Is the Perfect the Enemy of the Good?

Sean Luke
George Mason University
http://www.cs.gmu.edu/∼sean/

Liviu Panait
George Mason University
http://www.cs.gmu.edu/∼lpanait/

Abstract

Much of the genetic programming literature compares techniques using counts of ideal solutions found. These counts in turn form common comparison measures such as Koza's Computational Effort or Cumulative Probability of Success. The use of these measures continues despite past warnings that they are not statistically valid. In this paper we too criticize the measures for serious statistical problems, and also argue that their motivational justification is faulty. We then present evidence suggesting that ideal-solution counts are not necessarily positively related to best-fitness-of-run statistics: in fact they are often inversely correlated. Thus claims based on ideal-solution counts can mislead readers into thinking techniques will provide superior final results, when in fact the opposite is true.

1 INTRODUCTION

The best is the enemy of the good.
— Voltaire (1694–1778)

He who is determined not to be satisfied with anything short of perfection will never do anything to please himself or others.
— William Hazlitt (1778–1830)

The research methodology in the genetic programming (GP) has many unusual features. Some of these features are good. Some are not. But we tend to stick with the bad ones out of inertia: we do it that way because others did. Surprisingly, the literature does not have a large number of critics of the existing methodology. One notable exception is Jason Daida, who has criticized poor random number generator usage [1997], evaluation and verification methodology [1999a], and historical metaphors [1999b]. Paterson and Livesey [2000] have decried the poor statistics behind many claims, noting that many papers do no means testing at all. Angeline [1996] has criticized the statistical reliability of Koza's Cumulative Probability of Success measure, a criticism echoed in [Paterson and Livesey 2000].

Here we will continue the criticism of the popular Cumulative Probability of Success and other measures based on counting the number of ideal solutions discovered. There are serious statistical flaws with such measures, but that is not all. These measures also have questionable motivational philosophy, and most importantly, they are poorly correlated with other more accepted measures of run quality in the evolutionary computation community.

This paper was born out of experiments for another purpose: to test whether fitness might be improved and tree size reduced by increasing the noise of a GP breeding operator. The operator chosen was subtree crossover, and it was made noisier through increasing the number of times two parents were crossed over to create a child. Crossing over more times does in fact decrease the mean tree size by statistically significant amounts, but it also worsens the best fitness of the run by a statistically significant margin. But these experiments yielded another odd fact: ideal solution counts were not necessarily tied to fitness results, and in some cases were inversely correlated with them.

The remainder of the paper will discuss common ideal-solution count measures and their statistical weaknesses, and question the motivational philosophy behind them. Then the paper will present the evidence stemming from these experiments. The paper then finishes with discussion and recommendations.

2 A TALE OF TWO MEASURES

The non-GP evolutionary computation literature has traditionally compared techniques using the mean best fitnesses of a large (>30) sample of runs per technique, accompanied with so-called "best-so-far-curves" (plots of the mean best fitness discovered so far), plus a t-test[1] or other difference-of-means test. When generalizability is important, the mean best fitness results are supplemented with generalization ratings on a test set.

Koza [1992] presented a very different metric for computing the "quality" of an evolutionary procedure: how often and how rapidly it discovered the ideal solution. From this data were derived a variety of statistical metrics ultimately computing how many individuals would need to be evaluated before an ideal solution was expected to be found with some probability.

Koza defined four dependent statistical measures, given as follows. The *instantaneous probability of success* measure $Y(m,i)$ is the probability that a run with a population size of m will discover an ideal solution for the first time on generation i. From this it is simple to determine $P(m,i)$, the *cumulative probability of success*, which is the probability that a run will discover an ideal solution before or on generation i. Koza writes $R(m,i,z)$ as the number of evolutionary runs required to have a probability $z = 99\%$ of discovering a solution before generation i. He defines this as

$$R(m,i,z) = \left\lceil \frac{\log(1-z)}{\log(1-P(m,i))} \right\rceil$$

Sometimes this is shortened to just $R(z)$. The *individuals to be processed* measure $I(m,i,z)$ is then defined straightforwardly as $I(m,i,z) = m(i+1)R(m,i,z)$, at least for generational evolutionary procedures. The *computational effort* measure E is the minimum of $I(m,i,z)$ over all values of i.

