More on Computational Effort Statistics for Genetic Programming

Jens Niehaus and Wolfgang Banzhaf

System Analysis
Computer Science Department
University of Dortmund
D-44221 Dortmund, Germany
{jens.niehaus,wolfgang.banzhaf}@cs.uni-dortmund.de

Abstract. In this contribution we take a look at the *computational effort* statistics as described by Koza. We transfer the notion from generational genetic programming to tournament-selection (steady-state) GP and show why, in both cases, the measured value of the *effort* often differs from its theoretical counterpart. It is discussed how systematic estimation errors are introduced by a low number of experiments. Two reasons examined are the number of unsuccessful experiments and the variation in the number of fitness evaluations necessary to find a solution among the successful experiments.

1 Introduction

Although more and more work is done examining the theory of *genetic programming* (GP) most of the publications use an empirical approach to rate new findings and modifications of traditional GP. For comparison purposes different kinds of statistics are needed. One of those used traditionally is the *computational effort* statistics as presented in [4]. Lately, however, there were several publications which took a closer look at this measure [3,2,5] and came up with several problems regarding the accuracy of the empirically measured values. In this contribution we show how *computational effort* statistics can be used in conjunction with steady-state algorithms instead of generational GP (section 3). With such an approach it is possible to reduce the difference between a theoretical *effort* value and the measured one. We show further that other inaccuracies are still remaining. They relate to the number of unsuccessful experiments (section 5) and large differences in the number of fitness evaluations needed over several experiments (section 6).

2 Measurement and Calculation of the Computational Effort

In [4] Koza describes a method to compare the results of different evolutionary methods, e.g. different modifications of GP. The so called *computational effort*

C. Ryan et al. (Eds.): EuroGP 2003, LNCS 2610, pp. 164–172, 2003.

is calculated as the number of fitness evaluations needed to find a solution of a problem with a probability of success z of at least $z = 99\%$.

The number of fitness evaluations GP needs to solve a problem can differ a lot. Some experiments might find a solution very fast while others need many more fitness evaluations – still others won't find a solution at all in the given amount of time / evaluations. Calculating the *computational effort* for a problem is based on empirical data. We have to use relative frequencies instead of probabilities for finding the solution after a certain number of fitness evaluations. For calculating the *computational effort* $I(M, z)$ Koza defines the following equation:

$$I(M, z) = \min_i Mi \left\lceil \frac{\ln(1 - z)}{\ln(1 - P(M, i))} \right\rceil \tag{1}$$

In this formula M stands for the number of individuals in the population and i is the number of generations. Thus Mi represents the number of fitness evaluations calculated in one experiment. When i generations have passed without a solution found the run has to be restarted with a different initial population. The value $P(M, i)$ represents the estimated probability for finding a solution within $M \times i$ fitness evaluations and is calculated using the results of a certain number of experiments for the examined problem. The value z represents the confidence level of finding a solution and will be set to 0.99 throughout this work.

In [2] Christensen and Oppacher show that values calculated using equation (1) differ up to 25% from the theoretical *computational effort*. Among other things they show the influence of the ceiling and minimum operators, which both cause deviation of the results.

3 Computational Effort for Steady-State Algorithms

After a number of *runs* experiments with Koza's generational GP-system the probabilities $P(M, i)$ are calculated as relative frequencies $k/runs$ with k being the number of runs in which a solution was found within the first i generations. The values for $P(M, i)$ are always calculated after $M \times i$ fitness evaluations. Thus it has no influence whether the solution is always found near the beginning of a certain generation or near the end[1]. This is comparable to calculating the mean length of several lines in centimeters while each line is given in meters. The bigger a population is the more inaccurate the values of $P(M, i)$ become. Using a steady-state approach (with tournament selection) [1] the relative frequencies refer more precisely to a certain number of fitness evaluations. For the rest of this paper we use the steady-state approach with the additional option to stop an experiment after any fitness evaluation. In this way we can eliminate the ceiling operator of equation (1) and concentrate on other problems regarding the use of the *computational effort* statistics.

To calculate the number of required fitness evaluations we use the following equation:

$$1 - (1 - P(eval))^{\frac{effort}{eval}} \geq 0.99 \tag{2}$$

[1] Assuming the evaluations are executed sequentially.

