Achieving COSMOS: A Metric for Determining When to Give up and When to Reach for the Stars

Emma Tosch
Computer Science
University of Massachusetts
Amherst, MA 01003
etosch@cs.umass.edu

Lee Spector
Cognitive Science
Hampshire College
Amherst, MA 01002
lspector@hampshire.edu

ABSTRACT

The utility of current metrics used in genetic programming (GP) systems, such as computational effort and mean-best-fitness, varies depending upon the problem and the resource that needs to be optimized. Inferences about the underlying system can only be made when a sufficient number of runs are performed to estimate the relevant metric within some confidence interval. This paper proposes a new algorithm for determining the minimum number of independent runs needed to make inferences about a GP system. As such, we view our algorithm as a meta-metric that should be satisfied before any inferences about a system are made. We call this metric COSMOS, as it estimates the number of independent runs needed to achieve the Convergence Of Sample Means Of the Order Statistics. It is agnostic to the underlying GP system and can be used to evaluate extant performance metrics, as well as problem difficulty. We suggest ways for which COSMOS may be used to identify problems for which GP may be uniquely qualified to solve.

Categories and Subject Descriptors

I.2 [**Learning**]: Parameter Learning

General Terms

Algorithms, Performance

Keywords

Evolutionary Computation, Order Statistics, Convergence

1. INTRODUCTION

Over the past ten years, an increasing number of researchers have published criticisms of the statistical validity of popular metrics used in GP. These criticisms have been inspired by Angeline's original observation that Koza's computational effort [9] is a random variable [1] [2]. This body of work has contributed a variety of conclusions, ranging from modifying how cumulative probability of success [9] ought to be calculated, analyzed, and used [13] [17] [16] [18] [4], to questioning the very semantics of such metrics [12]. Some of the criticisms cast upon computational effort have been echoed for other metrics [15], revealing an underlying discomfort with the purpose of and statistics behind performance metrics in GP.

Designing new metrics requires a balance between the theoretically justified approaches of mathematical statistics and the realities of data-driven analysis. An apt illustration of this balance can be seen in the choice of algorithm for hypothesis testing. Daida et al. use the Mann-Whitney U-test, a non-parametric measure, in their work [5]. While there is strong justification for using this approach, Paterson and Livesey later found that such nonparametric approaches were no more useful than t-tests [14]. This example illustrates the merit of choosing techniques for analysis on the basis of sound statistical argumentation and then revising such approaches via experimental verification. This perspective strongly motivates the framework of COSMOS.

Any GP experiment can be viewed as a three-tiered system, where the parameter settings and variable choices at each tier contribute to the final outcome. At the first tier we have problem parameters (e.g. instruction sets). At the second tier we have system parameters (e.g. genetic operators and their settings). At the third tier we have experiment parameters, defined by the population size, the maximum number of generations, and the number of independent runs. The technique described in this paper focuses upon this third tier of parameters. We focus most intensely upon determining the appropriate number of independent runs for a system.

While others have addressed the importance of performing a sufficient number of independent runs, so far as we know the only paper to treat the question experimentally has been Luke 2001 [11]. This work specifically addressed the tradeoff between independent runs and number of generations. Our approach and motivation differs; where Luke addressed the question "Given finite CPU hours, how should I best schedule my runs?," we address the question, "Given a GP system with a set of parameters for some problem, how many runs should we perform before we are confident in our outcome?"

We answer this question using the convergence of the sample means for the order statistics, which we call COSMOS. We treat the population of individuals at some generation i as a sample drawn from the underlying program space. Of course, since an individual is rich collection of features, we

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.

must choose some function of the individual to study. The quality of this function has a large impact on the quality of this metric. We choose the fitness function, since its design ought to give complete information about the quality of an individual. Though we are now handling functions of individuals, we will refer to both the individual and the value of the function of the individual (i.e. fitness) as the individual and will clarify only when the context is not clear.

Once we have chosen the function of interest, we can order individuals. Then, depending upon our problem and system parameters, we choose a set of ordinals.[1] Take, for example, the individual with the maximum fitness. We then calculate the sample mean of the maximum fitness across all runs for some generation i. When we are confident that an additional data point will not have a significant impact on the sample mean of the maximum fitness at generation i, we return a recommended minimum number of runs for that generation. This process is performed across all selected ordinals across all runs, culminating in the return of the maximum number of runs. For reasons we will discuss in Section 2, it is possible for the recommended number of runs to not converge. If we find that the system converges, we argue that it is then statistically sound to make inferences about the data we have collected using that system.

The remainder of the paper is organized as follows. Section 2 reviews the mathematical constructs that underpin our arguments. Section 3 proceeds with a detailed description of the COSMOS framework. Section 4 describes the experiments we performed to first verify our hypotheses about the behavior of COSMOS. We conclude with recommendations on how to use COSMOS and outline our proposed future work in sections 6 and 5 respectively.

