The Reliability of Confidence Intervals for Computational Effort Comparisons

Matthew Walker, Howard Edwards, and Chris Messom
Institute of Information and Mathematical Sciences,
Massey University,
Auckland, New Zealand
{m.g.walker, h.edwards, c.h.messom}@massey.ac.nz

ABSTRACT

This paper analyses the reliability of confidence intervals for Koza's computational effort statistic. First, we conclude that dependence between the observed minimum generation and the observed cumulative probability of success leads to the production of more reliable confidence intervals for our preferred method. Second, we show that confidence intervals from 80% to 95% have appropriate levels of performance. Third, simulated data is used to consider the effect of large minimum generations and the confidence intervals are again found to be reliable. Finally, results from four large datasets collected from real genetic programming experiments are used to provide even more empirical evidence that the method for producing confidence intervals is reliable.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming—*Program synthesis*

General Terms

Experimentation, Measurement, Performance

Keywords

Genetic Programming, Computational Effort, Confidence Intervals, Wilson's Method

1. INTRODUCTION

In *Genetic Programming* [5], Koza described a statistic to assess the computational burden of using GP. It calculates the minimum number of individuals that must be evaluated in order to yield a solution 99% of the time. This statistic, minimum computational effort, E, was used heavily throughout Koza's first two books on GP [5, 6] to compare the performance of variations of GP.

Angeline [1] pointed out that a key problem with Koza's computational effort statistic was that, as defined, it was a point statistic with no confidence interval. Without a confidence interval, comparisons are inconclusive.

In an earlier paper [10] we offered three methods for the production of confidence intervals for minimum computational effort and concluded that a method based on Wilson's 'score' method [8] was the best choice: it is fairly easy to calculate, practically always produces a valid confidence interval, and the confidence intervals included the true computational effort an appropriate proportion of the time.

However the earlier work left a few questions unanswered. This paper extends our previous research and attempts to answer some of those open questions: How does dependence between the two variables used in the method affect performance? How does the method perform with different levels of confidence? Is the performance reliable for difficult problems which run for many generations?

1.1 Computational Effort

Given the cumulative probability of success, $P(i)$, of a number of GP runs, we can calculate how many runs would be required, $R(i,z)$, in order to find a solution at generation i with probability z[1] [10]:

$$R(i,z) = \begin{cases} \frac{\log(1-z)}{\log(1-P(i))} & \text{if} \quad P(i) < z \\ 1 & \text{if} \quad P(i) \geq z \end{cases} \tag{1}$$

The computational effort, $I(i,z)$ (the number of individuals that need to be evaluated to find a solution with probability z) for generation i with a population of M individuals is calculated by:

$$I(i,z) = (i+1) \times R(i,z) \times M \tag{2}$$

Koza's minimum computational effort, E, is the minimum value of $I(i,z)$ over the range of generations from 0 to the maximum in the experiment.

1.2 Producing confidence intervals

To calculate a 95% confidence interval for the true but unknown proportion of successes based on the observed sample proportion of successes, $p = r/n$, given r successes from n runs, Wilson's method [8] may be used (where the standard normal variable $z_{\text{norm}} = 1.96$):

$$upper(p,n) = \frac{2np + {z_{\text{norm}}}^2 + z_{\text{norm}}\sqrt{{z_{\text{norm}}}^2 + 4np(1-p)}}{2(n + {z_{\text{norm}}}^2)} \tag{3}$$

[1]As is common, z will be set to 0.99 throughout this work.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

$$lower(p,n) = \frac{2np + {z_{\text{norm}}}^2 - z_{\text{norm}}\sqrt{{z_{\text{norm}}}^2 + 4np(1-p)}}{2(n + {z_{\text{norm}}}^2)} \quad (4)$$

Using the formulae in equations 3 and 4, a confidence interval can be established for the proportion of successful runs: the upper bound is given by $upper(P(i), n)$ and the lower bound is given by $lower(P(i), n)$. The minimum and maximum of this range can then be used to calculate a maximum and minimum for the number of runs required to obtain a solution with probability z. These numbers can then be used with the known value for the population size, M, to find a 95% confidence interval for the true computational effort, $\mathcal{I}(i)$, at a given generation i[2]:

$$(i+1) \times R(upper(P(i),n)) \times M \leq \mathcal{I}(i) \leq (i+1) \times R(lower(P(i),n)) \times M \quad (5)$$

If j is the generation at which the minimum computational effort occurs (termed the *minimum generation*), then the confidence interval for $\mathcal{I}(j)$ can be used as a confidence interval for the true value of the minimum computational effort.