Ideal-solution count measures have since taken root in the GP community. Of these four, the two most common measures in the literature are Cumulative Probability of Success, for which higher values are better, and its derived measure Computational Effort, for which lower values are better. We performed an informal survey of the genetic programming and evolvable hardware non-poster papers in the three GECCO conferences so far (1999, 2000, 2001). Of these, 44 compared two or more techniques for solution quality. Eighteen compared the mean best fitness of run between techniques.[2] Twelve instead used statistics based on the number of ideal solutions discovered (most used the Koza measures). Fourteen used a train/test methodology borrowed from the machine learning community: typically they measured how long it took to discover a perfect solution to a training set, then tested generalization ability.[3]

We feel there are three problems with ideal-solution count measures. First, they rely on unacceptably poor statistics, at least as generally practiced in the community, and to remedy this would require a very large sample size. Second, they are founded on, in our opinion, an unclear motivational philosophy. The third problem, and most troubling, is that they appear to be uncorrelated with best-fitness-of-run comparisons. These problems call a fair chunk of the literature into question.

2.1 STATISTICAL PROBLEMS

Ideal solution count measures are statistically suspect.

First, the measures are based on a single point sample of the number of ideal solutions and the generations in which they were found. A point sample does not have a mean test: there is no accepted procedure to state that two such samples differ in a statistically significant way.

The point sample difficulty might be alleviated by doing a large number of runs, then dividing them into at least thirty piles, then counting the ideal solutions in each pile and using the mean number of ideal solutions per pile as a sample statistic. For Symbolic Regression this is feasible, as a large number of runs end in perfect solutions. But for many problem domains, ideal solutions are typically few and far between. In this paper we gathered samples of five hundred each, typically five to ten times the number found in most of the experimental literature. And still, the ideal-solution counts for the Artificial Ant and 11-Bit Boolean Multiplexer were so small (at most sixteen and twelve respectively) that dividing into piles was not doable.

Second, several of the measures are statistically dependent across generations. For example, to truly compute statistically independent Cumulative Probability of Success measures for both generation 4 and for generation 8 would require two separate, independent samples.

Third, changes in the Individuals to be Processed measure and its derived Computational Effort measure are both greatly exaggerated when small changes occur in ideal so-

[1] The validity of the t-test and ANOVA for GP were examined in [Paterson and Livesey 2000]. As the t-test and ANOVA assume normality, the authors had expected to find them wanting in the GP realm, but astonishingly they significantly outperformed non-parametric tests. Even so, the authors warned about the dangers of relying too much on t-tests for the skewed distributions common to GP. We agree! Still, we think that t-tests and ANOVAs should at least be the *bare minimum* for means testing in GP.

[2] Of the eighteen experiments which used fitness curves to compare techniques, only seven used good statistics. The other eleven had sample sizes that were too small (much less than 30) or unreported, or they did not indicate statistical significance results or variance information. One of us (Sean Luke) hastens to note that he too has published papers with statistical difficulties, though in his defense they were either corrected ([Luke and Spector 1997] fixed in [Luke and Spector 1998]) or openly acknowledged and justified in the paper itself [Luke 2001b].

[3] As many machine learning algorithms are deterministic, they do not require statistical mean tests. This is not true for stochastic evolutionary computation algorithms: yet relatively few train/test methodology papers in this survey presented statistical significance results.

lution counts. For example, note that the Computational Effort in Figure 12 is a strongly nonlinear function of the number of ideal solutions found, shown in Figure 10.

To overcome all the statistical problems detailed here would require multiple independent samples across generations, plus dramatically larger sample sizes for the more difficult domains. We feel these statistical flaws would be forgivable only if ideal-solution counts were the *only feasible* way to compare techniques. But they are not.

2.2 MOTIVATIONAL PHILOSOPHY

The evaluation functions used in the assessment of a GP individual's fitness do not correspond well to a stated goal of an ideal-solution end result. GP optimizes for as good a fitness as it can get, not for increasing probability of attaining the ideal. Many GP problem domains are highly deceptive, leading the evolutionary trajectory away from the ideal rather than toward it. Consider the Symbolic Regression domain: if the ideal solution is not found early on, and the population has fixated on certain near-solutions, it will continue to tack on code to the bottom of trees which can make the solutions only incrementally fitter. Many Symbolic Regression runs will ultimately generate very large trees with solutions very close to the answer, but far (in makeup) from anything remotely resembling *the* answer.