2. MATHEMATICAL PRELIMINARIES

Any entity in a stochastic system whose value is uncertain can and should be treated as a random variable. Recall that a random variable is drawn from an underlying distribution that is defined over a set, called its support. A probability density function (pdf) is a function that maps events to probabilities. The domain of a random variable is then the set of all possible events. All definitions for statistical terms are drawn from [3] unless otherwise noted.

We begin by considering how Koza's $\mathcal{Y}$[2] is generally treated, since this will provide the foundation for the event space of the random variables described in COSMOS. Koza's $\mathcal{Y}(i)$ is stated to be the probability that at least one individual satisfying some success predicate, which we shall call s, is found at generation i. This can be computed by counting the number of independent runs that find a successful individual at generation i and dividing the sum by the total number of runs. Let

$$g(i) = \begin{cases} 1 & \exists j \in M_i : s(j) = 1 \\ 0 & \forall j \in M_i : s(j) = 0, \end{cases}$$

where the value of s indicates whether the predicate has been met in the obvious way and M denotes the population. Then $\mathcal{Y}$ can be more clearly stated in terms of its complement, $\mathcal{Y}(i) = 1 - \mathbb{P}(g(i) = 0)$, since by the Kolmogorov axioms, we know that the probability that a given generation contains at least one success is the sum of the probabilities that a given generation contains from one success up to $|M|$ successes.

Suppose that we are running some GP system and are currently in generation i. We want to know the probability that the next generation contains a success. The event space described by this scenario is different from the one used to compute $\mathcal{Y}$. Since the composition of each generation depends upon the generation that preceeded it, the probability in question here would be described as

$$\mathbb{P}(g(i+1) = 1 | M_0, g(i) = 0, ..., g(0) = 0)$$

This difference highlights the necessity of studying the effects of performing multiple runs. We must not only perform multiple independent runs to estimate $\mathcal{Y}$, but also in the experimental process, to make metrics such as $\mathcal{Y}$ meaningful. That is, if we were to use $\mathcal{Y}$ to determine the number of generations needed to satisfy s, we would be basing such a decision on a metric that describes a larger event space than the one we are currently considering. We assume when we compute $\mathcal{Y}$ that the sample is sufficiently large to represent the underlying event space. This event space is the set of all possible populations reachable by generation i. Clearly this set has the potential to be intractably large. However, if we suppose that we have a large enough sample, i.e. number of runs, we can estimate the marginal probability, giving us $\mathcal{Y}$.

Since we do not know the underlying distribution of $\mathcal{Y}$, we must devise an estimator to approximate it. Recall that an estimator is a function from a sample to some value in the domain of the random variable being estimated. Calculating the frequency of generations containing a success, as described above, is an estimator for $\mathcal{Y}$, which we denote $\hat{\mathcal{Y}}$. As has been noted in [17] and [16], computational effort is a point estimator and should be reported with an appropriate confidence interval. Since computational effort can be seen as a function of $\mathcal{Y}$, the same argument applies.

Like $\mathcal{Y}$, COSMOS draws from the marginalized event space. Though drawing from the same underlying data set, COSMOS computes an entirely different set of values. Rather than considering the indicator random variable defined by g, COSMOS records the frequency map for some function f over every generation i for every run k. For now we will allow f to be the fitness function. However, as a result of this, all subsequent assertions about the behavior of COSMOS *are only as good as the fitness function.* Further implications are discussed in Section 5.

We let M_{i_k} denote the population of run k at generation i. Each of the j individuals in M_{i_k} can be ordered based upon $f(j)$. These individuals are treated as random variables, and so any function of the individuals is itself a random variable, provided that it meet certain criteria, again discussed in further detail in Section 5. Call the b^{th} ordered random variable produced by $f(j), X_{(b)}$. Then $X_{(b)}$ is the b^{th} order statistic.

Each b^{th} order statistic has a pdf that is defined by the underlying population's cdf and pdfs. These are not guaranteed to have a closed form. We will not be considering the form of the order statistics' pdfs, since they are complex and not analytically relevant to our analysis. The important point to note is that they depend upon both the pdf

[1] "Ordinals" throughout the paper will refer to the set of chosen order statistics.

[2] $\mathcal{Y}$ is the first statistic calculated when computing computation effort. We focus our attention on it because its computation is relevant to COSMOS. Since computational effort is a function of $\mathcal{Y}$, any assertions we make about applicable properties of $\mathcal{Y}$ also hold for computational effort

and cdf of the underlying distribution of the sample and thus also rely upon a unified event space. Order statistics are commonly used in nonparametic statistics, where data reduction is not possible. While it is possible that some problems may have, for example, populations that can be described by an exponential family distribution, we cannot make this assumption across all problems and so must assume that the underlying probability distribution is represented by an arbitrarily large number of parameters. While in theory we recommend computing COSMOS for all available order statistics, we recognize the infeasibility of such a task and instead recommend the standard min, max, and quartile samples.