2. DEPENDENCE ISSUES

In our previous work, it was assumed that GP practitioners would elect to use all their run-data to produce estimates of both the minimum generation (j) and the cumulative probability of success at the estimated minimum generation ($P(j)$). This is in contrast to the method proposed by Keijzer et al. [4] who used half their runs to estimate the minimum generation and the other half to estimate the cumulative probability of success. The advantage of using their method is that the estimation of the two variables is statistically independent. This section analyses these two variations on both the Wilson and Resampling methods.

Wilson's method has been defined in section 1.2. If we accept dependence between the estimation of $P(j)$ and j, then j is the generation at which the minimum computational effort occurs when calculated using the entire set of runs. $P(j)$ is estimated by the proportion of runs (as calculated over the entire dataset) which found a solution at or before generation j. This is the method that was used in our previous work and for this section will be termed *Wilson-Dependent*.

If independence between the estimation of $P(j)$ and j is desired, then the dataset should be divided in two. For this work we considered a division of 1:1 (as Keijzer et al. proposed), but other ratios could be used. j is estimated as the generation at which the minimum computational effort occurs when calculated using the first part of the dataset. $P(j)$ is estimated by the proportion of runs (as calculated over the second part of the dataset) which found a solution at or before generation j. We will term this method *Wilson-Independent*.

In our previous work, the *Resampling-Dependent* method where the estimation of j and $P(j)$ were dependent was shown to be an inferior choice to Wilson-Dependent. The Resampling-Dependent method will not be considered any further in this paper.

[2]The addition of one generation in these formulae is a correction to the formulae provided in our previous work [10].

1. Obtain n independent runs. Label these as the source set.
2. Divide the source set into two parts (S_1 and S_2) with n_1 runs in S_1 and n_2 runs in S_2.
3. Repeat 10,000 times:
   (a) Select, with replacement, n_1 runs from the set S_1 (s_1).
   (b) Calculate j, the generation at which the minimum computational effort occurs, for the selection s_1.
   (c) Select, with replacement, n_2 runs from the set S_2 (s_2).
   (d) Calculate the computational effort at generation j for the selection s_2. If zero runs succeeded, the computational effort is infinite.
4. Find the 2.5% and 97.5% quantiles of the 10,000 minimum computational effort statistics. These provide an upper and lower range on a 95% confidence interval for the true minimum computational effort.

Table 1: Algorithm for the Resampling-Independent method.

The *Resampling-Independent* algorithm is defined in table 1.

To compare the three methods, Resampling-Independent, Wilson-Dependent, and Wilson-Independent, the methods were applied to the same large datasets that were used in the previous paper:

- *Ant*: Christensen and Oppacher's 27,755 runs [3] of the artificial ant on the Santa-Fe trail; panmictic population of 500; best estimate of the true computational effort 479,344 at generation 18; $P(18) = \frac{2421}{27755} = 0.0872$
- *Parity*: 3,400 runs of even-4-parity without ADFs [5, 6]; panmictic population of 16,000; best estimate of true computational effort 421,074 at generation 23; $P(23) = \frac{3349}{3400} = 0.985$
- *Symbreg*: Gagné's 1,000 runs of a symbolic regression problem ($x^4 + x^3 + x^2 + x$) [5]; panmictic population of 500; best estimate of true computational effort 33,299 at generation 12; $P(12) = \frac{593}{1000} = 0.593$
- *Multiplexor*: Gagné's 1,000 runs of the 11-multiplexor problem [5]; panmictic population of 4,000; best estimate of true computational effort 163,045 at generation 25; $P(25) = \frac{947}{1000} = 0.947$

For each dataset six simulated run sizes were used: 25, 50, 75, 100, 200, and 500 runs.

For each of the 72 combinations of confidence interval method, dataset, and simulated run size, 10,000 samples of the specified number of runs were randomly selected from the problem domain's large dataset. For each sample the