Given this, and given the statistical problems behind ideal-solution count measures, what is the GP community's motivational justification for using ideal-solution counts at all? We believe that counts are popular because of a philosophical conceit that GP operates over problem domains which demand *correct programs*. More generally this can be thought of as an absolute hard constraint on the desired outcome: either the program works or it doesn't work, and a highly fit but suboptimal solution is not valuable. A term peculiar to GP, "discovery", reinforces this notion: either GP "discovers" the answer, or it doesn't.[4]

[4] In fact, though his influential text introduced the ideal-solution count measures discussed in this paper, Koza couched his support of this philosophy, writing:

"Several pages ago, when I spoke of writing a computer program to center the cart in optimal time, you probably assumed that I was talking about writing a *correct* computer program to solve the problem. Nothing could be further from the truth. In fact, this book focuses almost entirely on *incorrect* programs. In particular, I want to develop the notion that there are gradations in performance among computer programs. Some incorrect programs are very poor; some are better than others; some are approximately correct; occasionally, one may be 100% correct. Expressing this biologically, one could say that some computer programs are fitter than others in their environment. It is rare for any biological organism to be optimal." [Koza 1992, p. 130]

Regression	3	2	4	1	5	6	7	10	8	9
Multiplexer	2	1	3	4	5	6	7	8	9	10
Ant	2	1	4	6	3	5	7	8	9	10

Table 1: Statistical significance groupings for the mean best-fitness-of-run for each problem domain, using the Tukey post-hoc ANOVA test. Numbers indicate the number of crossovers for a given multi-crossover technique. Techniques are ordered by increasing (poorer) mean best-fitness-of-run. Bars connect techniques with statistically insignificant differences in means among them. Note that more crossovers generally results in worse mean best-fitness-of-run. Compare to Figures 1, 5, and 9.

It is clear that there exist valid and important GP problem domains where discovery is of paramount importance. However, we believe that many, and likely most, new problems in GP rarely require 100% correctness as a necessary attribute. These problems include neural networks, molecular structures, soccer softbot programs, probabilistic and quantum algorithms, analog electrical circuits, etc. The primary reason for this, we think, is that these new domains are significantly harder and their optima are often unknown. Discovering the optimum, and particularly discovering it enough times to make statistical conclusions, is a luxury usually reserved for only the simplest of problem domains.

We are now out of the proof-of-concept period for GP. For the technique to now be used realistically as a *tool*, we must assume it will typically be used to attack hard problems for which we do not know the optimum, do not expect it to discover the optimum, nor even know if there *is* an optimum. An engineer would ask: if we already know the answer, why bother to use GP to find it? What matters is not if technique A finds more perfect solutions than technique B does to Easy Problem C. What usually matters is that technique A gets a better answer than B does for Hard Problem D. We submit that if one can "discover" the optimum enough times to validly measure the performance of a technique against a given problem domain, then we are dealing with a toy problem.

If past literature used a weak methodology, we should discontinue its use. Nonetheless, we recognize that for those problems which demand perfection, ideal-solution-count statistics may have some usefulness. Later in this paper we will recommend experimental protocols which may incorporate ideal solution counts as one part of a comprehensive analysis. At the same time, we will propose an alternative which we think provides more useful information.

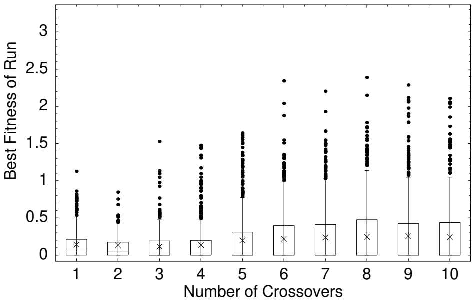

Figure 1: Boxplots of the distribution of best-fitness-of-run, by number of crossovers, Symbolic Regression domain. Lower fitness is better. Compare to Table 1.

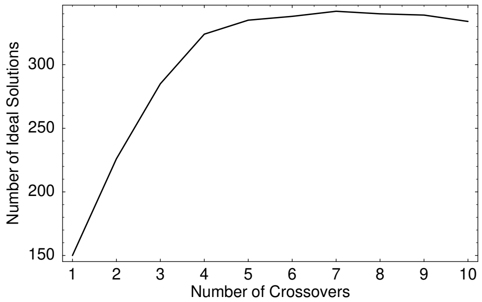

Figure 2: Number of ideal solutions found, by number of crossovers, Symbolic Regression domain.

