Statistical Distribution of Generation-to-Success in GP: Application to Model Accumulated Success Probability

David F. Barrero, Bonifacio Castaño [⋆], María D. R-Moreno, and David Camacho [⋆⋆]

Universidad de Alcalá, Departamento de Automática
Crta. Madrid-Barcelona, Alcalá de Henares, Madrid, Spain
david@aut.uah.es,bonifacio.castano@uah.es,
mdolores@uah.es,david.camacho@uam.es

Abstract. Many different metrics have been defined in Genetic Programming. Depending on the experiment requirements and objectives, a collection of measures are selected in order to achieve an understanding of the algorithm behaviour. One of the most common metrics is the accumulated success probability, which evaluates the probability of an algorithm to achieve a solution in a certain generation. We propose a model of accumulated success probability composed by two parts, a binomial distribution that models the total number of success, and a lognormal approximation to the generation-to-success, that models the variation of the success probability with the generation.

Keywords: Generation-to-success, success probability, measures, models, performance metrics

1 Introduction

Understanding the behaviour of Evolutionary Algorithms is far from being trivial. One consequence of this fact is the difficulty of selecting a set of properties able to explain what is happening in the algorithm. It is not easy to understand why the algorithm fails (or success), or which is the effect of changing a parameter of the algorithm. Actually, there is no a general consensus about what should be measured, and thus different studies use different measures.

Some authors, however, have tried to provide some clues about which metrics should be used in which cases [1], even a classification of measures has been proposed by Bartz-Beielstein [2]. He differentiates effectivity and efficiency measures. The former informs about whether or not the algorithm is able to find a solution while the latter measures how many resources are required to achieve it. Despite the lack of a general consensus about which measures should be used

[⋆] Bonifacio Castaño is with the Departamento de Matemáticas, Universidad de Alcalá.

[⋆⋆] David Camacho is with the Departamento de Informática, Universidad Autónoma de Madrid.

in the Genetic Programming (GP) experimentation, there are some widely used metrics, such as the success rate, the mean best fitness and the mean average fitness [1]. Metrics should be used with care since they represent different views of the object of study, and, as Luke observed with fitness and success rate [3], there may not be correlation between these measures.

Several metrics have been proposed to measure efficiency. In GP, Koza's computational effort [4] is a particularly popular measure. It estimates the minimum number of individuals that have to be processed to achieve, at least, one success with a certain fixed probability. Computational effort has been widely used in GP research to measure, fixing an algorithm, the difficulty of a problem or, on the contrary, fixing the problem, the performance of an algorithm. Other measures such as the success effort [5], tries to merge the measurement of efficiency and effectivity into one integrated statistic. All these composed measures share the use of the success probability.

Due to the stochastic nature of GP, there is no guarantee that the algorithm would achieve a success: the algorithm might find, or not, a solution. The success probability provides information about this property, and its calculation is straightforward; it is just the ratio between the number of success and the number of trials, or runs, in the experiment.

The main contribution of this paper is the proposal of a model of success probability in generational GP. This model considers the existence of two different -although related- problems: whether the algorithm is able to find a solution, and, given that it has been found, when that happens. The model is based on the observation that the run time, measured as generations required to find a solution, follows a lognormal distribution. If the generation is fixed, a classical binomial distribution is derived from our model [6]. Following it we discuss some practical applications of the model. For instance, given that the generation where the algorithm finds a solution (i.e. the *generation-to-success*) could be described with a known probability distribution, it would be determined when the algorithm is more likely to find a solution, and therefore, use this information to set the maximum number of generations in a well grounded way.

The paper is distributed as follows. Firstly some definitions are introduced to aid identify the semantics of the terms, then, in section 3 a general model of accumulated success probability is proposed. This general model assumes that the run time to find a solution is described by a distribution function which is empirically identified in section 4. After that, the complete model is presented in section 5, followed by a validation of the results using some experiments. Section 7 discusses some applications of the proposed model. The paper finishes with a description of the related work, an outline of the conclusions and the future work.

2 Initial definitions

In order to clarify the terminology used in this work, we first define some terms that will be widely used. We define the *generation-to-success* as the generation in which the execution of the algorithm achieves its first success. We do not think that other similar terms, such as run time, convergence generation or generation-to-termination [5] are suitable for our purposes because they include all the executions, regardless whether they found a solution or not, and we need to exclude from the definition all runs that were unable to find a solution.