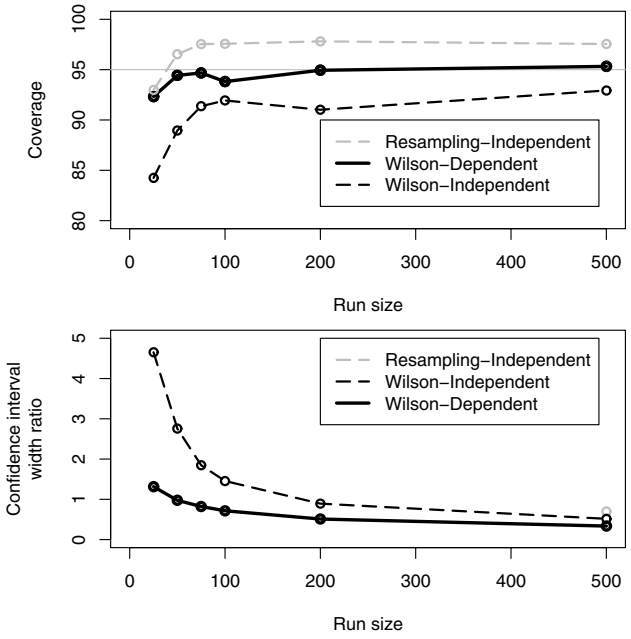

Figure 1: Average observed coverage (upper) and median confidence interval width ratios (lower) against run size for the three methods, averaged over the four problem domains.

95% confidence interval for the sample's computational effort was calculated using the specified method. Whether the confidence interval included the best estimate of the true computational effort (the *coverage*) was recorded, as was the width of the confidence interval relative to the best estimate of the true computational effort.

This process effectively simulates 10,000 genetic programming experiments over each of the four problem domains and each of the six run sizes, or a total of 240,000 experiments, for each of the three confidence interval methods.

The minimum computational effort, as calculated over the entire dataset, was used as the best estimate of the true computational effort for that dataset.

Figure 1 plots the results of these experiments[3]. The upper graph gives the observed coverage by run size (averaged over the four problem domains). The lower graph gives the median observed confidence interval width as a ratio of (i.e. divided by) the best estimate of the true computational effort (averaged over the four problem domains). Because Resampling-Independent produced an infinite-width median coverage for all bar one of the Ant experiments, only one data point for that method is plotted on the lower graph. 95% confidence intervals for the coverage results (upper graph) are smaller than ± 0.4 percentage points.

Of the three methods considered, Wilson-Dependent produces observed average coverage levels that are closest to the target 95%. This alone would make it the preferred choice, however it has two other advantages. The first advantage is that the width of the confidence intervals are notably tighter; and this is most obvious at the lower run sizes—sizes that are most commonly used. If the data is averaged over problem domain, then Resampling's infinite-width issue is confined to just the Ant data, and in this form the Resampling method produced widths that were at least 67% larger than those produced with Wilson-Dependent. Wilson-Independent produced widths that were at least 51% larger than Wilson-Dependent. The second advantage is that confidence intervals produced by the Wilson method were significantly less computationally expensive to obtain than those produced via the Resampling method. We conclude that Wilson-Dependent is the method of choice, and is the method used throughout the rest of this paper.

Finally, we have observed a correlation between the estimated value of j and the estimated value of $R(P(j))$. This correlation appears to be a critical component in explaining the enhanced coverage accuracy of Wilson-Dependent versus Wilson-Independent, but this remains an open issue.

3. VARYING ALPHA VALUES

In our previous research, the target coverage level was set at 95%. This section extends our earlier work by considering the effect on observed coverage levels when the target coverage level is varied.

Six commonly used target coverage levels (also known as $1 - \alpha$ values) were selected. They were: 80%, 85%, 90%, 95%, 99% and 99.9%. The previous experimental setup was repeated for each target level, that is: four problem domains were used (Ant, Parity, Symbreg, and Multiplexor) and six runs sizes were used (25, 50, 75, 100, 200, and 500 runs).

For each of the 144 combinations of target coverage level, problem domain, and run size, 10,000 samples of the specified number of runs were randomly selected from the problem domain's large dataset. For each sample the confidence interval for the sample's computational effort was calculated using Wilson's method (with a $1 - \alpha$ value as specified by the target coverage level). Whether the confidence interval included the true computational effort (the coverage) was recorded.

The cost of increasing the coverage level is an increase in the width of the confidence intervals. To assess the impact on the confidence interval width, we also recorded the width of the interval as a ratio of the true computational effort, for every sample's confidence interval.

Figure 2 plots the mean observed coverage and the ratio of the confidence interval width for each target coverage level (averaged over the four problem domains and six run sizes). Tables 2 and 3 give coverage and width statistics by problem domain and target coverage level (averaged over the six run sizes). Results where the sample did not include any successful runs could not produce a valid confidence interval; such samples were ignored for the calculation of the averages.

As can be seen from the upper plot of figure 2, the observed coverage levels are very close to the target levels up to about 95%. For the two higher cases of 99% and 99.9% the observed coverage is slightly smaller than the target. Although these results are highly statistically significant (with 95% confidence intervals of less than ± 0.2 percentage points), one could ask if the four datasets that were selected are a fair representation of the problem domains on which this method may be used. This question will always remain open, even though the domains selected are common GP problems. However, at the very least the

[3]Complete results for all the experiments in this paper are available from `www.massey.ac.nz/~mgwalker/CompEffort`.