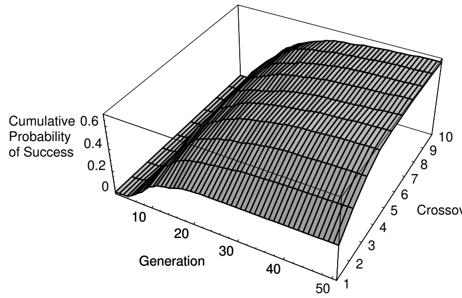

Figure 3: Cumulative Probability of Success per generation, by number of crossovers, Symbolic Regression domain.

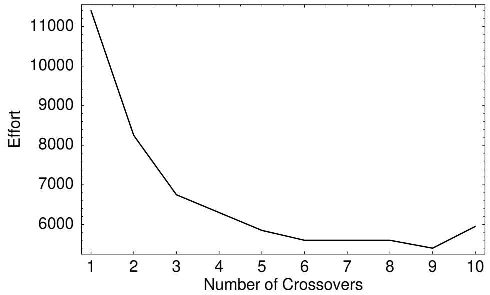

Figure 4: Computational Effort by number of crossovers, Symbolic Regression domain.

3 A TROUBLING LACK OF CORRELATION

At first it seems intuitive that a system which searches better for high-fitness solutions would also tend to find more ideal solutions. An experiment gone awry shows us that in fact this is not necessarily the case.

The experiment we performed applied what we call *multi-crossover* to GP. Ordinarily GP crossover selects two individuals, and performs one swap of randomly-chosen subtrees, producing two children. If a child passes validity constraints (such as maximal depth), it then enters the next generation; otherwise a copy of the child's mother enters in its stead. Multi-crossover is simply a composition of GP crossover operators: two parents are selected and crossover is performed, including validity constraints. Then the children become the "parents" for the next crossover operator, which produces two new children. This happens *N* times, then the final results enter the next generation.

We had two reasons for doing multi-crossover experiments. First, some models of code bloat (our own depth-based theory [Luke 2000], Defense Against Crossover [Banzhaf et al. 1998], and Removal Bias [Langdon et al. 1999]) assume that there is a single crossover per individual; hence the probabilities of choosing crossover point A or B respectively were not independent. While increasing the number of crossover events would not break this dependency, it could lower the effect the dependency had in causing bloat.

Second, adding more nonhomologous crossover events meant adding more variation (more noise) into the breeding procedure. This gave us a dial to turn which would effectively change the amount that crossover "randomized" individuals (an idea inspired by arguments made in [Angeline 1997]). Would more randomization be beneficial or detrimental to GP?

The experiments were done as follows. We ran for 51 generations, including the initial generation, using a population of 500, and tournament selection of size 7. Multi-crossover was the sole operator used. The three problem domains chosen were Symbolic Regression, 11-Bit Boolean Multiplexer, and Artificial Ant. Symbolic Regression used no Ephemeral Random Constants. Artificial Ant used the Santa Fe Trail. All other run and problem domain parameters were done as stipulated in [Koza 1992]. The evolutionary computation system used was ECJ [Luke 2001a].

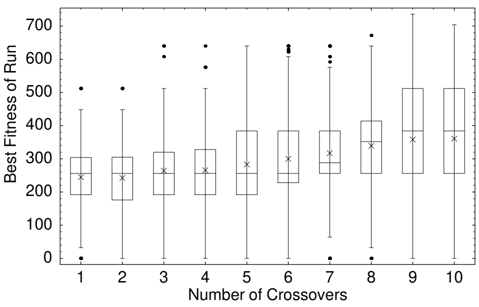

Figure 5: Boxplots of the distribution of best-fitness-of-run, by number of crossovers, 11-Bit Boolean Multiplexer domain. Lower fitness is better. Compare to Table 1.

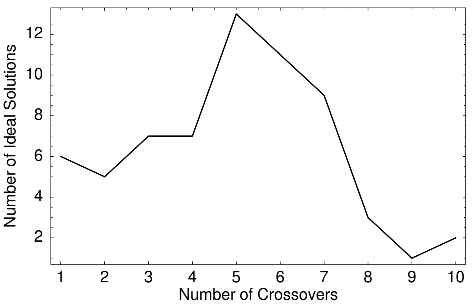

Figure 6: Number of ideal solutions found, by number of crossovers, 11-Bit Boolean Multiplexer domain.

We performed ten different experiments, with multi-crossover set to 1 through 10 crossovers respectively. Each experiment consisted of 500 independent runs. This number of runs is much higher than is necessary to produce best-fitness-of-run results, but we needed as many runs as possible to feel at least partially confident in our ideal-solution counts, and 15,000 total runs was the most we could afford to do. For each problem domain we used boxplots[5] to plot best-fitness-of-run distributions for different numbers of crossovers. We also plotted the number of ideal solutions found, and the Cumulative Probability of Success and Computational Effort measures.