A *run* is a single execution of an EA, while an *experiment* is a collection of n independent runs. Due to the random nature of EAs, many of their properties are stochastical, and thus they cannot be characterized using a run, but with an experiment. One of these properties is the *accumulative success probability*, or, using Koza's notation [4], $P(M, i)$, where M is the population size, and i the generation number. $P(M, i)$ is calculated as the ratio between the number of successful runs in generation i, $k(M, i)$, and the number of runs n in the experiment.

$$P(M, i) = \frac{k(M, i)}{n} \tag{1}$$

This estimator of the success probability is also its maximum likelihood estimator [7]. We define *Success Rate* (SR) as the accumulated success probability in an infinite number of generations, so $SR = \lim_{i\to\infty} P(M, i)$. The reader would agree with us if we state that running the algorithm for an infinite number of generations is not a general practice. Usually an experiment is run for a fixed finite number of generations, G, then the SR is given by $SR = \lim_{i\to\infty} P(M, i) \approx P(M, G)$. Since the true SR can hardly be measured in experiments, $P(M, G)$ is just an approximation to SR, and thus it can be seen from a statistical perspective as an estimator $\hat{SR} = P(M, G)$.

There is a relationship among SR, accumulated success probability and generation-to-success that is described in the next section.

3 A general model of accumulated success probability

Modeling the behaviour of success probability during execution of the generational algorithm is more difficult than modeling the number of success in a fixed generation. The latter case involves two variables: the number of success and the generation where they were found, while the former only involves the number of success.

Some of the properties of the dynamic behaviour in GP are shown by the generation-to-success. Given that a run k that has been a success, its generation-to-success g_k is a discrete random variable that can take any non-negative integer number. Let us suppose that its probability mass function is known, as well as its cumulative distribution function (CDF).

Accumulated success probability can be decomposed into two different, but related, components. One reflects how likely is the algorithm to find a solution; the other one is a probability that gives the proportion of runs that get a success before generation i. With these considerations we can state that, given an algorithm that is run for G generations, the accumulative success probability can be expressed as

$$P(M, i) = SR\, F(i) \tag{2}$$

where $F(i)$ is the CDF of the generation-to-success, and represents the probability $P(g_k < i)$ and $SR \approx \frac{k(M,G)}{n}$ is the estimation of the SR calculated in generation G. This equation provides an alternative representation of the accumulated success probability. Equation (2) has a underlying binomial nature. If we fix the generation number to, for instance, the last generation, G, and assuming that G is large enough, $F(G) \approx 1$ and thus $P(M, G) \approx k(M, G)/n$. By definition, $k(M, G)$ is a binomial random variable.

Unfortunately, the model given by (2) does not provide a close form of $P(M, i)$, but rather changes the problem, we need the CDF of generation-to-success to make (2) useful. Nevertheless, this problem is easier to address because it is a classical problem of model fitting [7]. An empirical study with some statistical tools could provide the knowledge we need to complete the model.

4 Empirical model of generation-to-success

In order to fit the generation-to-success with a known distribution, we study this random variable in four classical GP problems introduced by Koza in his first book [4]: Artificial ant with the Santa Fe trail, 6-multiplexer, even-4-parity and a polynomial regression with no Ephemeral Random Constants (ERC)[1]. We have used the implementation of these problems found in ECJ v18 with its default parameter settings.

One problem that is found is the lack of an exact knowledge about the true accumulated probability of the four problems under study. This information is necessary to have a reference to compare the model with. Since the true value of $P(M, i)$ is extremely difficult (if not impossible) to achieve, we have used another approach. Each problem has been run a large number of times, in this way the value of accumulated success probability is rather accurate, and therefore we can suppose this is the exact one without a great loss of accuracy. Three problems were run $100,000$ times whereas the fourth problem (4-parity) was run $5,000$ times because the population was larger and required more computational resources.

Measurements obtained using all the runs are the best estimations we have available for this study, so the best estimation of the SR is named $\hat{p}_{best}$. Table 1 presents the number of runs (n), number of success found (k) and best estimation available of the SR ($\hat{p}_{best}$). Looking at the SR shown in Table 1 we deduce that the difficulty of the problems ranges from very hard (4-parity, with SR around 0.06) to very easy (6-multiplexer, $SR \approx 0.96$). There are two problems whose hardness is intermediate, the artificial ant ($SR \approx 0.13$) and the regression ($SR \approx 0.29$).