Now that we have motivated our choice of object of study, we will provide the justification for the algorithm that follows. We appeal to the weak law of large numbers, which states that if $X_{(b)_0}, ..., X_{(b)_k}$ are independent and identically distributed (iid) and $\exists \mu : \mathbb{E}(X_{(b)_k}) = \mu$, then $\lim_{k\to\infty} \overline{X}_{(b)} = \mu$ (in probability), where $\overline{X}_{(b)}$ is the sample mean of the b^{th} order statistic. This gives rise to the name of our metric, since to answer the question of how many runs we need to perform, we will be looking for the convergence of the sample means for the order statistics. If the two assumptions on which the weak law of large numbers is predicated are true, then we know that this sample mean will eventually converge.

Before we continue with the algorithm, we will need to address two other considerations. The first is the validity of our assumptions. We have already argued how and why the b^{th} ordered sample at the i^{th} generation for any given run is independent from its analogue in any other run. The remaining assumption that we must consider is whether the underlying distribution has an expectation. Consider the case where the probability does not converge; then we know that one of our assumptions is wrong and would like to believe that the most likely candidate is the existence of the function's expectation. While this does not bode well for our computational resources, it does reveal an exciting property of COSMOS, allowing us to identify problems that GP may be uniquely qualified to explore. This aspect of COSMOS is discussed further in Section 6.

The second consideration is how to translate a statement about behavior occurring in the limit to the situation where we are constrained to a finite number of runs. COSMOS is an algorithm that either returns a recommended number of runs or an error indicating that more runs are needed. As such, it is more appropriately described as an estimator for the number of runs after which additional runs will only perturb the sample mean of each of the order statistics by some multiplicative error. We estimate its error using the bootstrap method [8].

Algorithm 1 COSMOS estimator. We start with 30 independent runs and calculate the mean fitnesses for each of the ordinals for these 30 runs. We then recalculate the means as we add more runs, until the ratio of the k^{th} and $k^{th}+1$ means are within a given range.

Input: ϵ, the acceptable error in successive means
Output: Recommended minimum number of runs

```
    R ← 2r independent experiments
    Partition R into two sets, R_1 and R_2
    A ← new array of size |Q|, initialized with zeros
    for generation i ← 1 to maximum generation do
 5:     for ordinal q ∈ Q do
            x̄ ← (1/r) Σ_{j∈R_1} err(q, i, j)
            r' ← r
            for k ∈ R_2 do
                x̄' ← (r * x̄ + err(q, i, k))/(r + 1)
10:             if 1 − ε < x̄/x̄' < 1 + ε then
                    A[q] ← r + 1
                    exit loop over R_2, begun at Line 8
                else
                    x̄ ← x̄'
15:                 r' ← r' + 1
                end if
            end for
        end for
    end for
20: if ∃a_k ∈ A : a_k = 0 then
        R_2 ← r more experiments
        Repeat this algorithm from Line 4
    else return max(A)
    end if
```

3. ALGORITHM

We first define our inputs, constants, and associated terms in Table 1. These terms are used in Algorithm 1.

This algorithm begins by launching a pre-specified minimum set of runs. We set our initial r to be 30. This number was chosen arbitrarily. The baseline number of runs could have been 10; it could also have been 50 or 100. We would like it to be large enough so that the chances of calculating the ratio of the k^{th} mean and the $k^{th}+1$ mean are small, but small enough so that we do not waste computational resources performing an excessive number of runs.

We argue that our algorithm stops launching new runs when the sample means for the set of ordered random variables converges. We would like to be able to say that for all k beyond some k_0^{th} run, $\overline{X}_{(b)_k} = (1 \pm \epsilon)\overline{X}_{(b)_{k+1}}$. ϵ is the true multiplicative error tolerance for the convergence of the of the sample mean for the random variable $X_{(b)}$. Then ϵ is also a random variable that we must estimate. We treat the "true" ϵ as an input to the algorithm. Consider, for increasing r, how ϵ behaves. We know that in the limit ϵ approaches zero. Its range for small r is determined by f, which consider here to be the fitness function. Our estimator only considers the multiplicative error for the last update.

Since it would be computationally expensive to calculate the means of all the order statistics for large populations (as well as difficult to graph!), we choose some subset of ordinals whose means to compute, which we call Q. The specific values of the ordinals will change depending upon the problem and the nature of the evaluation function. Standard practice is to use the minimum, maximum, and median ordinals, as well as the quartiles. For a population M, we let the minimum and maximum be their usual values, and set the ordinals located roughly at the median and upper and lower quartiles to be $M[\lfloor |M|/2 \rfloor]$, $M[\lfloor |M|/4 \rfloor]$, and $M[\lfloor 3|M|/4 \rfloor]$ respectively.

Table 1: Terms used in the COSMOS algorithm.

Name	Type	Default value	Description
r	constant	30	Minimum number of runs and run block size
r'	variable	30	The actual number of runs used to obtain our estimate
ϵ	input	0.01	Target multiplicative error
Q	constant	min, max, quartiles	Selected order statistics
$err(q, i, k)$	function	-	Returns the error of the k^{th} individual in generation i for run k

Table 2: Experiment Parameter Settings as reported in previous publications. k is the number of runs, $\max(i)$ is the maximum number of generations, and $|M|$ is the population size.