3.1 SYMBOLIC REGRESSION RESULTS

Symbolic Regression did in fact lower tree size. But it did so at the cost of a statistically significant worsening of fitness, though only gradually: large swaths of crossover experiments were in the same statistical equivalence class, as shown in Table 1. Increasing the amount of noise in the crossover procedure, thus moving the system more towards

[5]In a boxplot, the rectangular region covers all values between the first and third quartiles, the stems mark the furthest individual within 1.5 of the quartile ranges, and the center horizontal line indicates the median. Dots show outliers, and $\times$ marks the mean.

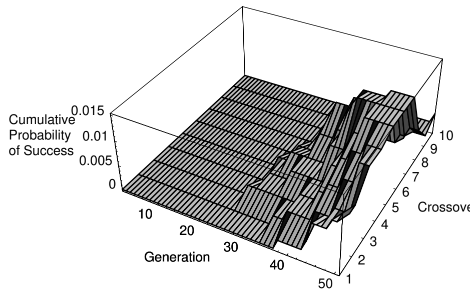

Figure 7: Cumulative Probability of Success per generation, by number of crossovers, 11-Bit Boolean Multiplexer domain.

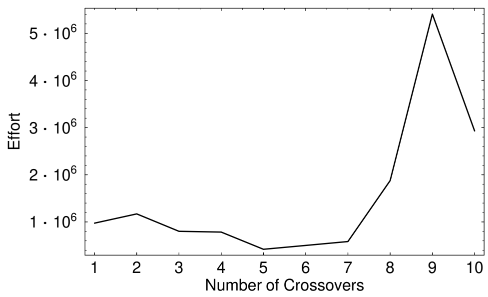

Figure 8: Computational Effort by number of crossovers, 11-Bit Boolean Multiplexer domain.

random search, yielded worse results on average, using the mean best-fitness-of-run measure as shown in Figure 1.

But one would not have known this from the ideal-solution counts. As noise in crossover increased, the number of ideal solutions increased rapidly from 150 per 500 runs with a single crossover, to stabilizing at about 350 per 500 runs with seven crossovers or more, as shown in Figure 2. This in turn resulted in an unexpected Cumulative Probability of Success curve, and a major decrease in Computational Effort as the number of crossovers increased, as shown in Figures 3 and 4.

Symbolic Regression is the easiest of the three problem domains presented: it finds the ideal solution very often (no less than 30% of the time in these experiments). Thus our counts were very high and the Cumulative Probability of Success curve was very smooth.

This outcome was very disturbing. Were we to have used ideal-solution count measurements as our basis of comparison in this experiment, our conclusions wouldn't just have been uncorrelated with best-fitness-of-run results: they

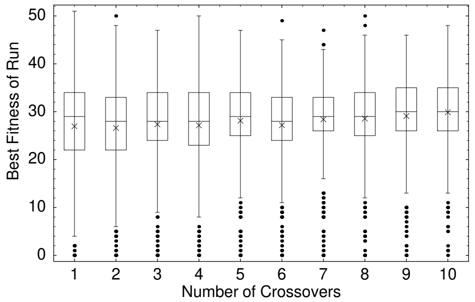

Figure 9: Boxplots of the distribution of best-fitness-of-run, by number of crossovers, Artificial Ant domain. Lower fitness is better. Compare to Table 1.

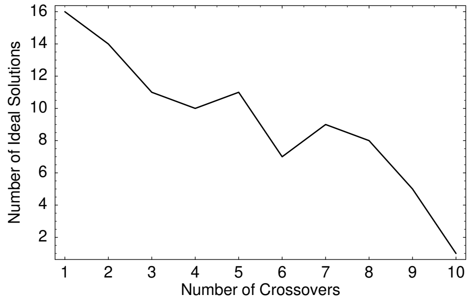

Figure 10: Number of ideal solutions found, by number of crossovers, Artificial Ant domain.

would have been the *opposite*. We would have concluded that increasing number of crossovers does a better job.

3.2 11-BIT BOOLEAN MULTIPLEXER RESULTS

11-Bit Boolean Multiplexer was to yield another surprise. First, it too lowered tree size, and like Symbolic Regression, it did so by statistically significantly worsening the fitness results. Again, increasing the amount of noise yielded worse results on average, when using the mean best-fitness-of-run measure, as shown in Figure 5. This time, the fitnesses worsened rapidly, with few in the same statistical equivalence class, as seen in Table 1.

