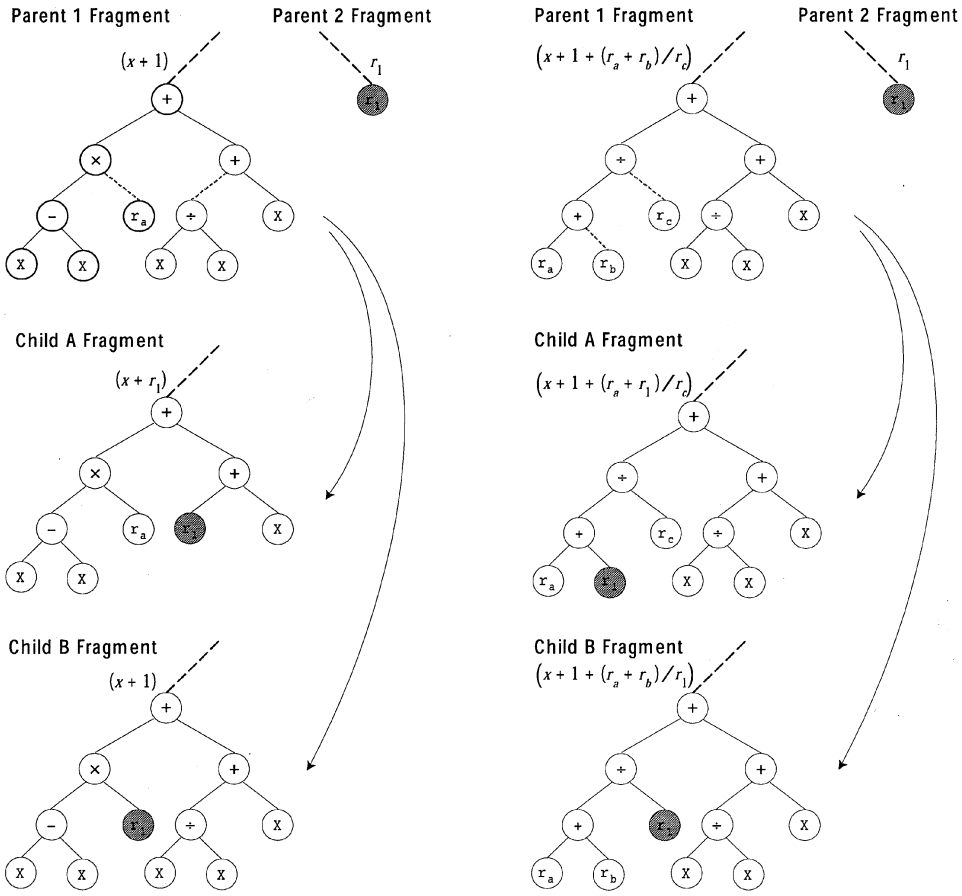


Figure 6. Illustration of two different kinds of context shifts. The context of ERC helps to determine its functional meaning: (a) for nocs; (b) for division.

appears in the expression of the tree fragment. We note that the magnitude of this conflict increases as $\mathtt{r}_1$ increases. The probability of this occurring increases as the range increases. For example, a value taken from the range $[-1, 1]$, say 0.9, appears alternately as $(x + 1)$ or $(x + 0.9)$. In contrast, a value taken from the range $[-1000, 1000]$, say 999.9, appears alternately as $(x + 1)$ or $(x + 999.9)$.

Nocs probably represent the most dramatic way an ERC value can result in inconsistencies. However, there are other ways that produce such conflicts (e.g., as shown in Figure 6b). As in the hypothetical example depicted in Figure 6(a), the meaning of $\mathtt{r}_1$ is conflicted: in one instance, $\mathtt{r}_1$ appears in the numerator and in the other instance, $\mathtt{r}_1$ appears in the denominator. We note that as in the previous example, the magnitude of possible conflict can increase as either $r_1 \geq 1$ and r_1 increases or $|r_1| < 1$ and r_1 decreases. We further note that Figures 6(a) and (b) represent just two of several avenues in which inconsistencies can arise in trying to ascribe worth to an ERC token.

We point out that context-driven inconsistency is not an either/or proposition. As a GP run progresses, it is not uncommon for "relatives" to exchange subtrees, which results in multiple instances of single token. We have shown in [34] that what starts out as a single instance of a particular token value at generation 0 can result at the end of a GP run, 10^3–10^4 instances of that same value. Not surprisingly, then, the same ERC value can simultaneously exist on both sides of an inconsistency.

Taken in a different perspective, context-dependency is a consequence that can occur as a result of crossover. Large swings in meaning can significantly affect the functional meaning of an individual and these swings can be either beneficial or deleterious. It is these deleterious swings that other researchers have labeled as "destructive" crossover. In a sense, varying a_{R} varies the destructive effect of crossover.

Evidence for the phenomena that we have described can be seen in Figure 3. Works by others have indicated general trends when destructive crossover has taken place. The amount of nonfunctional code increases with the destructiveness of crossover; the nonfunctional code serves as a sort of buffer. Consequently, an increase in destructive crossover tends to increase the amount of nonfunctional code, which in turn creates for larger and deeper individuals [44, 45]. Thus trends in program size and shape shown in Figure 3 support this. Shorter best-of-trial individuals tended to occur earlier in a GP-run; larger best-of-trial individuals tended to occur later. As the difficulty of the problem increased, the runs generally took longer and the programs were larger (Figure 3, column 2). Likewise, as the difficulty of the problem increased, the programs were deeper (Figure 3, column 2).