Source	Problem	k	$\max(i)$	$\|M\|$
Luke 2001	Symbolic Regression	50	8192	500
Luke 2001	Artificial Ant	50	2048	500
Luke 2001	Even 10-Parity	50	1024	200
Daida 2001	Binomial-3	600	200	500

Table 3: Baseline comparison of runs. k is the maximum number of recommended runs, $\min(i)$ is the minimum generation recommending the maximum number of runs, and Q' is the subset of ordinals recommending the maximum number of runs.

	Luke 2001		COSMOS		
Problem	Runs	Gens.	k	$\min(i)$	Q'
Sym. Regression	50	8	97	184	{0}
Artificial Ant	50	32	96	1879	{0}
Parity	50	∞	43	835	{50}

4. EXPERIMENTS

The following experiments were performed as proofs of concept. They establish the validity of COSMOS as a metric and illustrate the types of inferences that it allows us to make. Unless otherwise stated, all experiments use ECJ [10]. All parameter settings not mentioned are ECJ defaults, which correspond to those in [9], unless noted otherwise. For all experiments, we use i to denote generation, q to denote a particular ordinal drawn from the ordinal set, called Q. M is the population, which may be indexed by run or generation. Finally, runs are denoted by k.

4.1 Baseline experiments

Our first set of experiments replicated the three experiments performed in (Luke 2001) [11]. The three problems considered were Symbolic Regression, Artificial Ant, and Even 10-Parity. Symbolic Regression was run on $x^4 + x^3 + x^2 + x$ and used no ephemeral random constants (ERCs). Artificial Ant used the Santa Fe trail. These three problems use the same parameters as in [11], since this work addresses similar questions. Parameter settings are listed in Table 2.

The outcomes for these runs are listed in Table 3. Runs listed for (Luke 2001) are the number of independent runs performed. Generations for (Luke 2001) are the recommended minimum number of generations for some n runs. Since the 50 runs actually performed are all greater than the number of runs needed to reach the critical point identified, we list the minimum number of generations suggested. This is not meant to suggest that the author believes that optimal performance can be obtained by running the above reported number of runs for the listed number of generations. Rather, given his sample size of 50 runs per problem, he inferred the number of evaluations necessary for a critical point to be reached and calculated the minimum number of generations appropriately.

The numbers listed in the column marked k denote the numbers returned by the COSMOS algorithm. These are the maximum number of runs recommended over all generations and all $X_{(b)}$s. The column marked $min(i)$ is the minimum generation that had found the globally maximum number of runs. The column marked Q' is a subset of the Q order statistics for which the corresponding recommended runs is equal to the global minimum number of recommended runs at the minimum generation specified in the column marked $min(i)$. If the minimum, maximum, median, and quartiles all yield the same number of recommended runs, we indicate this by simply listing Q.

4.1.1 Symbolic Regression

Symbolic Regression yeilds the most interesting results. Due to issues of computational resources, we could only analyse two sets of incomplete data. The first set contains all ordinal recommended runs up to generation 50. The second set contain the minimum and lower quartile recommended runs for all generations. Both are plotted in Figure 1. While the lower quartile is fairly consistent, the minumum has high variability over a large range. In fact, six ordinals "max out," recommending the total number of runs that were performed, without reaching convergence. We will investigate such behavior in greater detail over a shorter generational period in the experiments that follow.

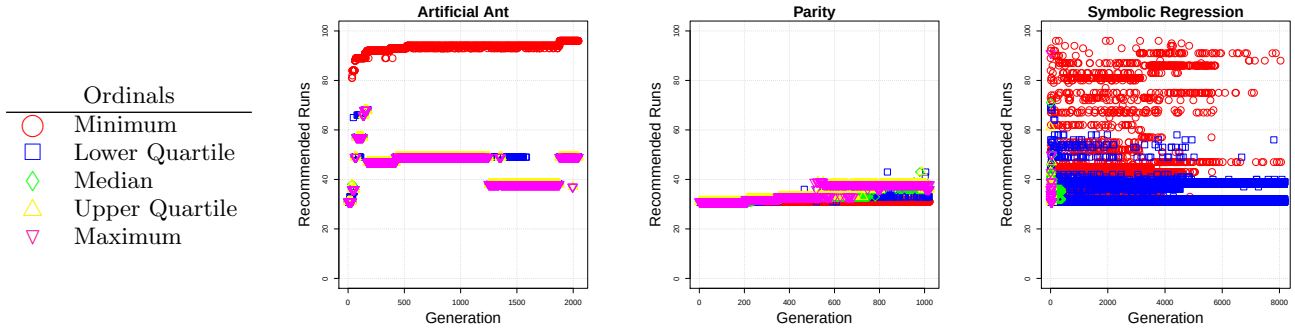
4.1.2 Artificial Ant

Given the incredibly small number of evaluations espoused by Luke [11], we expected an early convergence in Artificial Ant. Although the lowest generation for the maximum number of runs is fairly high, it is only marginally greater than previous generations' recommendations. In fact, it was a small amount of fluctuation in the other ordinals that allowed us to observe this variance. As shown in Figure 1, the maximum number of requested runs after the 250^{th} generation is fairly stagnant. Coupled with Luke's observations, this tells us that the version of the ant problem described here on ECJ is *extremely boring* and perhaps should only be used to establish baselines or test systems.