[1] All the code, configuration files, scripts and datasets needed to repeat the experiments can be obtained in `http://atc1.aut.uah.es/~david/eurogp2011/`.

Table 1. Best estimations of the four problem domains under study. It reports number of runs (n), number of successful runs (k) and best estimation of SR $\hat{p}_{best}$.

	Artificial ant	6-Multiplexer	4-Parity	Regression
n	100,000	100,000	5,000	100,000
k	13,168	95,629	305	29,462
$\hat{p}_{best}$	0.13168	0.95629	0.061	0.29462

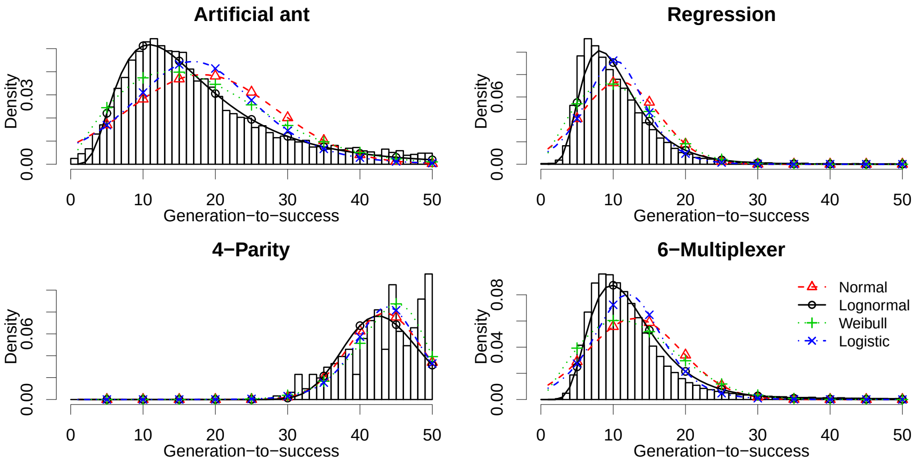

Fig. 1. Histograms of the generation-to-success of the four study cases compared with different probability density functions. The parameters of the functions have been calculated using maximum likelihood estimation. All the available successful runs have been used in the histogram and model fit.

To find the distribution that better describes the generation-to-success, we have first performed an exploratory experiment representing the generation-to-success of the four problems under study as histograms overlapped with some common statistical distributions. The parameters of the distributions have been calculated with all the successful runs and using a maximum-likelihood method implemented in R's function `fitdistr()`. Histograms for the four study cases are depicted in Fig. 1. We tried to fit data against several different distributions (Poisson, Students' t, ...), but for clarity, only the four fittest distributions are shown in Fig. 1. All the runs of the study cases were used to plot the histogram and fit the distributions.

Fig. 1 suggests that the distribution that better fits our experimental data is the lognormal. Other distributions such as the logistic or Weibull might fit data as well. However, the lognormal describes the data surprisingly well. Looking at the other study cases, the lognormal distribution fits data quite well in

Table 2. Pearson's χ^2 test for fit of generation-to-success against a lognormal distribution. Parameters of the distribution were calculated using maximum likelihood estimation. χ^2 has been calculated with a confidence level $\alpha = 0.05$. Values of χ^2 that provide evidence to reject the null hypothesis have been marked in bold.

Problem	$\hat{\mu}$	$\hat{\sigma}$	df	χ^2	$\chi^2_{\alpha,df}$
Artificial ant	2.74	0.59	4	6.56	9.49
6-Multiplexer	2.46	0.43	4	0.73	9.49
4-Parity	3.76	0.12	4	**18.89**	9.49
Regression	2.29	0.44	4	0.88	9.49

all the cases, specially the artificial ant. The 4-parity problems presents one particularity: the number of generations used to run this study case were not enough, and thus the histogram appears truncated, but even in this case the generation-to-success is well fitted by a lognormal distribution, however in this case the other distribution functions also fit well. Another particularity of the 4-parity histogram is the rugosity of its shape, the other histograms are notably smoother. It is because the histograms only show successful runs, and in case of the 4-parity they are much lower (305) than, let say, the artificial ant (13, 168).