Researchers have also argued that there are limits to this buffering effect and that there are emergent processes that occur as GP evolves individuals toward the depth limit, in part because of this code growth [44–46]. In Figure 3 column one, the trend in adjusted fitness for Unity–Hundred showed the distribution moving gradually from high fitness to low fitness. However, in Thousand, we not that the pattern for adjusted fitness collapsed; the pattern for generations became inchoate, and the pattern for depth no longer followed the general trend. We suggest that Thousand represents a case where the buffering effect, as well as associated emergent processes, was no longer able to overcome the destructive effect of crossover.

That context and content ultimately lie at the root of the destructive effect of crossover is indicated in Figure 3, Tenth. Figure 6(b) represents the case where context switching between numerator and denominator can be significant, particularly for values of $|r| \ll 1$.

For some, the issues of context and content are fairly obvious. For example, one could pose the binomial-3 problem as a search and selection of suitable "instructions": the better suited instructions are for solving a problem, the easier a problem is to solve. Such concepts have been explored in [47, 48]. While work like Levin's has bearing on the binomial-3 problem, of interest to us has been how these seemingly obvious concepts play out in GP dynamics. There are definite patterns in shape and size of individuals that have occurred that are not fully explainable by these early works concerning Kolmogorov complexity. Only recently has quantitative evidence been found for O'Reilly, and Oppacher's [49] conjecture that both context and content of subtrees matter [15–17]. Indeed, the results shown in Figure 3 and in [34] also substantiate O'Reilly and Oppacher's conjecture. Furthermore, our results provide quantitative evidence that links context and content with the phenomenon of destructive crossover. *Our results strongly suggest that destructive crossover and disruption are related, but not identical phenomena.* Under O'Reilly and Oppacher's conjecture, it is crossover that can destroy the organization of subtrees, consequently disrupting their meaning. However, in the case of single-node subtress, crossover within a leaf node is not allowable. What is left, instead, is the disruption of meaning that occurs when moving a wholly intact leaf node to another part of an individual.

We also note that these effects of context and content represent an internal consideration. They indicate that what makes a problem GP-hard is not solely an external consideration—i.e., an environment. The fitness function did not need to be "rugged" for GP to encounter difficulty. Moreover, it was a process of solving for the fitness function and not intrinsically the fitness function itself in which difficulty occurred. It is for reasons like these which beg the metaphor of landscapes in describing what makes a problem GP-hard (See [34, 49]).[1]

4.3. Emergent strategies of meaning

Is syntactical representation—which is often implicit in EC landscape theories—sufficient in describing what makes a problem GP-hard? Perhaps our current framing of context and content is incorrect. Perhaps we could recover aspects of the landscape metaphor by framing functions and terminals as part of the external environment, i.e., a landscape of content and context. Perhaps what needs to be articulated is a local syntactical neighborhood of content and context and not, as we have proposed, an alternative metaphor.

To a certain degree, it may be possible to consider context and content as purely syntactical considerations. For example, $(x + r_1)$ and $(x \times r_2)$ can be viewed as an injective algebraic mapping from the set of reals to reals. Changing syntax can change their mapping, i.e., $(x + x)$ and $(r_1 \times r_2)$. Given this interpretation, one can design a "neighborhood" based on various permutations of x, r_1, and r_2. Nevertheless, we would assert that syntax would account for only a part of the phenomena observed.

In Section 4.1, we introduced the notion that the combinatorial search occurs over ERC tokens, rather than ERC values. The notion is a fairly general one—we can extend this notion to the other elements of the function and terminal sets. Combinatorial search space would therefore translate as a search through permutations of various tokens in parse trees. However, what determines each token's *meaning?*

We speculate that a token's meaning and an individual's fitness (relative to a GP solving a specified problem) are not tantamount to each other and are therefore not bound solely by considerations of representation. Not only does GP work with syntactical traversals, say from $(x + r_1)$ to $(x + x)$, but GP also needs to determine workable meanings to the tokens that correspond to x, $+$, and r_1. Furthermore, each token's meaning may or may not have anything to do with an individual's fitness. We would subsequently consider a token's meaning to be a by-product of processes like recombination and selection in GP.

Before we continue, we should note that work described in this paper does not directly address the issue of the evolution of meaning for GP tokens. Nevertheless, this work does offer clues that suggest this direction of inquiry.

One can consider this work's experiments as exercises in GP "determining" the meaning of various ERC tokens. Each trial consists of several thousand unique ERC tokens. Some of these tokens represent values that are more meaningful to solving the binomial-3 problem than would other values. Intuitively, one would expect that values close to "1" would be more meaningful than say a value of "1000" in solving for $(x + 1)^3$. Intuitively, one would expect that the GP selection process would sift for values close to "1." If this were true, GP would have to solve not one, but two problems. One problem involves creating a mathematical model such that this model fits the supplied data points. This problem is the one a user specifies. The other problem involves creating error-correcting mechanisms to deal with errant ERC values that are not used or needed for a solution.

The error-correcting problem is an emergent one that GP would need to address to solve for $f(x)$. We can illustrate this need with the following scenario. Let $f^a(x)$ be an individual in a GP population. Furthermore let $f^a(x) = f(x) + r$, where r is an ERC with a value of 5. GP can obtain the desired solution $f(x)$ in the next generation by *eliminating* r from an individual, i.e., by exchanging r with a subtree that evaluates to zero. Alternatively, GP might eventually obtain $f(x)$ by *eliminating* r from a population, i.e., by placing r in individuals that are increasingly less likely to reproduce in subsequent generations. GP can obtain $f(x)$ by *absorbing* r, i.e., multiplying that ERC with a subtree that evaluates to zero. Finally, GP can obtain $f(x)$ by *incorporating* r into another individual $f^b(x)$, such that $f(x) = f^b(x) + r$. In this scenario, either elimination, absorption, or incorporation represent error-correcting mechanisms that deal with errant ERC values.