The value *effort* represents the number of evaluations we are looking for. After *eval* evaluations a run is stopped and a new independent one with a new population is started with the remaining number of fitness calculations. $P(eval)$ is the probability that a solution is found after at most *eval* evaluations. As this probability is unknown we have to replace it with the estimates $\hat{P}(eval)$ that are based on empirical data collected from a number of experiments carried out previously. More on the difference between P and $\hat{P}$ can be found in [2].

Transformation of equation 2 leads to:

$$effort = \min_{eval} eval \frac{\ln 0.01}{\ln\left(1 - \hat{P}\left(eval\right)\right)} \tag{3}$$

The differences to equation (1) are (i) the transition from generational to evaluation-based calculation and (ii) the loss of the ceiling operator.

4 Experimental Framework

In [2] some inaccuracies of the *computational effort* statistics were pointed out. This section offers two more aspects that can lead to differences between the calculated *computational effort* and the theoretical value. First, the influence of the number of runs that do not find a solution is examined and second, we look at the influence of the distribution of the number of fitness evaluations needed to find a solution.

For a better illustration we use a group of hypothetical GP problems that will not find a solution with a probability of P_{fail} [2]. The number of fitness evaluations needed to find a solution in the remaining runs follows a $(100,000, sd)$-normal distribution. As we are only interested in positive integer values for the number of evaluations we use the floor operator and discard all negative values. The assumption of a normal distribution does not hold for most of the problems GP is used with, but even with this well known distribution problems appear that have a large influence on the calculated *computational effort*.

To show the influence of varying values for P_{fail} and sd we chose

$$P_{fail} \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$$

and

$$sd \in \{1,000, 2,000, 5,000, 10,000, 11,000, 20,000, 30,000\}.$$

For each combination we calulated the theoretical *computational effort* using equation (3) and the normal distibution's density function:

$$P(eval) = (1 - P_{fail}) \frac{1}{sd\sqrt{2\pi}} \int_0^{eval} \exp \frac{-(x - 100,000)^2}{2sd^2} dx \tag{4}$$

The results are shown in Figure 1.

[2] With *solution* we do not necessarily mean *perfect solution*. We just count the number of runs that produce a solution that we can accept as *good enough* (see [5]).

The lowest *computational effort* has the combination ($P_{fail} = 0.2, sd = 1,000$) with 295,701 evaluations, the biggest one belongs to ($P_{fail} = 0.8, sd = 30,000$) with 3,214,423 evaluations – ten times as many. The influence of P_{fail} is much higher than the one of the standard derivation. Looking at equation (3) this is quite obvious. P_{fail} directly influences the value of $(1 - \hat{P}(eval))$. For values near 1 this has a large influence on the whole fraction, while a change in sd in most cases only influences the linear term of $eval$. As an example we take a look at all *effort* values with $P_{fail} = 0.2$. With changing values of sd (between 1,000 and 30,000), the number of evaluations $eval$ at which the minimum occurs changes from 103,041 to 153,192. At the same time, $\hat{P}(eval)$ remains between 0.769 and 0.799 resulting in rather similar *effort* values.

5 Influence of P_{fail} on the Computational Effort

With Figure 1 it already becomes apparent that different values of P_{fail} can have a large influence on the *computational effort*. This will result in problems when the number of independent experiments is very small, for example 50.

A small number of experiments can lead to estimated $\hat{P}_{fail}$ values differing from the original value P_{fail}. In Figure 1 this is comparable with a shift of the *computational effort* along the P_{fail}-axis.

For an empirical quantitative analysis of this behavior we used all hypothetical problems with $sd = 1,000$ and for each value of P_{fail} we created 500 experiments at random using the probability P_{fail} to decide whether an experiment finds a solution or not and for all successful experiments we randomly chose a number of evaluations needed to find the solution based on a $(100,000, sd)$-normal distribution. We used the floor operator to transform the values to integers and repeated an experiment if the picked number was negative[3].

For each value of P_{fail} we calculated the empirical *computational effort* $\widehat{effort}$ corresponding to these 500 experiments. On top of this, the $\widehat{effort}$-calculation was repeated 1,000 times with 500 different experiments. Executing 500 experiments at random means that the proportion of unsuccessful experiments $\hat{P}_{fail}$ differs from the exact value P_{fail}.