In order to give a more rigorous validation of the model, a Pearson's χ^2 test for fit against the lognormal distribution for the four study cases was done. The test used 300 random runs sampled from the datasets, and the parameters were estimated using maximum-likelihood with the whole datasets. Table 2 shows the results, including the estimated $\hat{\mu}$ and $\hat{\sigma}$ of the lognormal distribution. The degrees of freedom (df) are given by $df = k - p - 1$, where k is the number of bins and p the number of estimated parameters. The size and limits of the bins were manually set for each problem, they were selected to contain at least 7 observations to assure a fair test. Table 2 shows that the only problem where $\chi^2 > \chi^2_{\alpha,df}$, and thus we have to reject the null hypothesis, is the artificial ant, which is also the problem that fits worse to the lognormal due to the low number of generations used and the low number of successes found (see Fig. 1).

An explanation to the lognormality of the generation-to-success might be found in the Reliability Theory. The failure rate of a physical component usually follows the shape of an inverted tub. When the component is new, manufacturing defects produce high failure rates, but once the component has reached its maturity, its failure rate decreases and remains stable for a period of time. At the end of the component life cycle the degradation of its materials and other factors increases again its failure rate. A similar behaviour is observed in GP. In its early stages, evolution has not had enough time to find a solution, the search is almost a random search. Then, after some point, the algorithm begins to find more solutions, but if that solution is not achieved, the algorithm might have converged to a local maxima and it is unlikely to move out of there, reducing the instant success probability.

This relationship between GP and Reliability Theory provides another clue for the lognormality of the convergence generation. The modeling of the probability density of lifetime is a critical issue in the Reliability Theory, so this topic has been subject of intense research. There are three models that are widely used: exponential, when the component is memoryless (which is clearly not the case here); Weibull and lognormal. Experimentation reported in this section showed that the generation-to-success of the four study cases are well fitted with a lognormal distribution. This is not, of course, a solid theoretical explanation of the lognormality of generation-to-success, but shows a certain coherence in the model and is an interesting topic for further research.

Experimentation done in this section shows that it is reasonable to model the generation-to-success using a lognormal distribution. With this result we can complete the general model previously shown.

5 A specific model of accumulated success probability

Experimentation reported above showed that the generation-to-success in the four study cases was well described by a lognormal distribution, so it seem reasonable to assume a lognormal distribution from this point. If we make this assumption, then it is straightforward to then deduce a model of $P(M, i)$ from (2) that could be used in practice. It is well known that the lognormal CDF [8] is given by

$$F(i; \hat{\mu}, \hat{\sigma}) = \Phi\left(\frac{\ln i - \hat{\mu}}{\hat{\sigma}}\right) \tag{3}$$

where $\Phi(...)$ is the standard normal CDF. If there are m runs that have converged in the experiment, and $g_k, k = 1, ..., m$ is the generation-to-success of run k, then

$$\hat{\mu} = \frac{\sum_{k=1}^{m} \ln g_k}{m} \tag{4}$$

and

$$\hat{\sigma} = \sqrt{\frac{\sum_{k=1}^{m} (\ln g_k - \hat{\mu})^2}{m}} \tag{5}$$

Using (2) and (3) yields that the accumulated success probability can be expressed as

$$P(M, i) = \frac{k(M, i)}{n} \Phi\left(\frac{\ln i - \hat{\mu}}{\hat{\sigma}}\right) \tag{6}$$

All the parameters involved in this equation are known by the experimenter.

6 Experimental validation of the model of accumulated success probability

Although data has been fitted a lognormal distribution, there is no experimental support to claim that the model of accumulated success probability given by