We can partially illustrate this emergent determination of meaning by showing two things: evidence of sifting and selection of certain ERC values over other ERC values and evidence of a shift in strategies between Control and Unity. Concerning sifting and selection of ERC values, we note that all we would need to do is show that some ERC values are preferred over others. This preference would show up quantitatively as increases in the number of tokens that correspond to certain values—an increasingly "meaningful" token would result in increasing numbers of

that particular token as a GP trial progresses. All ERC values have exactly one token at start; any substantial increase of that number at the end of a GP trial would suggest at least some utility of that token in solving for $f(x)$. Furthermore, while it seems commonsensical that the ERC values close to the absolute value of "1" would appear to be most "meaningful," it is possible that entirely different ERC values are considered meaningful between various GP individuals. Evidence for consistent "meaning" would be reflected in the sifting and selection of certain ERC values not at the scale that observes individuals, but at, say the scale that observes trends across many populations.

Concerning a shift in strategies, we note that Control represents a circumstance in which the value "1" is relatively easy to attain, e.g., $(x \div x)$. It is also easy to show that it is possible to solve for $f(x)$ without having to resort to any constants whatsoever. The introduction of ERC tokens in Unity increases the likelihood that GP would need to respond with error correction. Error correction would result in a pronounced shift in approaches used in solving for $f(x)$. (We note that the terminal set for Control is a subset of that in Unity. It is entirely possible for there to be no shift in the approaches taken by the best-of-trial individual in Unity.) To show this shift quantitatively, we would need to classify the types of GP best-of-trial individuals and, subsequently, show the change in the number of individuals that populate these classes between Control and Unity. Such a shift would yield empirical evidence for the existence of the emergent error-correcting mechanisms.

As an aside, we mention that even for this modest amount of evidence, the effort required to obtain it has been involved. Indeed, most of the software development implicit in this paper lay not in the development of the binomial-3 problem, but in the development of custom tools that could aid in our collection, management and analysis of data (which currently amounts to a few gigabytes). For example, to collect evidence of sifting, we designed software that would detect in which generation a best-of-trial individual would be found, and then set up scripts which would rerun all 600 trials in Unity such that a "snapshot" would be taken of the entire population in which a best-of-trial individual appeared. After the Unity trials were rerun, another software tool parsed each population snapshot to extract counts for each ERC token—the counts were summarized in a file that contained token counts for all Unity trials. A third software tool binned and visualized the token counts.

To collect evidence on a strategy shift for just the 50-hit best-of-trial individuals for Control and Unity, we designed a software tool that could algebraically interpret and subsequently translate a best-of-trial individual to a *Mathematica* expression. This step was necessary and non-trivial, since *all* instances of divide-by-zero had to be replaced with "1." It was impractical to do this step by hand—a best-of-trial individual could easily span several pages of type and there were several hundred individuals to analyze. These *Mathematica* expressions were then algebraically simplified, factored, and plotted. Again we would add that the simplifications resulted in non-trivial expressions, some of which were rational polynomials equal to or greater than the order of 50. We subsequently classified each expression by hand, which meant perusing over 600 pages of polynomials.

Figure 7 shows a histogram of all ERC values from 30,000 individuals (500 individuals per population snapshot per trial, 600 trials). Each population snapshot was

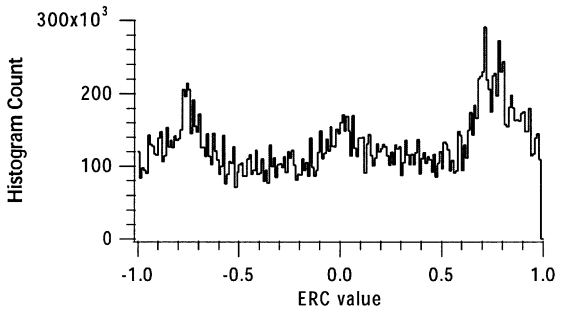


Figure 7. Histogram of ERC values for Unity ($a_R = 1$). Certain ERC values are consistently selected over others. This histogram plots the frequency values for 26.0 million ERC tokens, which were taken from all populations in Unity when a best-of-trial individual was identified.

taken when a best-of-trial individual was identified. The histogram has been discretized in 0.01 intervals and represents 26.0 million ERC values. The histogram clearly shows a pattern in which certain ERC values have been preferred over others. (See [34] for an extended discussion on the idiosyncracies of this distribution. A follow-up discussion of this distribution is also given in [27].

Figure 8 shows pie charts of how three broad classes of GP individuals are distributed between Control (no ERC tokens) and Unity (with ERC values distributed $[-1, 1]$). Only individuals that scored 50-hits (i.e., those that met the stop criterion) were classified. The three classes are *Perfect, Close Approximate*, and *Approximate*. *Perfect* refers to those individuals that simplify to $f(x) = (x + 1)^3$, exactly. *Close Approximate* refers to those individuals that simplify to a third-order polynomial with coefficients that are approximately equal to those in $f(x)$. *Approximate* refers to non-third order (including rational) polynomials that satisfy the hit criterion near each of the 50 fitness cases, but are not necessarily approximate to $f(x)$ anywhere else. A comparison between Control and Unity individuals shows a distinct change in approach. A much larger ratio of *Approximate* individuals occur in Unity than in Control (43–84%). A much smaller ratio of *Perfect* individuals occur in Unity than in Control (0.5–57%). There are no members of *Close Approximate* for the Con-

