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

Jason M. Daida*, John A. Polito 2, Steven A. Stanhope*, Robert R. Bertram*, Jonathan C. Khoo*, and Shahbaz A. Chaudhary***

*The University of Michigan, Artificial Intelligence Laboratory and Space Physics Research Laboratory
2455 Hayward Avenue, Ann Arbor, Michigan 48109-2143
**The MEDSTAT Group, 777 E. Eisenhower Parkway, Ann Arbor, Michigan 48108

Abstract

This paper addresses the issue of what makes a problem 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 show that for at least this problem, the metaphor is misleading.

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 genetic programming has come from work and ideas in areas like genetic algorithms (GA). For example, one could take a cue from previous work in genetic algorithms 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 genetic programming 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 1992], Chapter 8. In that 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 [1997] attempted to extend fitness landscape analysis in GA research (i.e., [Mathias and Whitley 1992; Horn and Goldberg 1995; Jones 1995]) to GP. Langdon and Poli [1998] 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 [Jefferson, et al. 1991; Koza 1992]).

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 [1975] or Ackley [1987] test suites in GA research). There have been several promising candidates, however. Koza [1992] 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 [1994] 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 Bumblebee problem. Punch et al. [1996] introduced a tunably difficult Royal Tree problem, which they have designed along the lines of the Royal Road problem [Mitchell, et al. 1992] in GA. Gathercole and Ross have proposed the MAX test suite [Gathercole and Ross 1996], which they have developed along the lines of the Ones-Max problem [Ackley 1987] in GA research. Foster and his colleagues have offered the Maximum Clique problem for GP [Soule, et al. 1996].

Other researchers have turned to examples from Koza's books [Koza 1992; Koza 1994] in place of a recognized suite of test problems. Typical suites have included Multiplexer, Lawnmower, symbolic regression, and Artificial Ant (in [Luke and Spector 1998]); Boolean parity, symbolic regression, Artificial Ant (in [McPhee, et al. 1998]); Boolean Parity, Sunspot, and Intertwined Spiral (in [Angeline 1997]). 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 easily to relatively difficult. We show that under certain conditions, the problem scales logarithmically in difficulty. 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 1932], 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 [Simpson 1944, p.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, California, USA) [Dobzhansky 1941; Depew and Weber 1995, p. 294].

In his thesis, Jones correctly noted that this common idea of adaptive landscapes is fraught with pitfalls for EC [Jones 1995, p. 46]. Instead, Jones proposed a one-operator/one-landscape view of fitness for genetic algorithms. 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 genetic algorithm's operators. By framing the fitness landscape as such, Jones and Forrest [1995] were able to devise a metric of problem difficulty that was largely independent of a genetic algorithm.

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 sole driver for 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 and context. 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" [Adams 1938] — 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 would be Edvard Munch's painting "The Scream"—an oil depicting an internally tortured soul on what would otherwise be a fairly mundane 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 {+, –, ×, ÷}, which correspond to 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 {+, –, ×, ÷}, which we presume for this paper. Other sets may include other permutations such as {+, ×} or { –, ×}.

A terminal set is a subset of {X, $\mathcal{R}$}, where X is the symbolic variable and $\mathcal{R}$ is the set of ephemeral random constants (ERCs). We presume that the ephemeral random constants are uniformly distributed over a specified interval of the form $[-a_{\mathcal{R}}, a_{\mathcal{R}}]$, where $a_{\mathcal{R}}$ 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, $\mathcal{R}$} (a binomial-3 problem with ERCs).

Tuning is achieved by varying the value associated with $a_{\mathcal{R}}$. We defer until Section 4 the discussion of how $a_{\mathcal{R}}$ 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 with other problems in GP. It requires symbol manipulation. It allows for introns (i.e., 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 introns, some of which involve the structure `( - X X)`. Multiplication of this structure to any other results in a value of 0. Division by this structure to any other results in a value of1. 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 [Daida, et al. 1999]).

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 [Daida, et al. 1999].

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 [Koza 1992], 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, bio-engineering, 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., [Angeline 1996; Evett and Fernandez 1998; Raidl 1998]).

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 [Daida, et al. 1999].