Target coverage	80%	85%	90%	95%	99%	99.9%
Ant	77.5%	82.7%	87.6%	92.4%	97.5%	99.1%
Multiplexor	79.2%	84.6%	90.7%	95.7%	97.9%	98.2%
Parity	85.2%	89.7%	91.9%	94.0%	95.1%	95.5%
Symbreg	79.1%	84.7%	90.0%	94.9%	99.0%	99.9%

Table 2: Coverage statistics by target coverage and problem domain

Target coverage	80%	85%	90%	95%	99%	99.9%
Ant	1.12	1.28	1.51	1.89	2.71	3.80
Multiplexor	0.37	0.42	0.49	0.60	0.82	1.12
Parity	0.37	0.43	0.51	0.63	0.90	1.24
Symbreg	0.38	0.42	0.49	0.59	0.79	1.04

Table 3: Confidence interval width ratios by target coverage and problem domain

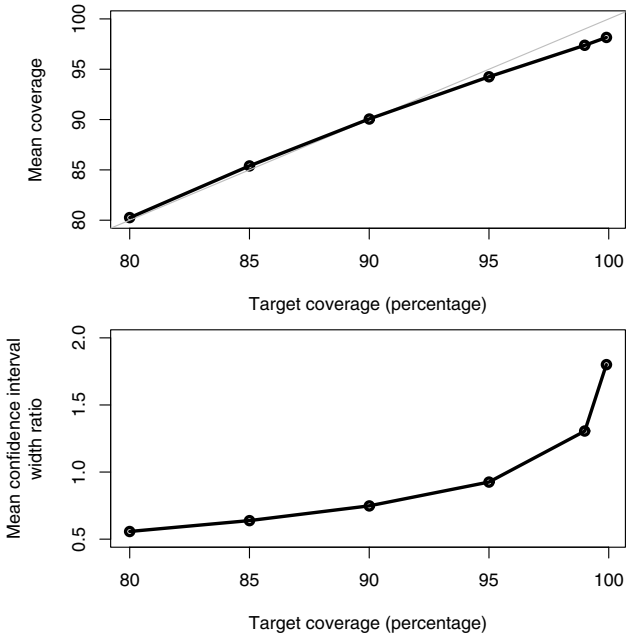

Figure 2: Observed coverage (upper) and confidence interval width ratios (lower) of Wilson's method against target coverage, averaged over all four problem domains and all six run sizes.

results from the four problem domains do give a good indication that Wilson's method performs as expected with different target coverage levels between 80% and 95%.

The lower plot of figure 2 shows, as expected, that as the target coverage is decreased, the width of the confidence interval also decreases. However even with a choice of 80% coverage, the mean observed width was not small: it was 56% of the true computational effort (with a range of 16% to just over two-fold). The Ant dataset has larger width ratios than the other datasets (due primarily to its low $P(j)$ value). But even with the Ant dataset removed, 80% coverage still gives a width ratio of 37%.

4. LARGE MINIMUM GENERATIONS

In our previous work the confidence interval generation methods were tested on a range of minimum generations that was quite tight: 12 to 25 generations. It was an open question as to whether any of the methods would continue to function acceptably if the true minimum generation were significantly outside this range.

For "difficult problems", Luke [7] concluded that a solution was more likely to be found with longer runs than with multiple shorter runs. This would produce a minimum generation that was much larger than would be obtained were the traditional approach, of terminating runs longer than 50 generations, taken.

To ensure that Wilson's method does not deteriorate with larger minimum generations, it is important to empirically check its validity in this area. Unfortunately it is far too computationally expensive to obtain thousands of runs on problem domains that find solutions only after many hundreds of generations. Instead we have elected to simulate GP experiments of that difficulty and to check Wilson's method using this simulated data.

4.1 Simulating GP experiments

4.1.1 Distribution selection

The distribution which models the generation at which a GP solution will appear is not known. In fact, it is highly likely that the distribution is problem-domain specific. However from the four large datasets that we have studied, it can be said that the distribution is a smooth curve that peaks at a specific generation and that may have a long tail to the right.