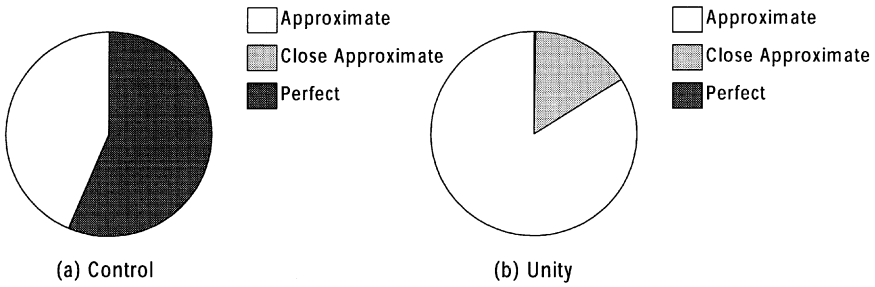


Figure 8. Effect of ERCs on types of solution approaches taken by GP individuals. The manner in which GP solves the binomial-3 problem varies substantially from not using ERCs to using ERCs. These pie charts compare and contrast three broad classes of solutions approaches for 50-hit individuals in Control and Unity: (a) Control; (b) Unity.

trol case, even though such individuals are possible. This shift in classification is consistent with the hypothesis of error correction.

Figure 9 shows an example of individuals taken from Control and Unity, respectively. Both individuals scored 50 hits. Both are typical individuals from the smaller end (i.e., fewer nodes per individual) of their respective groupings. The statistical trends of sifting (i.e., Figure 7) are reflected in the Unity individual. For example, the Unity individual illustrates how one ERC value tends to predominate over all other values in that individual. In this case, the value −0.82256 represents 45%

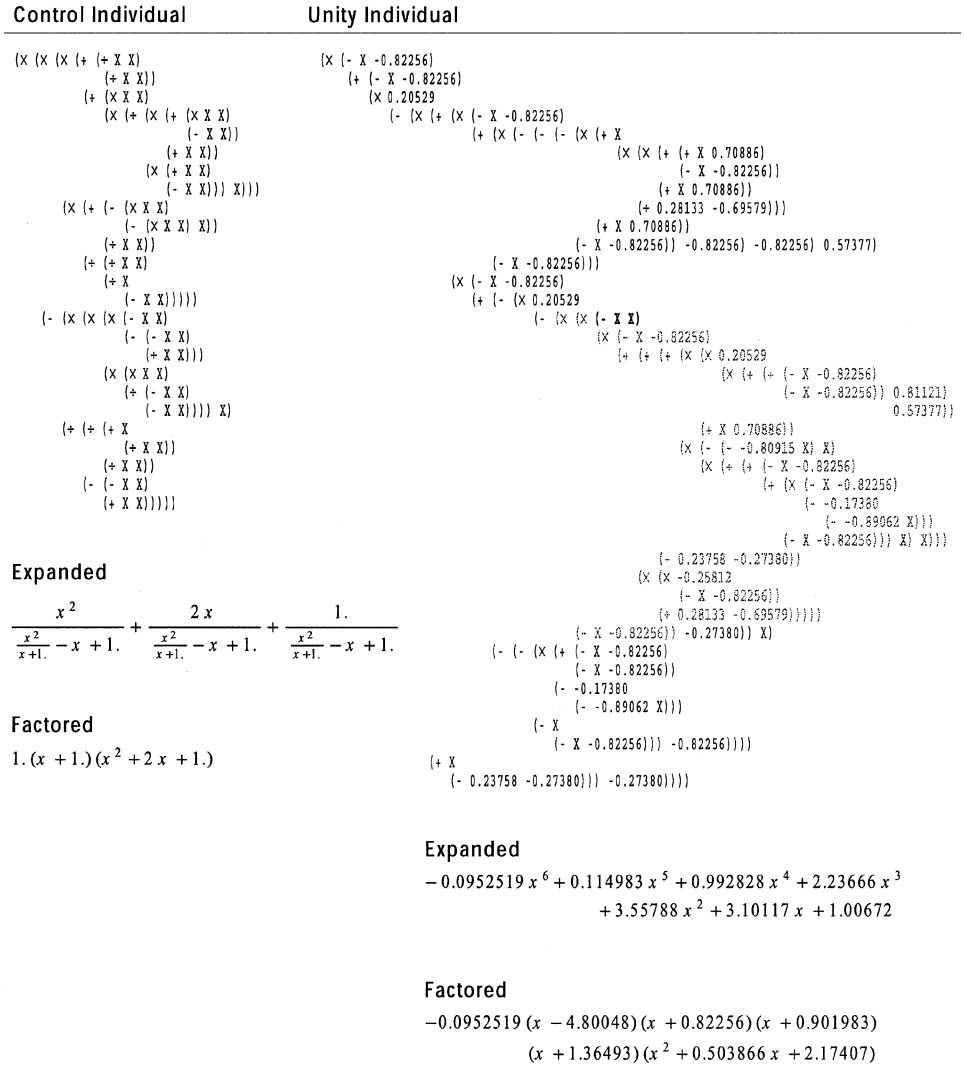


Figure 9. Representative examples from Control (no ERCs) and Unity ($a_R = 1$). Although one would anticipate that individuals from Control and Unity to be different, we note that these differences are reflected in this work's *quantitative* results.

of total number of ERC values in that individual (i.e., 47 ERC values total distributed among 13 unique values). The next highest number of ERC values (i.e., 4) represents only a fifth of that number.