4.1.3 Even 10-Parity

The ∞ generations listed in Table 3 represents the behavior that Luke observed, that Even 10-Parity continues improving over the entirety of its run. In our experiment it was even more stagnant in its runs requests than Artificial Ant. Its convergence illustrates an important distinction between COSMOS and other metrics. COSMOS is not

Figure 1: Baseline Runs. Two-hundred ninety-nine independent runs each were performed for Artificial Ant and Parity. Ninety-seven independent runs were performed for Symbolic Regression. Not all runs were used in the calculations. These merely provided a pool from which we could sample for COSMOS.



evaluating whether a system is good at finding solutions; it evaluates whether a given system and a given problem reach stasis when exploring their solution space. What we see with Even 10-Parity is a system that remains consistent across its generations. It might behoove us to attempt calculating COSMOS with a smaller r to see if there is a lower bound on the required runs (other than the obvious lower bound of 1). Luke's recommendation that Even 10-Parity be run for an indefinite number of generations is consistent with our conclusions.

4.2 Looking to the COSMOS

We now turn our attention to a second set of experiments. We reproduced the work of (Daida et al 2001) [6]. The authors addressed how, all other things being equal, varying parameter settings of a GP system could dramatically alter problem difficulty. The object of study was the binomial-3 problem: $x^3 + 3x^2 + 3x + 1$. This problem had previously been discussed in [5] in the context of the effect that random number generators had on GP systems.

Seven different ERC ranges were tested: $[-0.1, 0.1]$, $[-1, 1]$, $[-2, 2]$, $[-3, 3]$, $[-10, 10]$, $[-100, 100]$, and $[-1000, 1000]$. The authors of [6] report having dramatically better hit rates for ERCs in the ranges $[-1, 1]$, $[-2, 2]$, and $[-3, 3]$ than for the remaining ranges. As is shown in Table 3, they only ran their experiments for 200 generations.

Figure 2 shows the results of these seven experiments. The zeroth ordinal (marked with a red square in the image) varied far more dramatically for these problems than it did for our baseline. We first note that all of the problems converged; that is, COSMOS found a minimum number of runs for every problem. Given the greater variability of this dataset, we validated with the bootstrap method.

4.2.1 Bootstrap Results

As mentioned previously, COSMOS is an estimator and so its quality needs to be assessed. Since we cannot know the underlying true values of our error term at a given number of runs, we must use a substitute method that has been shown to reliably estimate such a value.

The bootstrap method is a simple procedure for estimating the mean of some statistic of some sample via resampling. Given some sample space S, we let our sample size be some value n. We will resample n items from S some B number of times. For each sample of size n, we compute some statistic. In this case, it is the mean of the individuals' fitnesses. We then average over the B statistics to get the mean of means of the samples [7].

Here we let each of the sample means be calculated from k randomly chosen runs, where k is our recommended minimum number of runs we obtained via COSMOS. These means over k runs are each now random variables whose sample means we wish to find. We set B, the number of samples of size k, to be 100. The larger the size of B, the better. Since we have 600 runs for each range to choose from, we could have chosen even larger B. In any case, the results of our bootstrap are recorded in Table 4.

What does the bootstrap mean give us? COSMOS is effectively returning an estimate for the lower bound on the sample size for each of the B samples in the bootstrap. The sample means are means of the underlying order statistics. The bootstrap means are means of these means. The bootstrap means should follow a Gaussian distribution by the Central Limit Theorem. The sample means are point estimators for the distribution of order statistics, which as previously stated, depend upon the underlying fitness function's cdf. However, the assertion we are making is that the sample means are close enough to the bootstrap's mean, and by extension, the true mean of the underlying distribution. The column labelled "Outliers" in Table 4 lists the number of runs that had means that were outside of two standard deviations of the bootstrap sample mean, using the bootstrap variance. If the underlying distribution of the bootstrap is Gaussian, then according to the so-called "Empirical Rule," 95% of the data points ought to lie within two standard deviations of the mean. Note that almost half of the runs from the range $[-1000, 1000]$ are outside where they should be. As we will discuss in the next section, we believe that this is related to the fact that $[-1000, 1000]$ ought to have been run for more generations.