The results of these experiments are presented in Table 1. The second column contains the theoretical *computational effort*. The third column holds the mean value of the 1,000 new $\widehat{effort}$ values each calculated with 500 experiments. The percentage relative to the original *effort* value is also shown. The fourth column contains the percentage of the smallest $\widehat{effort}$ value relative to the original *effort*, the fifth column shows the highest.

As expected the mean of all new $\widehat{effort}$ values is similar to the theoretical *computational effort*. On the other hand, the last two columns indicate that some

[3] Calculating the *computational effort* with integers based on a normal distribution and the *floor*-function gives slightly smaller values than expected theoretically. In our examples these differences are always smaller than 0.001 percent and thus will be ignored throughout this document.

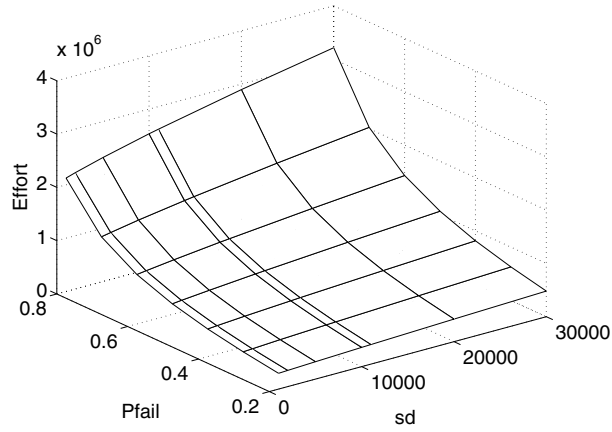

Fig. 1. The *computational effort* for the 49 hypothetical optimization problems

Table 1. Executing 500 runs with a calculated empirical *computational effort* of $\widehat{effort}$ and repeating this experiment a thousand times leads to a mean $\widehat{effort}$ near the theoretical value with a good chance of big deviation in single cases

P_{fail}	*effort*	$\sum \widehat{effort}\,/1,000$	min	max
0.2	295,701	294,566 (99.6 %)	85.1 %	119.6 %
0.3	394,995	393,505 (99.6 %)	83.5 %	121.1 %
0.4	518,746	515,964 (99.5 %)	81.3 %	117.2 %
0.5	685,480	683,113 (99.7 %)	80.9 %	120.5 %
0.6	929,849	927,447 (99.7 %)	77.7 %	123.6 %
0.7	1,331,376	1,330,270 (99.9 %)	71.8 %	126.0 %
0.8	2,127,615	2,128,350 (100.0 %)	72.2 %	136.6 %

series of experiments lead to empirical *computational efforts* that underestimate the true effort by 28% or overestimate it by up to 36% (both values for $P_{fail} = 0.8$). The higher P_{fail}, i.e. the more difficult a problem is, the larger a difference $effort - \widehat{effort}$ may result.

When decreasing the number of experiments the result becomes even more obvious. Instead of calculating an empirical *computational effort* $\widehat{effort}$ with 500 experiments we repeated the series with 200, 100 and 50 runs. Again, each of those series was repeated 1,000 times. The results are presented in Figure 2. The z-axis of the left diagram represents the differences of the percentage values of the smallest and the highest $\widehat{effort}$ in relationship to the original *effort*. For example, if the theoretical *computational effort* was 10,000 and the lowest and highest $\widehat{effort}$ values of the 1,000 series with 500 experiments were 9,000 and 10500, respectively, the percentage values (columns four and five in table 1) would be 90% and 105%. This would result in a difference of 105-90=15. Thus the higher the z value is, the larger possible mistakes in the empirically calculated *computational effort* might become.

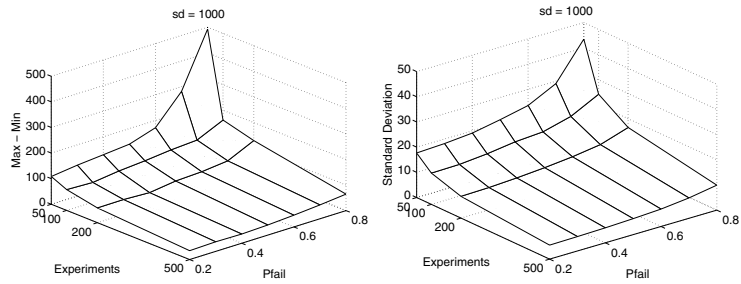

Fig. 2. Deviations (in percent compared to *effort*) between the lowest and highest $\widehat{effort}$ value

Representing 1,000 new empirical *computational effort* values $\widehat{effort}$ as percentage values in correspondence to the original *effort*, the right diagram in Figure 2 shows the standard derivations of these values.