The statistical trends of classes (i.e., Figure 8) are also reflected in the individuals shown in Figure 9. Each individual represents a predominant class: the predominant approach in Control is *Perfect*; for Unity, *Approximate*. Both Control and Unity feature error-correcting mechanisms of incorporation, elimination, and absorption. The degree to which these mechanisms exist, however, do differ. For example, we can examine the degree to which these mechanisms exists by examining $(-X\ X)$, a common 3-tuple structure that can be used in conjunction with multiplication or protected division to create such mechanisms. In Control, this element represents about 35% of all 3-tuple structures—a ratio that happened to be higher than what one would expect by chance alone (i.e., 25%). In Unity, that ratio dipped to 3%, which happened to be lower than one would expect by chance alone (i.e., 6%). In Control, $(-X\ X)$ contributed towards an appropriate solution. In Unity, the only incident of $(-X\ X)$ was used to absorb significant number of ERC values, several of which appeared only in this individual's noc. Note that in Figure 9, the nocs are highlighted while the structure $(-X\ X)$ is bolded.

Taken together, Figures 7–9 provide *quantitative* evidence that suggest meaning is not a purely syntactical consideration, but is partly the result of emergent processes in evolving populations. The findings suggest an additional wrinkle to our response of what makes a problem GP-hard. In particular, part of what makes a problem GP-hard is an inability of GP to find consistent meaning to its tokens—a failed "search for meaning." This finding augments Goldberg and O'Reilly's study of the role contextual semantics in GP dynamics [16]. In their work, contextual semantics were studied with a small set of primitives with distinct and precise meanings. They showed that contextual semantics can dramatically influence size and shape of individuals. In this work, contextual semantics are extended to a fairly large set of primitives (several thousand) with various and ambiguous shades of meaning. We show that contextual semantics can significantly influence the difficulty of a problem. It is this "search for meaning," this "evolution of contextual semantics," that we believe plays a substantial role in determining whether a problem is GP-hard.

We point out that much of our analysis on context-driven inconsistency did not depend on the presence or knowledge of a fitness function. In particular, the behaviors that were described, starting with Figure 6, would have occurred in the presence of *no* selection pressure at all (i.e., no fitness function). Although this does resonate with the notion of biological epistasis, we note that it resonates in the meaning of the term that is quite different from the usage of the term in statistical and quantitative (population) genetics. In particular, the population sense of the term does require the notion of an external environment that provides some form of selection pressure. There is no such pressure that has been presupposed in our analysis in context-driven dependency. Instead, the behavior that is noted in context-driven inconsistency is a kind of epistasis, but in the sense of molecular and biochemical genetics. Furthermore, as in the sense of molecular and biochemical genetics, epistasis is a phenomena that does *not* require a fitness landscape. This distinction is paramount, if only because work in genetic and evolutionary computation has

employed a population view of epistasis, and has subsequently borrowed the mathematics of this view. Our results suggest that perhaps this is not an appropriate borrowing to describe context-driven inconsistency. It is why we believe that rugged landscapes may not always be the most appropriate metaphor for what makes a problem difficult in GP and why Munch's painting might be, instead.[2]

4.4. A note on population size

Conventional wisdom suggests that larger population size can compensate for increasing problem difficulty. Figures 3 and 4 certainly support this wisdom, at least for the Unity case. Average adjusted fitness went up as population increased: 50 individuals, 0.41 average adjusted fitness, 0.24 standard deviation; 500 individuals, 0.76 average adjusted fitness, 0.11 standard deviation; 5000 individuals, 0.86 average adjusted fitness, 0.05 standard deviation. These results were statistically significant (Mann–Whitney U test, individual alpha of 0.001, for all permutations of adjusted fitness for the three populations.) We note, however, that while the average adjusted fitness did increase, the overall trajectory of individuals in size vs. adjusted fitness remained fairly consistent. We further note that the large attractor at the adjusted fitness score of about 0.8 in size vs. adjusted fitness *did not move*. Furthermore, in spite of a much larger population, no attractor appeared at the adjusted fitness score of 1.0, as it did in Control.

The findings suggest that *while increasing population size can improve the overall performance, a ceiling to increased performance can still exist.* We do not claim that such ceilings exists for all problems, or for that matter, all GP configurations using other crossover or mutation variations. However, in the binomial-3 case for our GP configuration, this "ceiling" occurred between the two attractors between 0.8 and 1.0 in adjusted fitness. In other words, increased population meant that more individuals gathered around the 0.8 attractor. It did not result in individuals gathering around the 1.0 attractor.

4.5. Practical implications

Our findings suggest a few practical implications for real-world applications, given the following caveat: our recommendations are based on a single, albeit extensive examination of the binomial-3 problem. Although we believe that the binomial-3 problem is representative of a large class of GP problems that involve data modeling, note that we have not considered other data models in this paper.

For applications of GP in data modeling, it may be beneficial in the long run to:

- Pay attention to the composition and possible interactions between functions and terminal sets. Ambiguities, like large contextual swings in meaning (e.g., the effect of the token with value "1000" as it passes in and out of a noc), can inadvertently contribute to making a problem difficult.