2.3 EXPERIMENT PROCEDURE

We used a patched version of lilgp [Zongker and Punch 1995] 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). We did replace the random number generator (RNG) in lilgp (Knuth subtractive RNG) with the Mersenne Twister [Matsumoto and Nishimura 1997; Matsumoto and Nishimura 1998]. The Mersenne Twister has excellent mathematical properties that make this RNG a reasonable candidate for theoretical work in GP. (See [Daida, 1997] for issues concerning RNGs). 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 were unique, with every ERC value having just one instance in an initial population. We configured lilgp to run as a single thread.

Most of the GP parameters were identical to those mentioned in Chapter 7 [Koza 1992]: 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 [Koza 1992], which specifies a maximum number of generations = 51 and a maximum depth = 17. Part of the reason we extended these parameters was to mitigate against possible effects that occur when GP processes individuals at these limits.)

The experiment involved varying the tuning parameter $a_{\Re}$. We used seven values of $a_{\Re}$: 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 4800 runs for the first part. 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 use 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 5 out of 6 trials resulting in a perfect score. We note that for the most part, the general trend is that if $a_{\Re} \geq 1$ and $a_{\Re}$ increasing, the problem becomes increasingly more difficult to solve. (That trend does not hold for $0 \leq a_{\Re} < 1$). Figure 1 plots the results for $a_{\Re} \geq 1$ in Table 1, with hit scores normalized to 100%. The regression coefficient is -0.997.

Table 1. Main Part Summary. This table shows the total number of trials (out of 600 trials) that scored perfectly, in the upper decile, and in the upper quartile.

$a_{\Re}$	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
1000	3	4	5

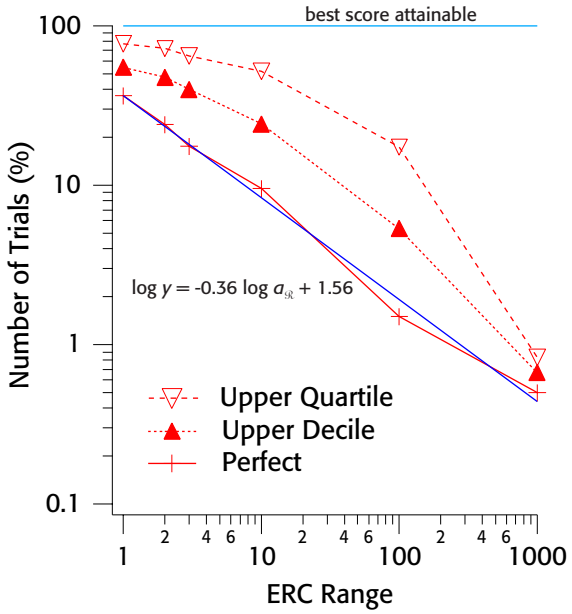


Figure 1 Hit Score v. a_{SR}. This log-log plot shows the relationship between the tuning parameter a_{SR} and the hit score. The problem becomes progressively more difficult as a_{SR} increases.

Figure 2 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 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 first column 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.

Figure 2 second column 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 0 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.

Figure 2 third column 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.

4. DISCUSSION

That problem difficulty can be tuned by means of varying a_{SR} has been demonstrated clearly by the experiment. As shown in Table 1 and Figure 1, the hit scores for perfect, upper decile, and upper quartile were monotonically decreasing for increasing a_{SR} for $a_{SR} \geq 1$. The hit scores for perfect are well described with a log-log regression fit.

The crux of this paper addresses why the binomial-3 problem is tunable in this way. A reasonable notion associated with increasing a_{SR} 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_{SR} 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_{SR}, what is changed is not the number of tokens, but the table of lookup values that are assigned to those tokens. Figure 3 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_{SR}. (Note: in actuality, two different random number seeds were chosen for each ERC range to generate two independent samples). The combinatorial search space remains statistically invariant even though a_{SR} varies. (For example, the number of ERCs allocated for a population of 500 individuals, regardless of a_{SR}, was roughly 4500±400.)