Given its tendency to bloat like Regression does, we fully expected Multiplexer to have similar ideal-solution counts. But this was not quite the case. As the number of crossovers increased, the number of ideal solutions increased, but then it then decreased again. Multiplexer is a relatively more difficult problem domain to find ideal solutions in: thus we found no more than thirteen ideal solutions in 500 runs, with the maximum peaking when we applied five crossovers. The nadir was a single solution discovered, when we applied nine crossovers, as shown in Figure 6.

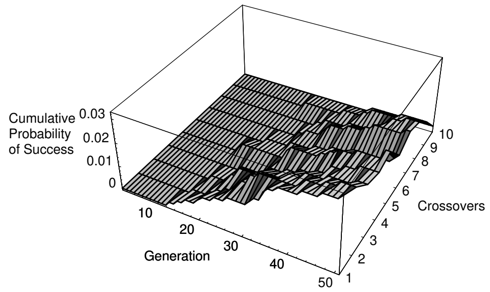

Figure 11: Cumulative Probability of Success per generation, by number of crossovers, Artificial Ant domain.

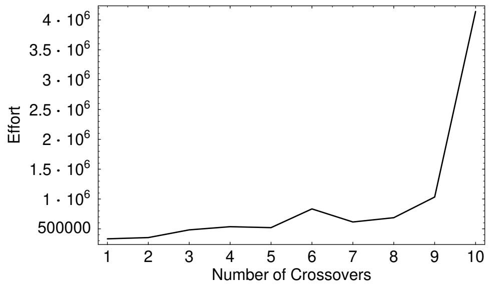

Figure 12: Computational Effort by number of crossovers, Artificial Ant domain.

The Cumulative Probability of Success similarly followed a bell curve distribution, as is shown in Figure 7. The relatively low number of ideal solutions resulted in the expected exaggerated swings in Computational Effort, as shown in Figure 8.

Here, while a best-fitness-of-run measure would state that increasing crossovers generally decreased quality, an ideal-solution count measure would argue that increasing crossovers somewhat (to five) made the result twice as good.

3.3 ARTIFICIAL ANT RESULTS

Artificial Ant gave us yet *another* result. Again, it lowered tree size, and once again, it also did so by statistically significantly worsening the fitness results (Figure 9) though like Symbolic Regression the results worsened only slightly, resulting in large numbers of experiments in the same statistical equivalence class (Table 1).

Figure 10 reveals that this time, the ideal-solution counts followed the statistically significant fitness: increasing the amount of noise decreased the number of ideal solutions discovered. As shown in Figure 11, the Cumulative Probability of Success followed a radically different curve than was the case in Symbolic Regression. Like Multiplexer, Artificial Ant discovers relatively few ideal solutions (no more than sixteen out of 500). And like Multiplexer, this exaggerates the Computational Effort results (Figure 12).

Here, as in Symbolic Regression, one would conclude on the basis of the best-fitness-of-run measure that the mean is getting just a little worse, though statistically significantly. But here an ideal-solution-count measure suggests that increasing the number of crossovers considerably worsens the result.

4 WHAT'S GOING ON?

For all three problems, fitness gradually worsened as the number of crossovers increased. But each problem yielded a wildly different result in the ideal-solutions count, and thus in the Cumulative Probability of Success and Computational Effort metrics. In short, the ideal-solution counts were not correlated with the best-fitness-of-run measure.

We think the reason for this phenomenon is that, as shown in Figures 1, 5, and 9, the number of ideal solutions is more closely linked to the variance of — rather than the mean of — the best-fitness-of-run distribution. As the variance increases, fitness is increasingly scattered both up and down. But there is a bound at zero (one cannot be better than perfect), whereas there is no bound on getting worse. Thus a wider variance tends to rack up more perfect scores. This trend is echoed in the data for all three problem domains, although the ideal-solution counts in the Artificial Ant and Multiplexer domains are low enough to make us wary about making a pronouncement.

In the Symbolic Regression domain, the distributions are skewed. As the number of crossovers increases, the mean increases a little, but not nearly as much as the variance does. Thus a worsening in the mean is not able to prevent ideal solutions from piling up. In the Multiplexer domain, the mean worsens only a little initially but rapidly at the end, while the variance generally increases steadily. This allows the mean to lose to, then catch up with the variance, which might explain the ideal-solution counts. Lastly, the Artificial Ant domain's mean worsens slowly while the variance *decreases*, which can explain the steady decrease in ideal-solution counts.