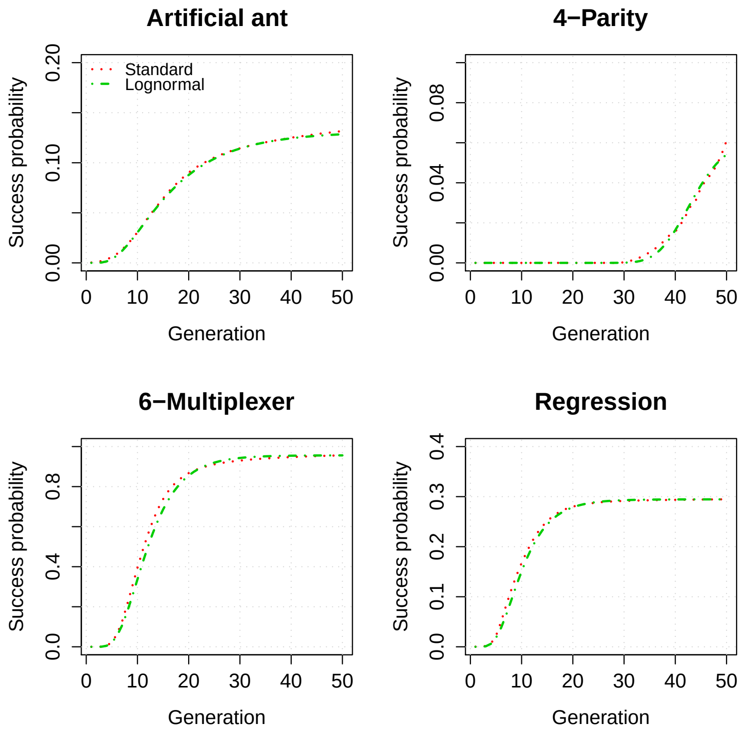

Fig. 2. Comparison between the best maximum-likelihood estimator of the accumulated success probability and the model approached using a lognormal distribution.

(6) is a correct model. So, additional experimental evidence is collected in this section.

Fig. 2 shows a comparison between $P(M, i)$ calculated using the standard maximum-likelihood method and the lognormal approximation. All the samples available in the datasets were used to calculate $P(M, i)$ with both methods. It can be seen in Fig. 2 that both methods achieve very similar results, and thus, in the study cases under consideration, when using a large number of runs, our proposal achieves estimations of $P(M, i)$ pretty close to the standard method. Nevertheless, this experiment shows an unrealistic scenario since the computational cost of running an experiment with real-world problems imposes a maximum number of runs much lower than the used in this experiment.

A collection of experiments were simulated using different values of n. Given the whole set of runs stored in the previous experiments, 25, 50, 100 and 200 runs were resampled with replacement, $P(M, i)$ calculated using both methods and finally they were depicted in Fig. 3. To give more elements to compare with, the best estimation of $P(M, i)$ (calculated with all the runs) was also included in Fig. 3.

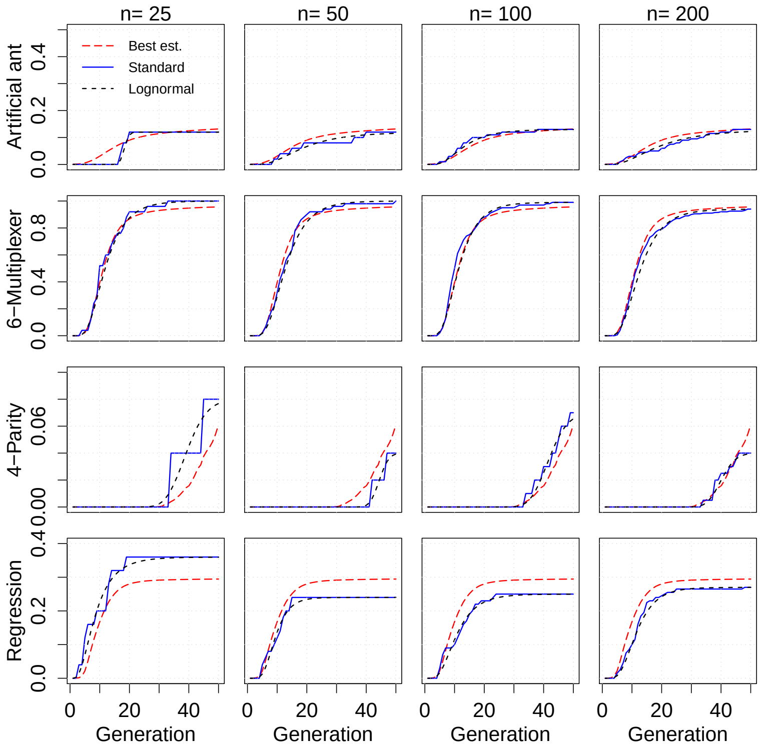

Fig. 3. Comparison between the best maximum-likelihood estimator of the accumulated success probability and the model approached using a lognormal distribution.