- Cull functions and terminals by trial and error. We have illustrated that there may be performance ceilings that may be intrinsic to a particular specification of a problem. For example, a GP with ERCs might be considered a more general problem solver than a GP without. However, we have shown that even the best GP performance using ERCs is only fair in comparison to GP using no ERCs—a finding that is particular to the binomial-3 problem. In a sense, it could be argued that we "tuned" the problem specification for enhancing GP performance (i.e., by removing ERCs from the function set).
- Consider treating data modeling as a two-stage process for GP. The first stage would be using GP to discover a select subset of terminals and functions from a more general set. The second stage would be using GP to determine an appropriate data model. Our findings for the binomial-3 problem is that as capable as GP is in working with general function and terminal sets, GP seemed to work best with a function and terminal set that is specific to the problem at hand. Furthermore, the existence of a performance ceiling may preclude the efficacy of relying solely on a one-stage process.

5. Conclusions

What makes a problem GP-hard? This paper has considered the metaphor of a fitness landscape in describing problem difficulty and has indicated that this metaphor may not have sufficient explanatory power. In particular, we reinforced the use of the metaphor of biochemical epistasis, and claim that node–node interactions play a significant role in determining problem difficulty that is distinct and separate from that which has been ascribed to a fitness landscape and population epistasis. We have examined one of the formalisms that have results from that metaphor and have shown that formalisms derived under GA do adequately account for phenomena observed in GP.

The particular phenomena that we have examined are results from the binomial-3 problem. The binomial-3 is our, albeit simple, test problem that does not have an antecedent in GA research, but is an instance from a domain that has had an extensive history of use in GP. We have quantitatively demonstrated that this problem is tunable while keeping the combinatorial search space invariant. We have also demonstrated that the tuning characteristics of this problem are well posed and monotonic with respect to the tuning parameter a_{R}.

Our analysis has shown that both content and context matter in determining problem difficulty. We have shown that conflicts in meaning can result when the context of the terminal content is switched. We have made a case that this conflict is an emergent phenomenon and is a result of GP attempting to ascribe consistent worth among subtrees. For that reason, we have suggested that the conflict in trying to ascribe worth is largely internal process, as opposed to an external environmental that is suggested by the metaphor a fitness landscape. The results provide quantitative evidence that supports conjectures in GP theory that both context and content are integral factors to consider.

Our analysis has also indicated that "meaning," specifically the meaning ascribed to individual nodes, can be viewed as an emergent phenomena. We have shown quantitative evidence for error-correcting mechanisms that have appeared, apparently as needed, to address the problem of nodes that turn out to have detrimental meanings. For this reason, we have suggested that syntax and structure only partially determine problem difficulty. Our findings have augmented work in contextual semantics in GP. We have shown that contextual semantics can significantly influence the difficulty of a problem. It is this "search for meaning," this "evolution of contextual semantics," that we believe plays a substantial role in determining whether a problem is GP-hard. It is also why we believe that rugged landscapes may not always be the most appropriate metaphor for what makes a problem difficult in GP and why Munch's painting of an internally tortured individual might be instead.

Finally, we have indicated a few practical implications of our work. In particular, we have indicated that substantial performance gains can occur by relatively minor modifications in function and terminal set specifications. We have also noted that "optimal" solutions derived under one set of problem specifications may not necessarily represent the best-possible solution. It is possible, we suggest, that a problem specification can introduce (inherent) performance ceilings. Larger population sizes may not be enough to compensate.

For more information and related papers on this subject, please see our website at www.sprl.umich.edu/acers.

Acknowledgments

We graciously thank W. Banzhaf for his kind invitation and persistent encouragement. Our work has benefitted extensively from others: D. Ampy, H. Li, and M. Ratanasavetavadhana, for experiment protocols; G. Eickhoff, P. Litvak, and S. Yalcin, for their philosophical analysis; S. Chang, for support software; D. Zongker and W. Punch for lilgp; S. Luke and P. Andersen for their patches to lilgp; M. Matsumoto and T. Nishimura for `mt19937.c`, their C implementation of the Mersenne Twister; S. Ross. J. McClain, and M. Holczer, for their previous unpublished work. We thank U.-M. O'Reilly, C. Jacob and the anonymous reviewers for their constructive comments on an earlier conference draft of paper. This research was partially supported through grants from U-M CoE, UROP-OVPR, and SPRL. We thank J. Vesecky and S. Gregerman for their continued support. The first author thanks I. Kristo and S. Daida.

Notes

1. Even to the extent to which we question, we emphasize that we have not set out to "disprove" the metaphor of landscapes. Indeed, one does not falsify any metaphor in a way that one falsifies a scientific hypothesis (see [28]). Likewise, we would also add that at some level, the metaphor of landscapes may be useful in describing broad classes of difficulty for GP. However, we do question the capability of current EC landscape theory to account for the phenomena noted in Section 3. We also question the efficacy of the landscape metaphor itself in accounting for these phenomena, as well.

2. Our findings are also an empirical complement to Altenberg's theoretical notion of constructional fitness [50]. See also Koza [1, pp. 619–641] for what he calls as the lens effect, which has some bearing on our findings. However, we have not discussed the lens effect in our discussion, if only because Koza's discussion focuses on automatically defined functions.