4.2.2 Qualitative Results

What does COSMOS tell us about the experiments we performed? Clearly convergence was achieved for every ordinal. Recall that Daida et al. found a significantly larger number of ideal solutions for the ranges $[-1, 1]$, $[-2, 2]$, and $[-3, 3]$. Note how higher ordinals in the lower generations for range $[-1, 1]$ vary more, but peter off beyond the 50th generation. The zeroth ordinal is highly concentrated about 35 recommended runs. We are not particularly concerned about the variation in the zeroth ordinal in the final generation, since it is not significantly more than the variation

Figure 2: Results of the Binomial-3 runs. The seven graphs correspond to the seven ephemeral random constant ranges. All other experiment parameters for these problems were identical.

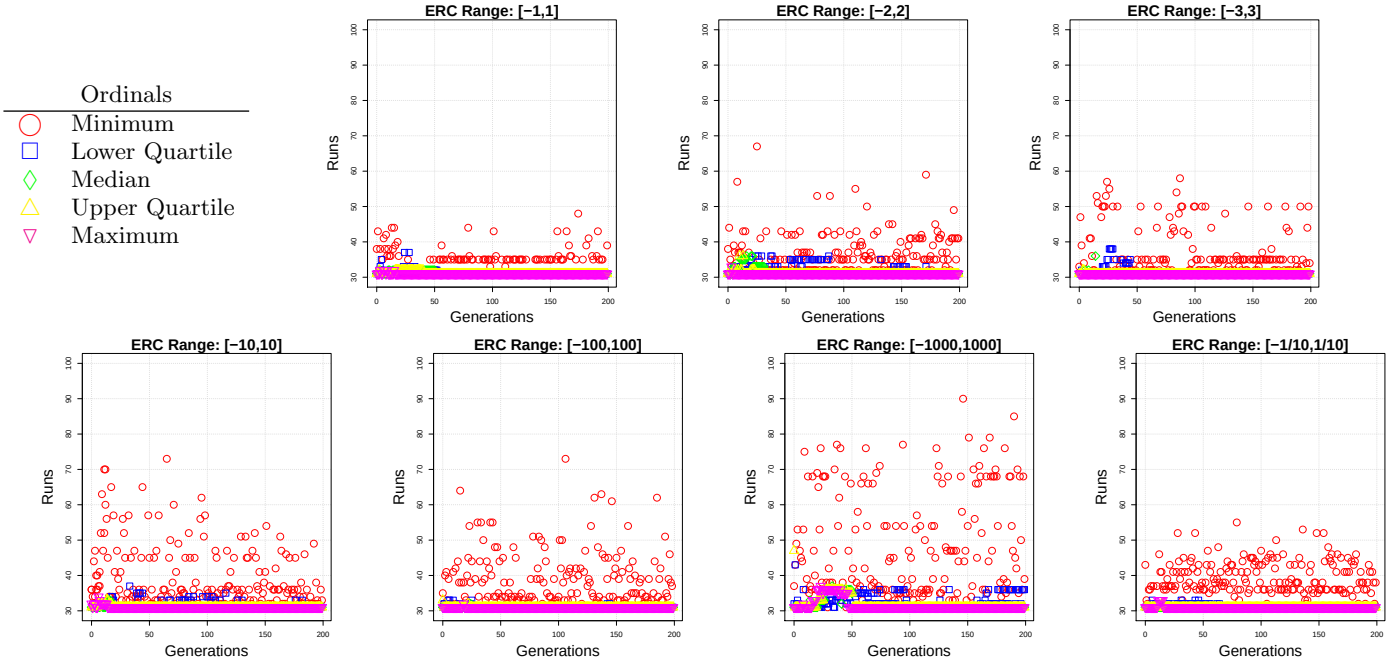


Figure 3: Comparison of sample means calculated as a result of the COSMOS algorithm against the bootstrap's means.