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

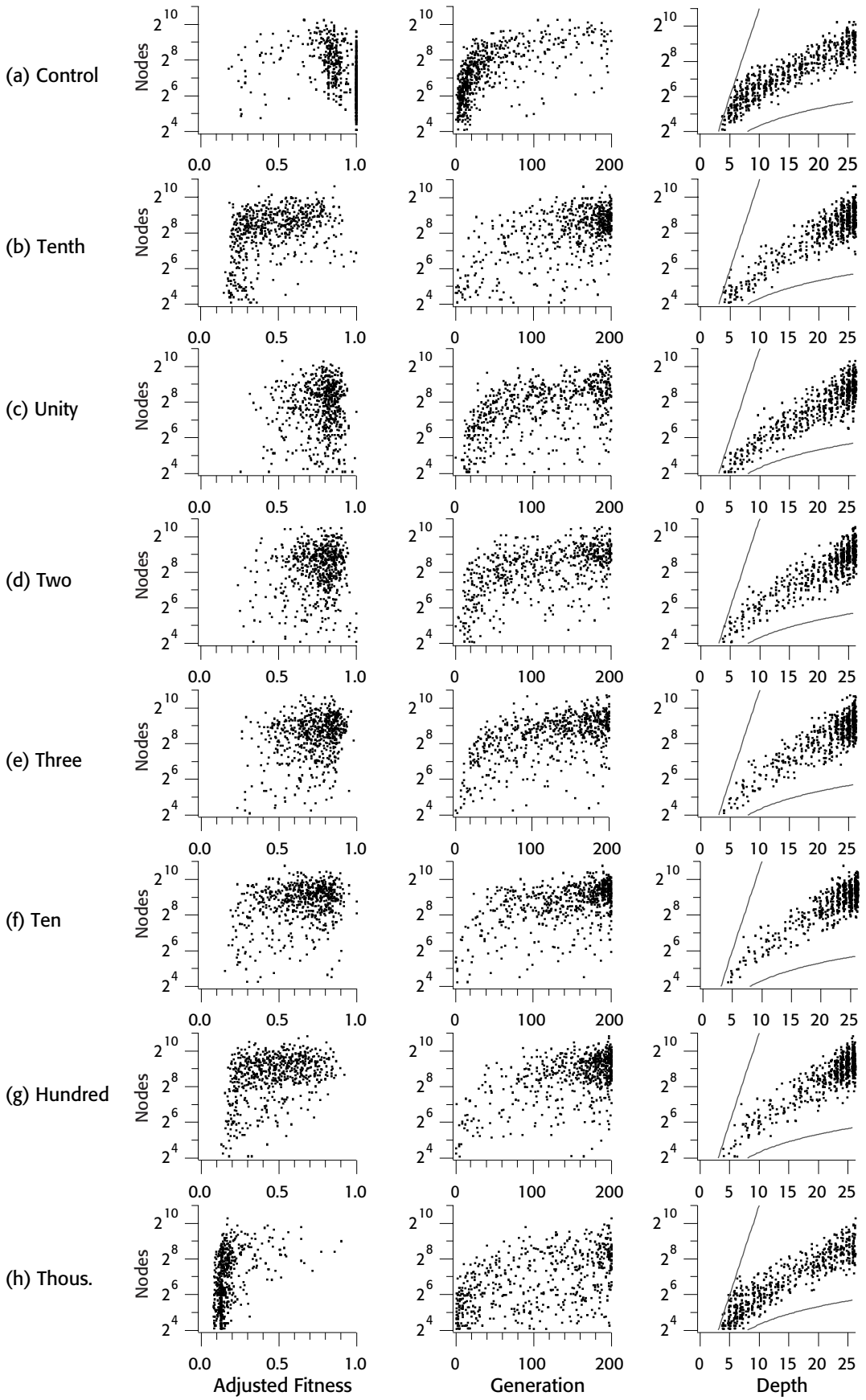
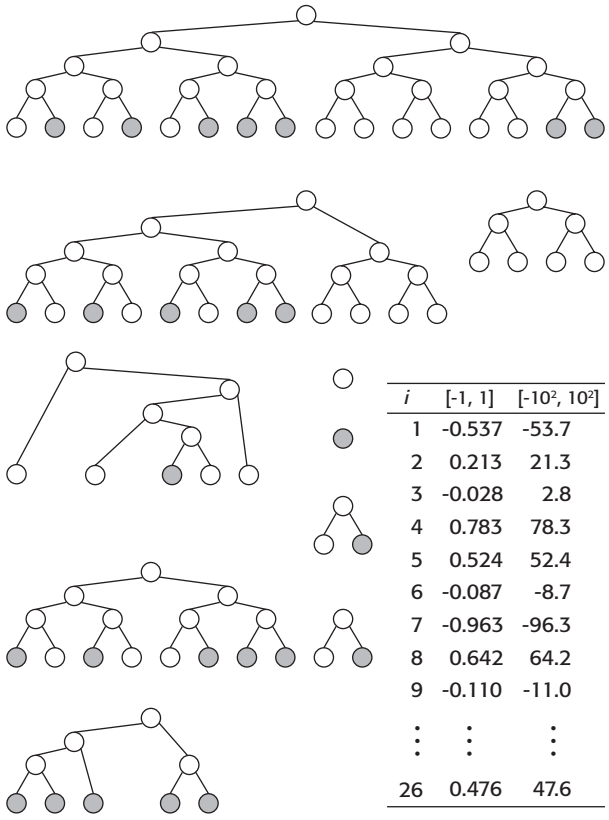