References

1. J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, The MIT Press: Cambridge, 1992.
2. U.-M. O'Reilly, "Using a distance metric on genetic programs to understand genetic operators." in Proc. 1997 IEEE Int. Conf. Systems, Man, and Cybernet., IEEE Press: Piscataway, 1997, pp. 4092–4097.
3. K. Mathias and L. D. Whitley, "Genetic operators, the fitness landscape and the traveling salesman problem," in Parallel Problem Solving in Nature, R. Männer and B. Manderick (eds.), Elsevier Science Publishers B.V: Amsterdam, 1992, pp. 219–228.
4. J. Horn, and D. E. Goldberg, "Genetic algorithm difficulty and the modality of fitness landscapes," in Foundations of Genetic Algorithms 3, L. D. Whitley and M. D. Vose (eds.), Morgan Kaufmann Publishers: San Francisco, 1995, pp. 243–269.
5. T. C. Jones, "Evolutionary algorithms, fitness landscapes and search," Ph.D. Dissertation, University of New Mexico, Albuquerque, 1995.
6. W. B. Langdon and R. Poli, "Why ants are hard," in Genetic Programming 1998: Proc. Third Ann. Conf., 22–25 July, 1998, University of Wisconsin, Madison, J. R. Koza, W. Banzhaf, K. Chellapilla, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1994, pp. 193–201.
7. D. Jefferson, R. Collins, et al., "Evolution as a theme in artificial life: the genesys/tracker system," in Artificial Life II, C. Langton, C. Taylor, J. Farmer, and S. Rasmussen (eds.), Addison-Wesley: Redwood City, 1991, pp. 549–578.
8. K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems." Ph.D. dissertation, Ann Arbor, The University of Michigan, 1975.
9. D. H. Ackley, A Connectionist Machine for Genetic Hillclimbing, Kluwer Academic Publishing: Boston, 1987.
10. J. R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs, The MIT Press: Cambridge, 1994.
11. W. F. Punch, D. Zongker, et al., "The royal tree problem, a benchmark for single and multiple population genetic programming," in Advances in Genetic Programming, P. J. Angeline and K. E. Kinnear, Jr. (eds.), The MIT Press: Cambridge, 1996, pp. 299–316.
12. M. Mitchell, S. Forrest, et al., "The royal road for genetic algorithms: fitness landscapes and GA performance," in Proc. First Euro. Conf. Artif. Life. Toward a Practice of Autonomous Systems, F. J. Varela and P. Bourgine (eds.), The MIT Press: Cambridge, 1992, pp. 245–254.
13. C. Gathercole and P. Ross, "An adverse interaction between crossover and restricted tree depth in genetic programming," in Genetic Programming 1996: Proc. First Ann. Conf., 28–31 July 1996, Stanford University, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (eds.), The MIT Press: Cambridge, 1996, pp. 291–296.
14. T. Soule, J. A. Foster, and J. Dickinson, "Using genetic programming to approximate maximum cliques," in Genetic Programming 1996: Proceedings of the First Annual Conference, 28–31, July, 1996, Stanford University, J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (eds.), The MIT Press: Cambridge, 1996, pp. 400–405.
15. U.-M. O'Reilly, "The impact of external dependency in genetic programming primitives," in Proc. 1998 IEEE Int. Conf. Evol. Comput., IEEE Press: Piscataway, 1998, pp. 306–311.
16. D. E. Goldberg and U.-M. O'Reilly, "Where does the good stuff go, and why? How contextual semantics influences program structure in simple genetic programming," in Proc. First Euro. Conf. Genetic Program., Paris, France, W. Banzhaf, R. Poli, M. Schoenauer, and T. Fogarty (eds.), Springer Verlag: Berlin, 1998.

17. U.-M. O'Reilly and D. E. Goldberg, "How fitness structure affects subsolution acquisition in genetic programming," in Genetic Programming 1998: Proc. Third Ann. Conf., 22–25 July, 1998, University of Wisconsin, Madison, J. R. Koza, W. Banzhaf, K. Chellapilla, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1998, pp. 269–277.
18. S. Luke and L. Spector, "A revised comparison of crossover and mutation in genetic programming," in Genetic Programming 1998: Proc. Third Ann. Conf., 22–25 July, 1998, University of Wisconsin, Madison, J. R. Koza, W. Banzhaf, K. Chellapilla, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1998, pp. 208–213.
19. N. F. McPhee, N. J. Hooper, et al., "Impact of types on essentially typeless problems in GP," in Genetic Programming 1998: Proc. Third Ann. Conf., 22–25 July 1998, University of Wisconsin, Madison, J. R. Koza, W. Banzhaf, K. Chellapilla, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1998, pp. 232–240.
20. P. J. Angeline, "Subtree crossover: building block engine or macromutation?" in Genetic Programming 1997: Proc. Second Ann. Conf., 13–16 July, 1997, Stanford University, J. R. Koza, K. Deb, M. Dorigo, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1997, pp. 9–17.
21. S. Wright, "The roles of mutation, inbreeding, crossbreeding and selection in evolution," in Proc. Sixth Int. Congr. Genet., 1932, pp. 356–366.
22. G. G. Simpson, Tempo and Mode in Evolution, Columbia University Press: New York, 1944.
23. T. Dobzhansky, Genetics and the Origin of the Species, Columbia University Press: New York, 1941.
24. D. J. Depew and B. H. Weber, Darwinism Evolving: Systems Dynamics and the Genealogy of Natural Selection, The MIT Press: Cambridge, 1995.
25. T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in Proc. Sixth Int. Conf. Genet. Algo., L. J. Eshelman (ed.), Morgan Kaufmann Publishers: San Francisco, 1995, pp. 184–192.
26. B. Naudts and L. Kallel, "A comparison of predictive measures of problem difficulty in evolutionary algorithms," IEEE Tran. Evol. Comput., vol. 4(1), pp. 1–15, 2000.
27. O. Chaudhri, J. M. Daida, et al. "Characterizing a tunably difficult problem in genetic programming," in GECCO-00: Proc. Genet. Evolut. Comput. Conf., 9–12 July, 2000, Las Vegas, Nevada USA, 2000, pp. 395–402.
28. J. M. Daida, S. J. Ross, et al. "Challenges with verification, repeatability, and meaningful comparisons in Genetic Programming." in Genetic Programming 1997: Proc. Second Ann. Conf., 13–16 July, 1997, Stanford University, J. R. Koza, K. Deb, M. Dorigo, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1997, pp. 64–69.
29. M. J. Wade, "Epistasis," in Keywords in Evolutionary Biology, E. F. Keller and E. A. Lloyd (eds.), Harvard University Press: Cambridge, 1996, pp. 87–91.
30. A. S. Fraser, "Simulation of genetic systems by automatic digital computers. I. Introduction," Aust. J. Bio. Sci., vol. 10, 1957, pp. 484–491.
31. R. E. Smith and J. E. Smith, "An examination of tunable, random search landscapes," in Foundations of Genetic Algorithms 5, W. Banzhaf and C. Reeves (eds.), Morgan Kaufmann Publishers: San Francisco, 1999, pp. 165–181.
32. R. B. Heckendorn, S. Rana, and L. D. Whitley, "Test function generators as embedded landscapes," in Foundations of Genetic Algorithms 5, W. Banzhaf and C. Reeves (eds.), Morgan Kaufmann Publishers: San Francisco, 1999, pp. 183–198.
33. A. Adams, Sierra Nevada: The John Muir Trail, Archetype Press: Berkeley, 1938.
34. J. M. Daida, R. B. Bertram, J. A. Polito 2, and S. A. Stanhope, "Analysis of single-node (building) blocks in genetic programming," in Advances in Genetic Programming 3, L. Spector, W. B. Langdon, U.-M. O'Reilly, and P. J. Angeline (eds.), The MIT Press: Cambridge, 1999a, pp. 217–241.
35. P. J. Angeline, "An investigation into the sensitivity of genetic programming to the frequency of leaf selection during subtree crossover," in Genetic Programming 1996: Proc. First Ann. Conf., 28–31 July, 1996, Stanford University, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (eds.), MIT Press: Cambridge, 1996, pp. 21–29.
36. M. Evett and T. Fernandez, "Numeric mutation improves the discovery of numeric constants in genetic programming," in Genetic Programming 1998: Proc. Third Ann. Conf., 22–25 July 1998, University of Wisconsin, Madison, J. R. Koza, W. Banzhaf, K. Chellapilla, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1998, pp. 66–71.