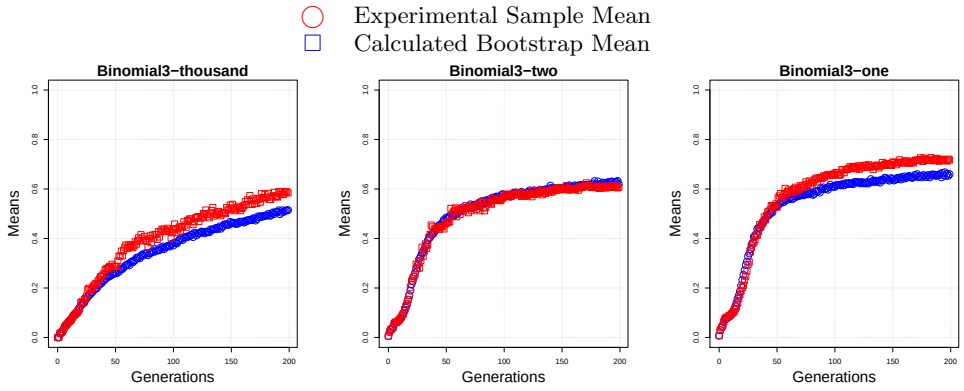


Table 4: COSMOS data for the Binomial-3 problem

ERC Range	k	$\min(i)$	Q	$\widehat{X}_{(k)_{\min(i)}}$	Bootstrap $\overline{X}_{(k)_{min(i)}}$	Bootstrap $S_{(k)_{min(i)}}$	Outliers
$[-0.1, 0.1]$	55	79	0	0.0025077515516979125	0.002426019616689402	3.0361977338903115E-7	18
$[-1000, 1000]$	90	146	0	3.5610724883002216E-4	1.1652259268579838E-4	3.628311661897609E-9	298
$[-100, 100]$	73	106	0	3.404128999658628E-4	2.756694720138498E-4	2.6154871044745273E-8	12
$[-10, 10]$	73	65	0	0.002012428747336066	0.002203810692886493	4.3686612445135404E-7	23
$[-3, 3]$	58	87	0	0.006597798204774809	0.007402139900564542	1.8373880939784997E-6	27
$[-2, 2]$	67	25	0	0.005312467235404943	0.00842622948054855	1.5694617101539245E-6	3
$[-1, 1]$	48	174	0	0.014475639242731488	0.013910600131531288	3.6513049530637364E-6	5

at the beginning of evolution. If we were unhappy with the performance of this system, in terms of producing ideal individuals, we would recommend running for more generations. The variability of the requests for more runs at the end of evolution indicates that there is still diversity in the worst individuals in the population, indicating that it is possible move out of a local maximum into a new portion of the program landscape.

The results for ranges $[-2, 2]$ and $[-3, 3]$ are similar, but with a greater range for the recommended runs. While the maximum number of runs recommended for the range $[-2, 2]$ is greater than the maximum number of runs recommended for the range $[-3, 3]$, $[-3, 3]$ a new cluster of recommended runs begins to emerge about the 50 mark.

Increasing the range to $[-10, 10]$ begins to give us the convexity of the zeroth ordinal that we have been expecting to see when the system converges and has been run for a sufficient number of generations. There appears to be some more variability in the ordinal corresponding to the lower quartile, but it is unknown whether this is statistically significant. We cannot compare it with the behavior of the lower quartile for lower ERC ranges because their variability is obfuscated by the minimum run requirement (here, 30). Its behavior appears very similar to the behavior of the range $[-0.1, 0.1]$.

The behavior of the range $[-100, 100]$ is less clear. It leads into an examination of the range $[-1000, 1000]$. We start to see more variability in all ordinals up to the fiftieth generation. Note that that the maximum number of runs recommended occurs late in evolution, at the 174^{th} generation. We would highly recommend running this experiment for more generations and re-evaluating.

5. FUTURE WORK

This paper is an exploratory analysis of a new approach for analyzing the performance of different genetic programming systems on different problems. As such, there are a number assumptions that ought to proven. There are also alternate avenues of investigation, which we examine below.

Improve the reliability of the algorithm. The results we found in the Binomial-3 example indicated that our algorithm may fail when the number of generations is insufficient. It also indicates that we may want to take the variance of the recommended runs into account.

There is one obvious area for improvement. The COSMOS algorithm estimates ϵ by only considering the ratio of the k^{th} and the $k^{th} + 1$ mean. Consider what might happen were to encounter samples drawn from a distribution that has an infinite expectation, e.g. a Cauchy distribution. It is possible to us to falsely conclude convergence when encountering a Cauchy distribution if we suddenly see a value that is within the multiplicative error of our current sample mean. However, given the high variance of such distributions, we consider the likelihood of encountering this situation to be very small. Furthermore, we chose to study the convergence of the sample mean of the order statistics, rather than the convergence of the sample medians of the order statistics, precisely because the mean is more sensitive to outliers. However, a more thorough mathematical and experimental analysis should be done. We recommend modifying the COSMOS algorithm to also track the rate of change of the ratios of the sample means and only return values when the error of the error is within some range. There is extensive related work in the machine learning literature regarding the estimation of hyper-parameters that may be useful for such modifications.

Draw from the MCMC literature for additional evaluation techniques. The choice of the number of initial runs, r, warrants further study. We feel it is directly analogous to determining the burn-in period of Gibbs Sampling.

Revisit the fitness function. Clearly individuals in a population are not single points from a single distribution, but instead a bag of features. If we take, for example, the error function of the individuals, this may represent an alternate distribution from the fitness function. We hope the distributions are the same, but do cannot know without investigating further. As mentioned earlier, COSMOS is only as good as the function generating our data. That function must map individuals to ordered values. If we consider unique genotypes (programs) to be elements of the set doing the mapping, then the error function may not be injective. This should not be surprising to most practitioners of genetic programming; the space of possible phenotypes is often significantly smaller than the space of genotypes. However, in the absence of a semantic understanding or ordering of the genotype, we are restricted in our analysis to meaningful measures - which in the above examples are error terms. We must be careful to not choose a function alters the ordering of the underlying system.