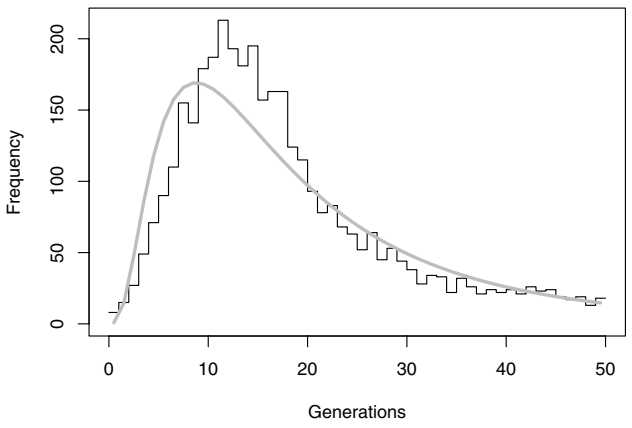

Figure 3: The best model (thick grey line) that was found for the Ant dataset (black line): a log-normal distribution with $\mu = 16.2$ and $\sigma = 2.2$.

Many such distributions have been defined that pass this description. In order to assess the match between these distributions and the large datasets, we optimised the distributions' parameters and then tested their quality as models.

The distributions we considered were: normal, log-normal, gamma, weibull, and beta. Each distribution takes two parameters to describe its shape. The optimal values for these parameters were found with a numerical method[4]. Figure 3 shows one of the better models. Once the distributions' parameters were obtained, the models were then compared to the real dataset using χ^2 tests.

To quantify how well the distribution modelled the real data, χ^2 tests were executed for multiple run sizes (25, 50, 100, 200, 300, 400, and 500 runs). For each run size, 100 samples of that number of runs were randomly generated using the distribution (with its optimised parameters). For each of the 100 samples a χ^2 test was executed (on bin sizes of one generation, unless the real dataset contained fewer than 5 successes in a given bin, in which case adjacent bins were combined until the enlarged bin contained at least 5 successes). If the p-value of the χ^2 test was greater than 0.05 then it was considered that the distribution successfully modelled the dataset. The proportion of times that the distribution sufficiently modelled the data was recorded and those proportions are plotted for the log-normal and normal distributions in figure 4.

Although sample sizes of 100 give a 95% confidence interval for the proportions of up to ± 0.1, the results are sufficient to conclude that the log-normal distribution models the results of genetic programming runs with an accuracy that is, at best, only mediocre. Of perhaps more interest is the normal distribution which performs very well with the Parity problem, but was unable to satisfy any tests for the Symbreg problem.

Because we were unable to find a distribution function that successfully modelled the real datasets beyond 100 runs, we elected to use the best two distributions (normal and log-normal) for simulating GP runs with large minimum generations.

[4]An implementation of the Nelder and Mead method was used as defined by the optim function in the statistical software R [9].

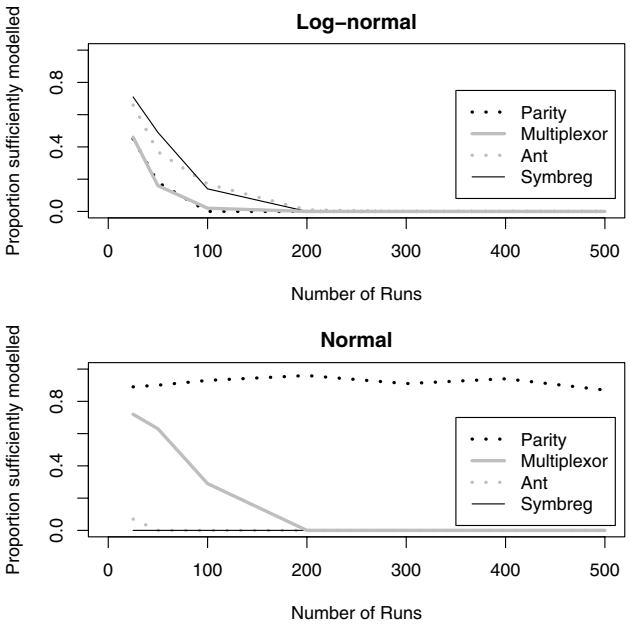

Figure 4: The performance of the log-normal (upper) and normal (lower) distributions as models for the four large datasets.

4.1.2 *Probability of success*

When considering the cumulative probability curves of the large datasets (most notably Symbreg and Ant), it appeared that GP runs may asymptotically tend to a cumulative probability of success that is less than one. Although it seems reasonable that if a GP run were left to evolve indefinitely it would eventually find a solution, it would seem that the tails must become *very* long indeed.

To model this asymptotic behaviour, a "second level" was added. This asked the question, "does a given run have any chance of success?". If the answer was "yes", then the chosen distribution was used to answer the question "does success occur before the cut-off generation?".

4.2 Testing performance on simulated data

Because neither the normal nor the log-normal distributions had any consistent acceptable level of success past 100 runs, we elected to limit the use of the models to 25, 50, and 100 runs.