37. G. R. Raidl, "A hybrid GP approach for numerically robust symbolic regression," in Genetic Programming 1998: Proc. Third Annu. Conf., 22–25 July, 1998, University of Wisconsin, Madison, J. R. Koza, W. Banzhaf, K. Chellapilla, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1998, pp. 323–328.
38. D. Zongker and W. Punch, *lilgp*, Lansing, Michigan State University Genetic Algorithms Research and Applications Group, 1995, http://garage.cps.msu.edu/software-index.html.
39. M. Matsumoto and T. Nishimura *mt/19937.c.* Keio, Department of Mathematics, 1997, Keio University. http://www.math.keio.ac.jp/∼matumoto/emt.html.
40. M. Matsumoto and T. Nishimura, "Mersenne Twister: a 623-dimensionally equidistributed uniform pseudorandom number generator," ACM Trans. Model. Comput. Simulation, vol. 8(1), pp. 3–30, 1998.
41. J. M. Daida, S. P. Yalcin, et al., "Of metaphors and darwinism: deconstructing genetic programming's chimera," in Proc. 1999 Cong. Evol. Comput., 6–9 July, 1999, Mayflower Hotel, Washington DC, IEEE Press: Piscataway, 1999c, pp. 453–462.
42. J. M. Daida, D. S. Ampy, et al., "Challenges with verification, repeatability, and meaningful comparison in genetic programming: Gibson's magic," in Proc. Genet. Evol. Comput. Conf., 13–17 July, 1999, Orlando, FL, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1999b, pp. 1851–1858.
43. M. M. Meysenberg, and J. A. Foster, "Randomness and GA performance revisited," in Proc. Genetic Evol. Comput. Conf., 13–17 July 1999, Orlando, Florida, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1999, pp. 425–432.
44. T. Soule and J. A. Foster, "Code size and depth flows in genetic programming," in Genetic Programming 1997: Proc. Second Ann. Conf., 13–16 July, 1997, Stanford University, J. R. Koza, K. Deb, M. Dorigo, et al. (eds.), Morgan Kaufmann Publishers: San Francisco, 1997, pp. 313–320.
45. W. Banzhaf, P. Nordin, et al., Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications, Morgan Kaufmann Publishers, Inc.: San Francisco, 1998.
46. N. F. McPhee and J. D. Miller, "Accurate replication in genetic programming," in Proc. Sixth Int. Conf. Gen. Algo., L. J. Eselman (eds.), Morgan Kaufmann Publishers: San Francisco, 1995, pp. 303–309.
47. L. A. Levin, "Universal sequential search problems," Problems of Information Transmission, vol. 9(3), pp. 265–266, 1973.
48. L. A. Levin, "Randomness conservation inequalities: information and independence in mathematical theories," Inform. Cont., vol. 61, pp. 15–37, 1984.
49. U.-M. O'Reilly and F. Oppacher, "The troubling aspects of a building block hypothesis for genetic programming," in Foundations of Genetic Algorithms 3, L. D. Whitley and M. D. Vose (eds.), Morgan Kaufmann Publishers: San Francisco, 1995, pp. 73–88.
50. L. Altenberg, "The evolution of evolvability in genetic programming. "In Advances in Genetic Programming, K. E. Kinnear, Jr. (ed.), MIT Press: Cambridge, 1994, pp. 47–74.