This combination of mean and variance has a closely related effect: even if low-variance technique A has a higher expected value than high-variance technique B for a single run, B can still have a higher expected best result of N independent runs (Figure 13). This phenomenon was explored in [Luke 2001b].

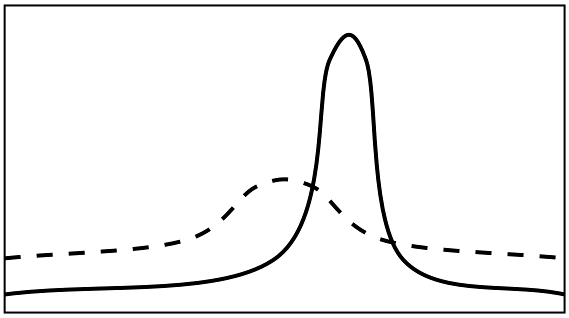

Figure 13: Example of two distributions which differ in mean and in variance. While one sample from the flatter distribution will give a lower expected result (the mean) than the tall distribution, in this case the best of two or more samples from the flatter distribution will give a *higher* expected result.

We will not venture a guess as to *why* increasing the number of crossovers changed the mean and variance results in the way it did. We had expected that increasing the number of crossovers would consistently increase the variance, but clearly it does not.

Regardless of the outcome, we do not believe that these experiments bode well for the body of GP literature which relies on ideal-solution counts to compare the quality of different techniques. We are concerned that upon reading this literature, experimenters may mistakenly conclude that certain techniques are better than others on average, when in some cases better ideal-solution counts are actually indicative of a *worse* result in mean best-fitness-of-run.

5 RECOMMENDATIONS

One downside to mean-best-fitness-of-run results is demonstrated in the Symbolic Regression domain: techniques which produce large, bloated trees that are functionally "close" to the ideal are given high marks, even though the results do not remotely resemble our notion of what the ideal individual ought to look like. Ideal-solution counts can be somewhat useful here in weeding these pretenders out. But given their statistical problems, we feel that ideal-solution count results alone simply cannot be justified.

If such incrementalism is the critical sticking-point tempting an author to only report ideal-solution counts, we suggest instead that the author produce results showing both mean best fitness of run *and* mean tree size. If a technique is superior both in tree size and mean best fitness, it is more difficult to argue that the improvements in mean best-fitness-of-run are due to incrementalism, since this would likely also increase tree size. Another approach would be to change the methodology to one showing generalization, using training and testing sets. Incrementalism tends to result in solutions custom-tailored to the training set: such poor generalizability would then show up when the final solution is compared to the testing set.

If instead one were performing runs in a domain which demanded perfection, then it would obviously be useful to know how often a technique was likely to find the ideal solution. Another situation also commonly arises: trying to beat an existing record, where it's useful to know not if a technique is likely to find the ideal, but whether or not it is likely to find any solution better than the record. In these situations we suggest that counts could supplement, but not replace, the mean best-fitness-of-run results.

But for most cases we suggest a different point statistic to use as a supplement. The procedure specified in [Luke 2001b] gives the expected maximum best-fitness-of-run for N total runs. This procedure, like the ideal-solution-count procedure, suffers statistically from the assumption that the sample is exactly representative of the population, and from its reuse of statistically dependent data. However, as a supplement to mean best-fitness-of-run results, we think it is a more useful metric than ideal solution counts in most cases.

The procedure is as follows. Perform a large number m runs for a given technique T. Sort the runs by fitness and assign ranks $1,...,m$ to the runs, where rank 1 is the worst-fitness run for the technique. Let $F(r)$ be the fitness of the run ranked r. Then if one were to perform N runs and return the best run, the expected best fitness $E(T,N)$ among those N runs would be:

$$E(T,N) = \sum_{r=1}^{m} F(r) \frac{r^N - (r-1)^N}{m^N}$$

This presumes that higher fitness values are better. Now consider two techniques A and B, where beyond some point $N \geq C$, $E(B,N)$ is consistently greater than or equal to $E(A,N)$. Ideally, C would be 1. This would lend evidence to the belief that not only is A better than B on average, but it is also superior no matter how many runs you are likely to perform. This gives strong weight to the claim that A really is better than B. Further, it seems likely that if technique A finds many more ideal solutions than technique B, that $E(A,N)$ will surpass $E(B,N)$ above some point N where finding ideal solutions with A becomes sufficiently common.