As can be seen in Fig. 3, when there are a high number of runs, the differences between the three curves tend to disappear, and the estimation with both methods tend to be closer to the best estimation available. More interesting is the relationship between the two methods, they yield similar estimations of the accumulated success probability, which is logical because they use the same data; if one method makes a bad estimation of the accumulated success probability, the other one also makes a bad estimation. It leads us to an almost tautological conclusion: there is no magic. With a small number of runs there are not much information available, and without information, it is simply not possible to reach good conclusions.

Despite the lack of magic of the proposed method, Fig. 3 shows an interesting property: the proposed method is able to interpolate values using the experimental data, yielding much smoother curves than the standard method. And apparently, it can be done without sacrificing the accuracy of the measure. This fact is rather clear, for instance, in the 4-parity problem with $n = 25$. A similar property is the ability of the lognormal approximation to extrapolate values of the accumulated success probability. This is interesting in early generations, where there are no success due to a low, but not null, success probability. In these cases the standard method yields a null estimation of the success probability while the lognormal yields a non zero value.

Another interesting fact that can be found in Fig. 3 is the excellent estimation made in the 4-parity problem. Despite the fact that the experiment was run for too few generations and it was the domain with the poorest model fit, it generates a nice approximation to the maximum-likelihood estimator of the accumulated success probability. This fact could be quite interesting to reduce the number of generations needed to study the performance of GP using less computational resources, however we feel that this issue requires more study.

7 Discussion

Experiments that were carried out showed evidence about the good properties of the model and the lognormal nature of generation-to-success. In particular, how the generation-to-success of the four problems studied fits a lognormal distribution opens some interesting applications. For instance, it might be used to set the maximum number of generations that the algorithm should be run using a well grounded criteria.

The experimenter would carry out an exploratory experiment with a relatively low number of runs, then the estimators $\hat{\mu}$ and $\hat{\sigma}$ could be calculated. The determination of the number of runs needed to estimate the parameters is a classical problem in statistical inference, and the transformation between normal and lognormal is straightforward, $X \sim N(\mu, \sigma) \Rightarrow e^X \sim LN(\mu, \sigma)$ and $X \sim LN(\mu, \sigma) \Rightarrow \ln(X) \sim N(\mu, \sigma)$ [8], using this transformation the number of runs in the exploratory experiment can be determined using

$$n = \frac{z_{\alpha/2}^2 \; s^2}{e^2} \tag{7}$$

where e is the desired level of precision for the estimation of the mean, given in the same unit than s, s the standard error of the samples and $z_{\alpha/2}$ is the upper-$\alpha/2$ critical point from $N(0, 1)$ [9]. The probability of getting at least one success in generation i is given by $P(M, G) \; F(i; \hat{\mu}, \hat{\sigma})$, while the probability of not getting any success in generation G is

$$\varepsilon = 1 - P(M, G) \; F(G; \hat{\mu}, \hat{\sigma}) \tag{8}$$

This equation provides an estimation of the error ε that is introduced by limiting the maximum number of generations to G. Moreover, if we set a maximum error that is tolerable, we could calculate G from (8), yielding the maximum number of generations that the algorithm should be executed to achieve that error. This is a grounded method able to determine the maximum number of generations.

8 Related work

Despite the importance of understanding the statistical properties of the several measures used in the experimentation of GP, there are not too many studies available. One of the metrics that has motivated some papers is Koza's computational effort. Angeline first observed that computational effort [10] is actually a random variable, and concluded that the stochastic nature of the computational effort should be handled with statistical tools. Some time after, Keijzer [11] calculated computational effort using confidence intervals (CIs), achieving a remarkable conclusion: when success probability is low, CIs of the computational effort are almost as large as the computational effort. In order words, the variability of the computational effort is similar to its magnitude, and thus, in that case, the high dispersion of computational effort degrades its reliability.

To the author's knowledge, the only systematic attempt made to understand why the computational effort presents the variability observed by Keijzer was presented by Christensen [12]. He identified three sources of variability and provided empirical data that gave some light to the circumstances that reduces the reliability of the computational effort. More research in this area was done by Walker [13, 14], who studied how to apply CIs to the calculus of computational effort, and Niehaus [15], who investigated the statistical properties of the computational effort in steady-state algorithms.

9 Conclusions and future work

One of the most important metrics in GP is the accumulated success probability. This measure reflects how likely is the algorithm to find a solution in a given generation, and has been widely used in research and practice to understand the performance, or compare different algorithms.