Figure 2. Best-of-Trial Results. Each row summarizes a data set, where each data set consisted of 600 trials. This figure shows the effect of increasing ERC values on the size and shape of best-of-trial individuals.

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 (e.g., [Gilbert, et al. 1998]).

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_{sR} 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 1000. Clearly, the "obviously wrong" values would be selected against. By positing such a hypothesis, 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.*



i	[-1, 1]	[-10², 10²]
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 3. Hypothetical Population with Ephemeral Random Constants (ERCs). ERCs are denoted in gray. ERC values are show in the accompanying table. Although the values change, the number of ERCs do not.

After all, the binomial-3 problem became *harder* as a_{sR} 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 4 illustrates two common inconsistencies that can arise.

Figure 4a shows the inconsistencies that arise when the context of an ERC value switches from an intron 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 $\mathtt{r}_1$ 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 $(x + 999.9)$.

Introns probably represent the most dramatic way an ERC value can result in inconsistencies. However, there are other ways that produce such conflicts. Figure 4b shows the another possibility. As in the hypothetical example depicted in Figure 4a, 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 4a and 4b represent just two of several means 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 a single token. We have shown in [Daida, et al. 1999] 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_{sR} varies the destructive effect of crossover.

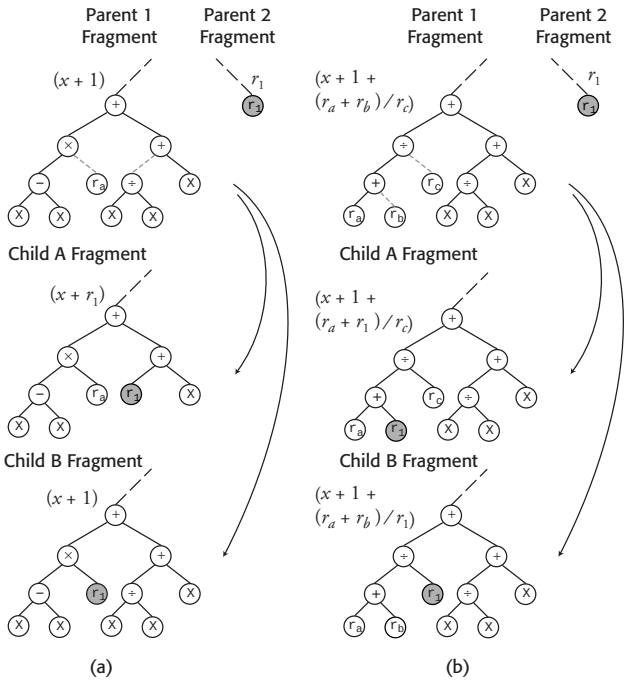


Figure 4. Context of Content Matters in Determining Meaning. The context of ERC determines itsfunctional meaning. (a) For introns. (b) For division.

Evidence for the phenomena that we have described can be seen in Figure 2. 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. [Soule and Foster 1997; Banzhaf, et al. 1998]. The trends in program size and shape shown in Figure 2 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 2 column 2). Likewise, as the difficulty of the problem increased, the programs were deeper (Figure 2 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. [McPhee and Miller 1995; Soule and Foster 1997; Banzhaf, et al. 1998]. In Figure 2 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 note 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 shown in Figure 2, Tenth. Figure 4b represents the case where context switching between numerator and denominator can be significant, particularly for values of $|r| << 1$.

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. We have examined one of the formalisms that have results from that metaphor and show that formalisms derived under GA do not 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 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 $a_{\Re}$.

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 a largely internal process, as opposed to an external environmental that is suggested by the metaphor of a fitness landscape.