For the series with 500 experiments the values of both diagrams are quite low. This means that a number of 500 experiments will result in an empirically calculated *computational effort* not too different from the theoretical value. The smaller the number of experiments becomes, the larger a difference we see.

Using only 50 experiments and the parameter $P_{fail} = 0.8$ one of the test series resulted in an empirical *computational effort* of 11,345,515, which is more than 500 percent of the theoretical value. $P_{fail} = 0.8$ corresponds to the fact that only 10 out of 50 runs should lead to a solution. In this special case only two of the 50 runs solved the problem, which drastically lowered the denominator of equation (3) leading to a high $\widehat{effort}$ value.

Up to now we only varied P_{fail} leaving sd at a constant value of 1,000. Nevertheless, all series of experiments were repeated using values of sd = 2,000, 5,000, 10,000, 11,000, 20,000 and 30,000. The results show no significant difference to those for $sd = 1,000$. Again, for each number of runs 1,000 tests were performed. For some (number of runs/P_{fail}/sd) combinations there are differences in the maximum/minimum value to those shown in the left graph of Figure 2, but the differences to the standard deviation of the 1,000 newly computed $\widehat{effort}$ values for $sd = 1,000$ – as seen in the right graph of Figure 2 – is always insignificantly small. The main cause for differences between empirically calculated $\widehat{effort}$ values and theoretical values stems from the differences between P_{fail} and the empirical estimate $\widehat{P}_{fail}$ for this value.

6 Influence of *sd* on the Computational Effort

The previous section discussed the influence of the difference between the probability of success for finding a solution within one run $(1 - P_{fail})$ and its estimate $(1 - \hat{P}_{fail})$, calculated with a certain number of runs on the difference between the *computational effort* and an $\widehat{effort}$ value calculated using those runs.

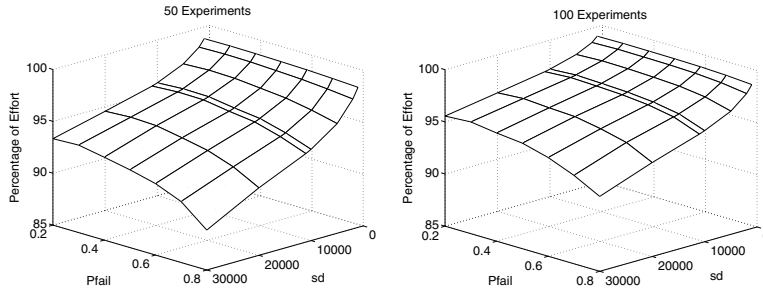

Fig. 3. The average percentage of the $\widehat{effort}$ values calculated on the base of 50 and 100 experiments in relationship to the theoretical *computational effort*

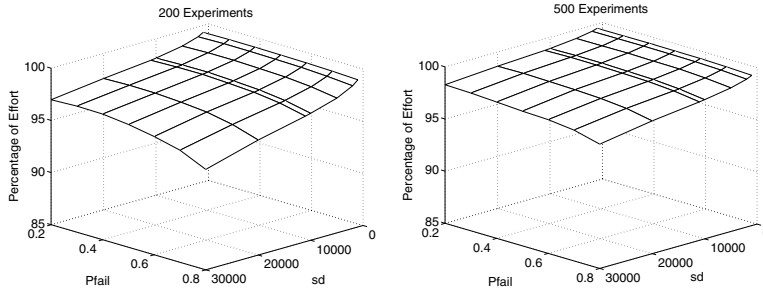

Fig. 4. The average percentage of the $\widehat{effort}$ values calculated on the base of 200 and 500 experiments in relationship to the theoretical *computational effort*

In this section we show that the values of *effort* and $\widehat{effort}$ may even vary if the estimates for P_{fail} are chosen to be identical to the theoretical probability.

We need to take a look at the distribution of the number of fitness evaluations needed to find a solution in successful runs. As written before, for each $(P_{fail}/sd/runs)$-combination ($runs \in \{50, 100, 200, 500\}$) we did *runs* experiments of which now exactly $P_{fail} \times runs$ fail to find a solution. The number of fitness evaluations needed in sucessful ones is calculated as before (see section 5). This procedure guarantees that P_{fail} equals $\hat{P}_{fail}$.