Because appropriate parameters for each model are unknown a range of parameters were used. For both models the mean was set to 25, 100, 500, and 1000 generations. For the log-normal distribution the standard deviation was set to 0.5, 1.0, and 2.0 times the mean. For the normal distribution the standard deviation was set to $\frac{1}{16}$, $\frac{1}{8}$, and $\frac{1}{4}$ times the mean. The probability of success (at the second level) was set to 0.2, 0.5, and 0.8. The cut-off was set to 1,000 generations. The number 500 was used as a population size, but this was just a scaling factor that had no bearing on the model nor the coverage results.

It was found that when the standard deviation values used

Mean	25	100	500	1000
	93.6%	91.5%	90.6%	91.1%
Std. dev.	0.5	1.0	2.0	
	93.6%	91.7%	89.9%	
Success prop.	0.2	0.5	0.8	
	90.1%	92.2%	92.7%	
Runs	25	50	100	
	90.2%	92.0%	93.0%	

Table 4: Average coverage statistics for the log-normal model per parameter

Mean	25	100	500	1000
	94.8%	94.7%	94.8%	94.4%
Std. dev.	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	
	94.9%	94.8%	94.3%	
Success prop.	0.2	0.5	0.8	
	94.1%	94.9%	94.9%	
Runs	25	50	100	
	94.4%	94.9%	94.7%	

Table 5: Average coverage statistics for the normal model per parameter

for the log-normal distribution were applied to the normal distribution, they were sufficiently large to produce a non-zero probability of success at the initial generation. This non-zero probability was sufficient to set the minimum generation to generation zero. Koza studied the probability of finding a solution at generation zero for both the 11-multiplexor and 6-multiplexor problems [5, page 207]. He tested up to 10,000,000 individuals, and found that none were successful; that is a probability of success of less than 0.000001. Because this work was intended to model problems significantly harder than the two Koza studied, the standard deviations used for the normal distribution were reduced.

For each of the 216 combinations of distribution, mean, standard deviation, probability of success, and number of runs, one million simulated runs were generated using the specified distribution. From these simulated runs, 10,000 samples of the specified number of runs were randomly selected. For each sample the 95% confidence interval for the sample's computational effort was calculated using Wilson's method. Whether the confidence interval included the true computational effort (the coverage) was recorded.

The true computational effort was obtained by calculating

$$\min_i (i+1) \times R(P(i) \times p) \times M \tag{6}$$

where: R is the function for calculating the number of runs required (equation 1); $P(i)$ is the cumulative proportion given by the distribution function; p is the probability of success (at the second level); i is the generation which ranged from 0 to 1000; and M is the population size (where 500 was used).

Tables 4 and 5 summarise these results. The log-normal model had an average coverage of 91.7% and the normal model had an average of 94.6%. These should be compared with the desired coverage of 95%.

So, for the log-normal model (table 4), The average coverage for a mean of 25 generations (as seen in the top left cell) was 93.6%. This coverage is an average coverage for all parameter combinations where the mean was set to 25. Similarly, for the normal model (table 5), for all parameter combinations where run size was set to 100 runs, an average of 94.8% of the samples produced a confidence interval that included the true computational effort.

For the log-normal distribution, the minimum generation ranged from 7 to 1000 generations with a mean of 299 and an upper quartile of 523 generations. For the normal distribution the minimum generation ranged from 28 to 1000 generations with a mean of 447 and an upper quartile of 799 generations. Thus the minimum generations that were considered in this study were significantly larger than those observed in the experiments originally executed.

Both models showed reduced coverage as the standard deviation increased. Both models produced increased coverage levels as the success proportion increased, and increased coverage as the number of runs increased.

From these results, if your GP data follows a normal or log-normal distribution, it appears that the confidence interval generation method based on Wilson's method produces coverage levels that are a good approximation to a 95% confidence interval.

4.3 Arbitrary distributions

Given the success with the normal and log-normal distributions, we continued the investigation with some arbitrarily selected distributions. The objective in this study was to see if the earlier success depended on the selection of the distribution functions.

Figure 5 shows the four new distributions that were selected. All four distributions had zero-values between 0–49 generations and 951–1000 generations and had a cumulative probability (the area under the graph) of one. The first was a rectangular shape. The second was a triangular shaped distribution that sloped from 0 at 50 generations to a peak at 950 generations (termed Right-Triangle). The third was a triangular shape that sloped from a peak at 50 generations down to 0 at 950 generations (termed Left-Triangle). The fourth was a semi-ellipse.