Our results support conjectures in GP theory that both context and content of subtrees are integral factors to consider. (See [O'Reilly and Oppacher 1995; Daida, et al. 1999]).

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

Acknowledgments

Our work has benefitted extensively from others in our research group: D. Ampy, O. Chaudri, H. Li, and M. Ratanasavetavadhana, for experiment protocols; G. Eickhoff, P. Litvak, and S. Yalcin, for their philosophical analysis; S. Chang, for support software; S. Ross, J.McClain, and M. Holczer, for their previous unpublished work. We thank U.-M. O'Reilly, C. Jacob and the other (anonymous) reviewers for their constructive comments. 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.

References

Ackley, D. H. (1987). *A Connectionist Machine for Genetic Hillclimbing*. Boston: Kluwer Academic Publishing.

Adams, A. (1938). *Sierra Nevada: The John Muir Trail*. Berkeley: Archetype Press.

Angeline, P. J. (1996). "An Investigation into the Sensitivity of Genetic Programming to the Frequency of Leaf Selection During Subtree Crossover." In J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference: July 28–31, 1996, Stanford University*. Cambridge: The MIT Press. pp. 21–29.

Angeline, P. J. (1997). "Subtree Crossover: Building Block Engine or Macromutation?" In J. R. Koza, K. Deb, M. Dorigo, et al (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13-16, 1997, Stanford University*. San Francisco: Morgan Kaufmann Publishers. pp. 9–17.

Banzhaf, W., P. Nordin, et al. (1998). *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. San Francisco: Morgan Kaufmann Publishers, Inc.

Daida, J. M., R. B. Bertram, J. A. Polito 2, and S. A. Stanhope. (1999). "Analysis of Single-Node (Building) Blocks in Genetic Programming." In L. Spector, W. B. Langdon, U.-M. O'Reilly and P. J. Angeline (Eds.), *Advances in Genetic Programming 3*. Cambridge: The MIT Press (In press).

Daida, J. M., S. J. Ross, et al. (1997). "Challenges with Verification, Repeatability, and Meaningful Comparisons in Genetic Programming." In J. R. Koza, K. Deb, M. Dorigo, et al. (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13-16, 1997, Stanford University*. San Francisco: Morgan Kaufmann Publishers. pp. 64–69.

De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. dissertation. Ann Arbor, The University of Michigan.

Depew, D. J. and B. H. Weber (1995). *Darwinism Evolving: Systems Dynamics and the Genealogy of Natural Selection*. Cambridge: The MIT Press.

Dobzhansky, T. (1941). *Genetics and the Origin of the Species*. New York: Columbia University Press.

Evett, M., and T. Fernandez (1998). "Numeric Mutation Improves the Discovery of Numeric Constants in Genetic Programming." In J. R. Koza, W. Banzhaf, K. Chellapilla, et al (Eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference, July 22–25, 1998, University of Wisconsin, Madison*. San Francisco: Morgan Kaufmann Publishers. pp. 66–71.

Gathercole, C. and P. Ross (1996). "An Adverse Interaction between Crossover and Restricted Tree Depth in Genetic Programming." In J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference: July 28–31, 1996, Stanford University*. Cambridge: The MIT Press. pp. 291–296.

Horn, J. and D. E. Goldberg (1995). "Genetic Algorithm Difficulty and the Modality of Fitness Landscapes." In L. D. Whitley and M. D. Vose (Eds.), *Foundations of Genetic Algorithms 3*. pp. 243–269.

Jefferson, D., R. Collins, et al. (1991). "Evolution as a Theme in Artificial Life: The Genesys/Tracker System." In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen (Eds.), *Artificial Life II*. Redwood City: Addison-Wesley. pp.549–578.

Jones, T. and S. Forrest (1995). "Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms." In L. J. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco: Morgan Kaufmann Publishers, Inc. pp. 184–192.

Jones, T. C. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search*. Ph.D. Dissertation. Albuquerque: University of New Mexico.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: The MIT Press.

Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge: The MIT Press.

Langdon, W. B. and R. Poli (1998). "Why Ants Are Hard." In J. R. Koza, W. Banzhaf, K. Chellapilla, et al (Eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference, July 22–25, 1998, University of Wisconsin, Madison*. San Francisco: Morgan Kaufmann Publishers. pp. 193–201.