In any case, given their grievous statistical problems, we strongly urge that an ideal solution count measures never be used alone as proof that one technique is superior to another, except under special circumstances and with the appropriate disclaimers. If used at all, they should be only used to bolster a more statistically viable fitness comparison procedure.

Last, we recommend that experimenters more closely adopt difference-of-means tests (at least t-tests and ANOVAs), and a reasonable sample size (30 at a minimum) in published evolutionary computation experiments, or give justifications for doing otherwise.

6 CONCLUSION

Many papers in the GP literature use ideal-solution counts in one way or another, usually to compare the quality of techniques. This notion was popularized by Koza's Cumulative Probability of Success, Individuals to be Processed, and Computational Effort measures. This use continues despite warnings that counts are poor estimates, as they are a point statistic with no associated means test; as they make assumptions about dependencies across generations; and as they are exaggerated by small count sizes.

In this paper we reiterated this warning about counts, and noted motivational concerns. Specifically, we noted that for most "difficult" problems, the goal is to find *as good* a solution as possible. Counting the number of times the ideal solution is found does not help achieve this goal. Further, if one can find the ideal reliably, then the problem is trivial.

We also demonstrated the disturbing fact that ideal solution counts are not well correlated with mean best-of-fitness measures. In fact, for some problem domains, we showed that ideal solution counts may lead to the *opposite* conclusion that mean best-of-fitness measures lead to. This begs a reevaluation of much of the GP literature, as published results may be dubious, and in some cases the opposite of their intended meaning.

Acknowledgments

Our thanks to Lee Spector, Paul Wiegand, and Ken De Jong for their insights and responses, and to our reviewers for their helpful comments.

References

Angeline, P. J. (1996). An investigation into the sensitivity of genetic programming to the frequency of leaf selection during subtree crossover. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 21–29, Stanford University, CA, USA. MIT Press.

Angeline, P. J. (1997). Subtree crossover: Building block engine or macromutation? In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings* *of the Second Annual Conference*, pages 9–17, Stanford University, CA, USA. Morgan Kaufmann.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, dpunkt.verlag.

Daida, J., Ross, S., McClain, J., Ampy, D., and Holczer, M. (1997). Challenges with verification, repeatability, and meaningful comparisons in genetic programming. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 64–69, Stanford University, CA, USA. Morgan Kaufmann.

Daida, J. M., Ampy, D. S., Ratanasavetavadhana, M., Li, H., and Chaudhri, O. A. (1999a). Challenges with verification, repeatability, and meaningful comparison in genetic programming: Gibson's magic. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1851–1858, Orlando, Florida, USA. Morgan Kaufmann.

Daida, J. M., Yalcin, S. P., Litvak, P. M., Eickhoff, G. A., and Polito, J. A. (1999b). Of metaphors and darwinism: Deconstructing genetic programming's chimera. In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 453–462, Mayflower Hotel, Washington D.C., USA. IEEE Press.

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

Langdon, W. B., Soule, T., Poli, R., and Foster, J. A. (1999). The evolution of size and shape. In Spector, L., Langdon, W. B., O'Reilly, U.-M., and Angeline, P. J., editors, *Advances in Genetic Programming 3*, chapter 8, pages 163–190. MIT Press, Cambridge, MA, USA.

Luke, S. (2000). *Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat*. PhD thesis, Department of Computer Science, University of Maryland, A. V. Williams Building, University of Maryland, College Park, MD 20742 USA.

Luke, S. (2001a). ECJ 7: An evolutionary computation research system in Java. Available at http://www.cs.umd.edu/projects/plus/ec/ecj/.

Luke, S. (2001b). When short runs beat long runs. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 74–80, San Francisco, California, USA. Morgan Kaufmann.

Luke, S. and Spector, L. (1997). A comparison of crossover and mutation in genetic programming. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 240–248, Stanford University, CA, USA. Morgan Kaufmann.

Luke, S. and Spector, L. (1998). A revised comparison of crossover and mutation in genetic programming. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., and Riolo, R., editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 208–213, University of Wisconsin, Madison, Wisconsin, USA. Morgan Kaufmann.

Paterson, N. and Livesey, M. (2000). Performance comparison in genetic programming. In Whitley, D., editor, *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pages 253–260, Las Vegas, Nevada, USA.