To test the coverage of Wilson's method on these distributions, three variables were required: the distribution, the success proportion (at the second level), and the number of runs. The success proportion was given the same three values as before: 0.2, 0.5, and 0.8. However, the number of runs in each sample was extended to include 25, 50, 100, 200, and 500 runs.

For each of the 60 combinations of distribution, success proportion, and runs size, 100,000 simulated runs were generated using the specified distribution. From these simulated runs, 1,000 samples of the specified number of runs were randomly selected. For each sample the 95% confidence interval for the sample's computational effort was calculated using Wilson's method. Whether the confidence interval included the true computational effort (the coverage) was recorded.

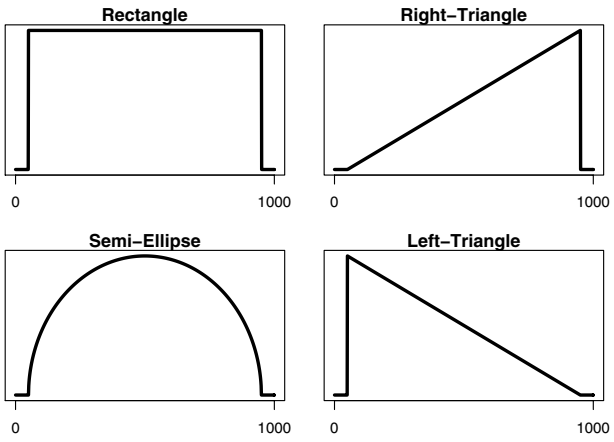

Figure 5: Density versus generation for the four arbitrarily-selected distributions.

Success prop.	0.2	0.5	0.8		
	92.7%	95.1%	95.3%		
Runs	25	50	100	200	500
	93.2%	94.4%	94.5%	94.9%	94.8%

Table 6: Average coverage statistics for the Rectangle model per parameter

For Rectangle and Right-Triangle, the minimum generation was 951. Left-Triangle's minimum generation ranged from 341 to 692 generations and Semi-Ellipse's ranged from 817 to 928 generations—with the specific value dependent on the success proportion.

Tables 6, 7, 8 and 9 show the coverage results for the four distributions. Wilson's method produced an average coverage of 94.4% for Rectangle, 94.9% for Right-Triangle, 91.1% for Left-Triangle, and 94.0% for Semi-Ellipse.

Although the average coverage for Left-Triangle dips to 91.1%, this should be compared to Robert Newcombe's analysis of the performance of the normal-approximation method for confidence intervals for a proportion [8]. He showed that method to have an estimated mean coverage of 88%. In that light, Wilson's method on the Left-Triangle data performs better than that generally-accepted and widely used method.

These results are very interesting. They show that the performance of Wilson's method is not affected by the distribution of successes—even with these four far-from-typical distributions. The results also give further evidence that the method is not sensitive to the magnitude of the minimum generation.

Success prop.	0.2	0.5	0.8		
	94.9%	95.3%	94.6%		
Runs	25	50	100	200	500
	95.3%	95.3%	94.0%	95.1%	95.0%

Table 7: Average coverage statistics for the Right-Triangle model per parameter

Success prop.	0.2	0.5	0.8		
	89.7%	90.7%	92.8%		
Runs	25	50	100	200	500
	88.1%	89.9%	91.3%	92.5%	93.7%

Table 8: Average coverage statistics for the Left-Triangle model per parameter

Success prop.	0.2	0.5	0.8		
	92.5%	94.0%	95.4%		
Runs	25	50	100	200	500
	93.7%	93.4%	93.4%	94.3%	95.1%

Table 9: Average coverage statistics for the Semi-Ellipse model per parameter

5. MORE LARGE DATASETS

Steffen Christensen executed some enormously large number of runs on the Ant problem domain [2]. This section analyses the performance of the confidence intervals for these real datasets.

The four datasets were all based on the artificial ant on the Santa-Fe trail, a problem domain detailed in *Genetic Programming* [5, section 3.3.2] and commonly used as a benchmark for variations of GP. The four datasets that Christensen produced vary by population size and the generation at which the runs were cut off. The four datasets are:

- *Ant m10000g25*: Panmictic population of 10,000; cutoff of 25 generations; 12,280 runs; best estimate of the true computational effort 478,506 at generation 15; $P(15) = \frac{9,647}{12,280} = 0.786$
- *Ant m1000g150*: Panmictic population of 1,000; cutoff of 150 generations; 40,010 runs; best estimate of the true computational effort 446,801 at generation 17; $P(17) = \frac{6,775}{40,010} = 0.169$
- *Ant m250g60*: Panmictic population of 250; cutoff of 60 generations; 400,625 runs(!); best estimate of the true computational effort 488,518 at generation 19; $P(19) = \frac{18,445}{400,625} = 0.0460$
- *Ant m250g1000*: Panmictic population of 250; cutoff of 1,000 generations; 8,000 runs; best estimate of the true computational effort 503,594 at generation 20; $P(20) = \frac{355}{8,000} = 0.0444$

These datasets are interesting because, whereas our previous research covered $P(j)$ values that were biased towards one, these datasets are biased towards a cumulative success at the minimum generation of zero. To enable direct comparison with the earlier results, run sizes were simulated at 25, 50, 75, 100, 200, and 500 runs.