For each combination $(P_{fail}, sd, runs)$ we performed a series of 1,000 experiments, calculated $\widehat{effort}$ for each of the experiments and finally calculated the mean of all $\widehat{effort}$ values for each series.

Figure 3 and 4 show the results of these experiments. Each value of *runs* has its own graph. The x- and y-axis represent the different values of *sd* and P_{fail} while the z-axis represents the corresponding mean $\widehat{effort}$ over the 1,000 series. This value is expressed as a percentage of the theoretical *computational effort*.

In all cases the *computational effort* is slightly underestimated. For larger *sd* and P_{fail} the inaccuracy becomes more apparent. For small values of *sd* the probability of P_{fail} has only little effect on the mean $\widehat{effort}$ value. For larger *sd* the influence of P_{fail} grows. The smaller the number of runs is the more the *computational effort* is underestimated.

The reason for the underestimation is as follows: With a discrete number of experiments the minimum of equation (3) is, in most cases, associated with the highest number of fitness calculations needed within the successful runs[4]. For this value of *eval* the denominator of the fraction of equation (3) equals $\ln(P_{fail})$. To reach the same denominator-value for the theoretical *computational effort* using equations (3) and (4) the required value of *eval* is higher in most cases and results in a higher *effort*. For smaller *eval* values the product *eval* $\times$ fraction becomes larger thus rendering this product unimportant for calculating the minimum in equation (3). The discete number of experiments means that $P(eval)$ is larger than the corresponding value of the continuous case when calculating the theoretical *computational effort*.

7 Summary

In this paper we showed how to use Koza's *computational effort* statistics with steady-state GP-systems using tournament selection. Although the correspondence of *effort* values between theory and experiment should be higher than for generational GP-systems there were still differences between both.

We showed two different factors that have an influence on the size of errors:

1. To calculate the *computational effort* $P(eval)$ is needed. This stands for the probability a solution is found within *eval* fitness evaluations. As this value is usually unknown it has to be estimated based on empirical data. Using relative frequencies leads, on average, to a *computational effort* similar to the theoretical value but, depending on the number of runs carried out to derive the empirical value, can differ a lot in some cases.
2. The distribution of numbers of evaluations needed to find a solution plays a smaller but nevertheless important role. As an example we used a normally distributed set and demonstrated that the calculated $\widehat{effort}$ underestimates the theoretical value by up to 12 percent (Fig. 3).

While the second issue was only an example for a very specific distribution and the effect will be different for other problems and other distributions, the first issue affects all calculations of the *computational effort*.

Sections 5 and 6 showed that the differences between the theoretical value of the *computational effort* and the calculated one increases when the number of runs decreases on which the relative frequencies for $\hat{P}(eval)$ are based. We showed that calculating *effort* based on only 50 experiments may lead to values quite off the theoretical values, and that even 200 experiments often are not sufficient.

Furthermore, the more experiments fail to find a solution and the higher the differences in the number of fitness evaluations needed to find a solution in different runs are, the larger the deviation of empirical *computational effort* might become. In such cases more experiments should be carried out.

[4] For the series with $runs = 50$ and $sd < 30,000$ there was on average less than one *eval* value higher than the one the minimum was realized with.

References

1. W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone: Genetic Programming: An Introduction. San Francisco, CA: Morgan Kaufmann, 1998
2. S. Christensen and F. Oppacher: An Analysis of Koza's Computational Effort Statistic for Genetic In: J. A. Foster, E. Lutton, J. Miller, C. Ryan, and A. G. B. Tettamanzi,(eds.), Proceedings of the 5th European Conference on Genetic Programming, EuroGP 2002, volume 2278 of LNCS, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag, pages 182–191
3. M. Keijzer, *et al*: Adaptive Logic Programming. In: Spector, L., *et al* (eds.): Proceedings of the 2001 Genetic and Evolutionary Computation Conference: GECCO 2001, Morgan Kaufmann, pages 42–49
4. J. R. Koza: Genetic Programming: On the Programming of Computers by Natural Selection. Cambridge, MA: MIT Press, 1992
5. S. Luke and L. Panait: Is the Perfect the Enemy of the Good? In: W. B. Langdon *et al.* (eds.), Proceedings of the 2002 Genetic and Evolutionary Computation Conference: GECCO 2002, Morgan Kaufman, pages 820–828