Luke, S. and L. Spector (1998). "A Revised Comparison of Crossover and Mutation in Genetic Programming." In J. R. Koza, W. Banzhaf, K. Chellapilla, et al (Eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference, July 22–25, 1998, University of Wisconsin, Madison*. San Francisco: Morgan Kaufmann Publishers. pp. 208–213.

Mathias, K. and L. D. Whitley (1992). "Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem." In R. Männer and B. Manderick (Eds.), *Parallel Problem Solving in Nature*. Amsterdam: Elsevier Science Publishers B. V. pp. 219–228.

Matsumoto, M. and T. Nishimura (1997). *mt19937.c*. Keio, Department of Mathematics, Keio University. http://www.math.keio.ac.jp/~matumoto/emt.html.

Matsumoto, M. and T. Nishimura (1998). "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator." *ACM Transactions on Modeling and Computer Simulation* 8(1): 3–30.

McPhee, N. F., N. J. Hopper, et al. (1998). "Impact of Types on Essentially Typeless Problems in GP." In J. R. Koza, W. Banzhaf, K. Chellapilla, et al (Eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference, July 22–25, 1998, University of Wisconsin, Madison*. San Francisco: Morgan Kaufmann Publishers. pp. 232–240.

McPhee, N. F. and J. D. Miller (1995). "Accurate Replication in Genetic Programming." In L. J. Eshelman (Eds.), *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco: Morgan Kaufmann Publishers, Inc. pp. 303–309.

Mitchell, M., S. Forrest, et al. (1992). "The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance." In F. J. Varela and P. Bourgine (Eds.), *Proceedings of the First European Conference on Artificial Life. Toward a Practice of Autonomous Systems*. Cambridge: The MIT Press. pp. 245–254.

O'Reilly, U.-M. (1998). "The Impact of External Dependency in Genetic Programming Primitives." In *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics*. Piscataway: IEEE Press. pp. 306–311.

O'Reilly, U.-M. (1997). "Using a Distance Metric on Genetic Programs to Understand Genetic Operators." In *Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics*. Piscataway: IEEE Press. pp. 4092–4097.

O'Reilly, U.-M. and D.E. Goldberg (1998). "How Fitness Structure Affects Subsolution Acquisition in Genetic Programming." In J. R. Koza, W. Banzhaf, K. Chellapilla, et al (Eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference, July 22–25, 1998, University of Wisconsin, Madison*. San Francisco: Morgan Kaufmann Publishers. pp. 269–277.

O'Reilly, U.-M. and F. Oppacher (1995). "The Troubling Aspects of a Building Block Hypothesis for Genetic Programming." In L. D. Whitley and M. D. Vose (Eds.), *Foundations of Genetic Algorithms 3*. San Francisco: Morgan Kaufmann Publishers. pp. 73–88.

Punch, W. F., D. Zongker, et al. (1996). "The Royal Tree Problem, a Benchmark for Single and Multiple Population Genetic Programming." In P. J. Angeline and K. E. Kinnear, Jr. (Eds.), *Advances in Genetic Programming*. Cambridge: The MIT Press. pp. 299–316.

Raidl, G. R. (1998). "A Hybrid GP Approach for Numerically Robust Symbolic Regression." In J. R. Koza, W. Banzhaf, K. Chellapilla, et al (Eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference, July 22–25, 1998, University of Wisconsin, Madison*. San Francisco: Morgan Kaufmann Publishers. pp. 323–328.

Simpson, G. G. (1944). *Tempo and Mode in Evolution*. New York: Columbia University Press.

Soule, T. and J. A. Foster (1997). "Code Size and Depth Flows in Genetic Programming." In J. R. Koza, K. Deb, M. Dorigo, et al (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference, July 13-16, 1997, Stanford University*. San Francisco: Morgan Kaufmann Publishers. pp. 313–320.

Soule, T., J. A. Foster, and J. Dickinson (1996). "Using Genetic Programming to Approximate Maximum Cliques." In J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference: July 28–31, 1996, Stanford University*. Cambridge: The MIT Press. pp. 400–405.

Wright, S. "The Roles of Mutation, Inbreeding, Crossbreeding and Selection in Evolution." In *Proc of the Sixth International Congress of Genetics.* pp. 356–366.

Zongker, D. and W. Punch (1995). *lilgp*. Lansing, Michigan State University Genetic Algorithms Research and Applications Group. http://garage.cps.msu.edu/software/software-index.html.