For each of the 24 combinations of dataset and run size, 10,000 samples of the specified number of runs were randomly selected from the specified large dataset. For each

Run size	25	50	75	100	200	500	Average
m10000g25	95.9%	95.4%	95.3%	94.7%	95.0%	94.9%	95.2%
m1000g150	92.4%	94.4%	94.0%	94.8%	95.2%	94.7%	94.2%
m250g60	74.9%	89.4%	90.1%	92.4%	93.5%	93.8%	89.0%
m250g1000	78.4%	88.8%	91.8%	92.1%	93.3%	94.6%	89.8%
Average	85.4%	92.0%	92.8%	93.5%	94.3%	94.5%	92.1%

Table 10: Coverage statistics, by run size and dataset, for the four extra Ant datasets.

sample the 95% confidence interval for the sample's computational effort was calculated using Wilson's method. We recorded whether the confidence interval included the best estimate of the true computational effort (the coverage). The best estimate of the true computational effort was the computational effort calculated over the entire dataset.

Table 10 shows the results of these experiments. 95% confidence intervals for these results are at most ±1 percentage point, and in most cases will be no more than ±0.5 percentage points.

The performance of Wilson's method is very good, averaging 95.2% and 94.2% coverage, for the two datasets with the higher values for $P(j)$ (m10000g25 and m1000g150). For the other two datasets (m250g60 and m250g1000), the performance is good except for the smaller run sizes (specifically 25 runs).

However, for the smaller run sizes, it is worth noting that the granularity of the estimate of $P(j)$ is of the same order as the best estimate of the true value. In other words, for the case where 25 runs are being sampled, we should expect just one of the runs to succeed ($\frac{1}{25} = 0.04$ and the two values for $P(j)$ were 0.0460 and 0.0444). Thus, a variation in success of just one run is a variation of approximately ±100% of the true value.

It is also worth noting that, although the only difference between m250g60 and m250g1000 was that the former had a shorter cutoff value (a variable that would not have affected the true computational effort), there was still a 3.1% difference between their estimated computational efforts—and that was with an enormous number of runs.

So, as might be expected, if the cumulative probability of success at the minimum generation is very small compared to the granularity produced by the number of runs, the coverage of Wilson's method deteriorates.

However, the general picture provided by these datasets is that the method to produce confidence intervals for minimum computational effort is reliable.

6. CONCLUSIONS

This research has extended our previous work on the production of confidence intervals for Koza's minimum computational effort statistic. We have shown that the confidence interval production method is reliable; specifically that:

- It out-performs other methods, in terms of both appropriate coverage levels and tighter confidence interval widths.
- It performs well at different target coverage levels, especially those between 80% and 95%.
- It performs well across a large range (10 to 1,000 generations) of minimum generations on simulated datasets.
- It appears to be insensitive to the distribution of generations of successful runs.
- It performs well on a number of datasets collected from real GP runs.

7. ACKNOWLEDGEMENTS

We would like to acknowledge Steffen Christensen for giving us access to his enormous Ant datasets. Our thanks also go to Christian Gagné for the use of his Symbreg and Parity datasets.

8. REFERENCES

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

[2] S. Christensen. *Towards Scalable Genetic Programming*. PhD thesis, Ottawa-Carleton Institute for Computer Science, Ottawa, Canada, 2007.

[3] S. Christensen and F. Oppacher. An analysis of Koza's computational effort statistic for genetic programming. In *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 182–191. Springer-Verlag, 2002.

[4] M. Keijzer, V. Babovic, C. Ryan, M. O'Neill, and M. Cattolico. Adaptive logic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 42–49. Morgan Kaufmann, 2001.

[5] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[6] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge Massachusetts, 1994.

[7] S. Luke. When short runs beat long runs. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 74–80. Morgan Kaufmann, 2001.

[8] R. G. Newcombe. Two-sided confidence intervals for the single proportion: comparison of seven methods. *Statistics in Medicine*, 17:857–872, 1998.

[9] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006.

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