Use for knowledge discovery and building a taxonomy of problems. If you're reliably getting good results a high percentage of the time (based on traditional metrics), was the problem worth using GP to begin with? Could another technique have done better? Probably. We suggest using COSMOS to identify which problems are "interesting." GP's stochasticity gives it a clear edge for problems that confound simpler models. In other branches of machine learning, we are limited to a finite number of parameters and cannot jump out of our local space. GP allows us to explore areas of the program space that the designer may not have considered. We believe that high variability in the lower quantiles may allow a GP system to overcome deception. From this it would follow that GP is uniquely qualified to find problems that either (a) do not converge or (b) only converge with a large number of runs, while showing high variability in the recommended runs over many generations. These hypotheses should be tested in future work.

Expand the applicability of COSMOS**.** The methods described in this paper have thus far only been applied to genetic programming systems. We would like to verify its utility for a broader category of techniques, from the more general evolutionary computation community, to perhaps even other forms of stochastic search.

6. CONCLUSIONS

The work outlined herein began as a an attempt to evaluate the utility of popular performance metrics for genetic programming. As we considered the problems inherent in computational effort and mean-best-fitness, it became apparent that we needed to step back and reconsider whether we were even asking the right questions. We realized we couldn't adequately answer, "under some given parameter settings, can we find solutions (i.e. zero error programs) in fewer computations and/or a higher percentage of the time," until we answered, "given our black box system, cou-

pled with some input parameter settings, are we adequately exploring our program space?" While we have not answered that question here, we believe COSMOS provides a foundation for how to answer it. Past work has focused on describing problem landscapes, we accept that they are vast and possibly unknowable. We instead focus on determining how much of that space a given system with a given set of parameters can know. We also realize that performance metrics will vary depending upon the desired outcome. Fundamentally, we are not seeking one metric to rule them all. Instead we recognize the plurality of not only solutions, but systems.

7. ACKNOWLEDGMENTS

The authors would also like to thank Alexandre Passos for his statistical expertise and the CI Lab members at Hampshire College and Theory Lab members at the University of Massachusetts for their feedback. This work is supported in part by the National Science Foundation under NSF grants #CNS-0619337 and #1017817. Any opinions, findings, conclusions or recommendations expressed here are the authors' and do not necessarily reflect those of the sponsors.

8. REFERENCES

[1] P. J. Angeline. An investigation into the sensitivity of genetic programming to the frequency of leaf selection during subtree crossover. In *Proceedings of the First Annual Conference on Genetic Programming*, GECCO '96, pages 21–29, Cambridge, MA, USA, 1996. MIT Press.

[2] D. F. Barrero, B. Castaño, M. D. R-Moreno, and D. Camacho. Statistical distribution of generation-to-success in gp: application to model accumulated success probability. In *Proceedings of the 14th European conference on Genetic programming*, EuroGP'11, pages 154–165, Berlin, Heidelberg, 2011. Springer-Verlag.

[3] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, 2nd edition, June 2001.

[4] D. Costelloe and C. Ryan. On improving generalisation in genetic programming. In *Proceedings of the 12th European Conference on Genetic Programming*, EuroGP '09, pages 61–72, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] J. M. Daida, D. S. Ampy, M. Ratanasavetavadhana, H. Li, and O. A. Chaudhri. Challenges with verification, repeatability, and meaningful comparison in genetic programming: Gibson's magic. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2 of *GECCO '99*, pages 1851–1858, Orlando, FL, USA, 1999. Morgan Kaufmann.

[6] J. M. Daida, R. R. Bertram, S. A. Stanhope, J. C. Khoo, S. A. Chaudhary, O. A. Chaudhri, and J. A. I. Polito. What makes a problem gp-hard? analysis of a tunably difficult problem in genetic programming. *Genetic Programming and Evolvable Machines*, 2(2):165–191, June 2001.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2nd edition, Nov. 2001.

[8] B. Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[9] J. R. Koza. *Genetic programming: On the programming of computers by means of natural selection*. Complex adaptive systems. MIT Press, 1993.

[10] S. Luke. Ecj : An evolutionary computation research system in java.

[11] S. Luke. When short runs beat long runs. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 74–80, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[12] S. Luke and L. Panait. Is the perfect the enemy of the good? In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 820–828, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[13] J. Niehaus and W. Banzhaf. More on computational effort statistics for genetic programming. In *Proceedings of the 6th European conference on Genetic programming*, EuroGP'03, pages 164–172, Berlin, Heidelberg, 2003. Springer-Verlag.

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

[15] N. P. Troutman, B. E. Eskridge, and D. F. Hougen. Is "best-so-far" a good algorithmic performance metric? In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1147–1148, New York, NY, USA, 2008. ACM.

[16] M. Walker, H. Edwards, and C. Messom. Confidence intervals for computational effort comparisons. In *Genetic Programming. Proceedings of the 10th European Conference, EuroGP 2007*, 2007.

[17] M. Walker, H. Edwards, and C. Messom. The reliability of confidence intervals for computational effort comparisons. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1716–1723, New York, NY, USA, 2007. ACM.

[18] M. Walker, H. Edwards, and C. Messom. Success effort and other statistics for performance comparisons in genetic programming. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 4631–4638, September 2007.