When we fix the generation in GP, the number of runs that finds a solution can be modeled using a binomial distribution, but its variation with the generation is no longer described with a binomial, and a more complex model is required. The dynamic behaviour of the algorithm can be modeled using the generation-to-success, or generation in which the run achieves its first success. Experiments suggested that generation-to-success can be fitted with a lognormal distribution.

The combination of SR, which is a binomial random variable, and the CDF of generation-to-success can model the accumulative success probability. Experiments carried out show that the accumulated success probability calculated with this model yields curves quite close to the maximum-likelihood method, with a smoother shape and it is able to extrapolate values in zones where no success run has been found. The lognormality of generation-to-success opens some interesting applications, such as a well grounded stopping condition in GP.

The experimentation that has been done used some well known problem domains in GP, and despite we do not find any reason that may limit the generality of the results. The accumulated success of probability is used in other Evolutionary Computing disciplines, such as Genetic Algorithms, and it seems reasonable that we could expect the same behaviour of the generation-to-success. It could also be interesting to answer why the generation-to-success follows a lognormal distribution.

Acknowledgments. This work was partially supported by the MICYT project ABANT (TIN2010-19872) and Castilla-La Mancha project PEII09- 0266-6640.

References

1. Eiben, A.E., Smith, J.E.: Working with Evolutionary Algorithms. In: Introduction to Evolutionary Computing. Springer-Verlag (2009) 241–258
2. Bartz-Beielstein, T.: Tuning evolutionary algorithms: overview and comprenhensive introduction. Technical Report 148/03, Universit́at Dortmund (2003)
3. Luke, S., Panait, L.: Is the perfect the enemy of the good? In: Genetic and Evolutionary Computation Conference, Morgan Kaufmann (2002) 820–828
4. Koza, J.: Genetic Programming: On the programming of Computers by Means of Natural Selection. MIT Press, Cambrige, MA (1992)
5. Walker, M., Edwards, H., Messom, C.H.: Success effort and other statistics for performance comparisons in genetic programming. In: IEEE Congress on Evolutionary Computation, Singapore, IEEE (2007) 4631–4638
6. Barrero, D.F., Camacho, D., R-Moreno, M.D.: Confidence Intervals of Success Rates in Evolutionary Computation. In: GECCO '10: Proceedings of the 12th annual conference on Genetic and Evolutionary Computation, Portland, Oregon, USA, ACM (2010) 975–976
7. Montgomery, D.C., Runger, G.C.: Applied Statistics and Probability for Engineers, 4th Edition. 4th edn. John Wiley & Sons (2006)
8. Limpert, E., Stahel, W.A., Abbt, M.: Log-normal distributions across the sciences: Keys and clues. BioScience **51** (2001) 341–352
9. Smith, M.F.: Sampling Considerations In Evaluating Cooperative Extension Programs. In: Florida Cooperative Extension Service Bulletin PE-1, Institute of Food and Agricultural Sciences. University of Florida. (1983)
10. Angeline, P.J.: An investigation into the sensitivity of genetic programming to the frequency of leaf selection during subtree crossover. In: GECCO '96: Proceedings of the First Annual Conference on Genetic Programming, Cambridge, MA, USA, MIT Press (1996) 21–29
11. Keijzer, M., Babovic, V., Ryan, C., O'Neill, M., Cattolico, M.: Adaptive logic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), San Francisco, California, USA, Morgan Kaufmann (2001) 42–49
12. Christensen, S., Oppacher, F.: An analysis of koza's computational effort statistic for genetic programming. In: EuroGP '02: Proceedings of the 5th European Conference on Genetic Programming, London, UK, Springer-Verlag (2002) 182–191
13. Walker, M., Edwards, H., Messom, C.H.: Confidence intervals for computational effort comparisons. In: EuroGP. (2007) 23–32
14. Walker, M., Edwards, H., Messom, C.: The reliability of confidence intervals for computational effort comparisons. In: GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, New York, NY, USA, ACM (2007) 1716–1723
15. Niehaus, J., Banzhaf, W.: More on computational effort statistics for genetic programming. In Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E., eds.: Genetic Programming, Proceedings of EuroGP'2003. Volume 2610 of LNCS., Essex, Springer-Verlag (2003) 164–172