

Solution-Locked Averages and Solution-Time Binning in Genetic Programming

Riccardo Poli

School of Computer Science and Electronic Engineering, University of Essex, UK

Abstract. Averaging data collected in multiple independent runs across generations is the standard method to study the behaviour of GP. We show that while averaging may represent with good resolution GP's behaviour in the early stages of a run, it blurs later components of the dynamics. We propose two techniques to improve the situation: solution-locked averaging and solution-time binning. Results indicate that there are significant benefits in adopting these techniques.

1 Introduction

In recent work on averaging event-related potentials (ERPs) [5], we highlighted the fact that while averaging temporal sequences of data points has the beneficial effect of reducing noise, it also presents a serious drawback. When some of the components (or waves) of a signal occur at different times in different trials, a blurring effect takes place as a result of averaging, i.e., averaging may introduce *systematic errors* while it reduces stochastic ones. This limits how much we can learn from standard temporal averages. While the problem has been known for a long time in electrophysiology (e.g., see [4,2,7]), no easy solution is available. [5] suggested the use of a simple technique to reduce this blurring effect: subdividing the data into bins based on the time at which subjects gave their response. Bin optimisation via GP further improved results [1].

The similarity between the averaging of ERPs and standard analysis practise in GP is striking. Statistics such as mean and best fitness, fitness standard deviation, diversity, solution size, etc. are routinely recorded during GP runs. Because GP is stochastic, multiple independent runs of the system are performed and the statistics recorded in each run are then combined, typically by averaging the data corresponding to each generation. So, if averaging of ERPs presents problems of blurring and mis-representation of statistics, then we may expect averaging to produce similar effects also in GP.

To understand this let us consider the typical dynamics of GP runs. For many problems runs are not homogeneous marches towards a solution. Runs may have different phases, e.g., an initial period of rapid evolution, followed by one or more periods of quasi stagnation separated by short transients of rapid evolution when improved solutions are found. While events of this kind will be present in most runs, not all runs will have the same events. For example, unsuccessful runs will lack the final rush to the solution. Also, even when qualitatively the same events occur in some runs, it is unlikely that these will always occur at exactly the same time in different runs.

Why is this a problem? Let us imagine the statistics gathered over three independent runs presented a rapid initial increase in the measurements followed by a static

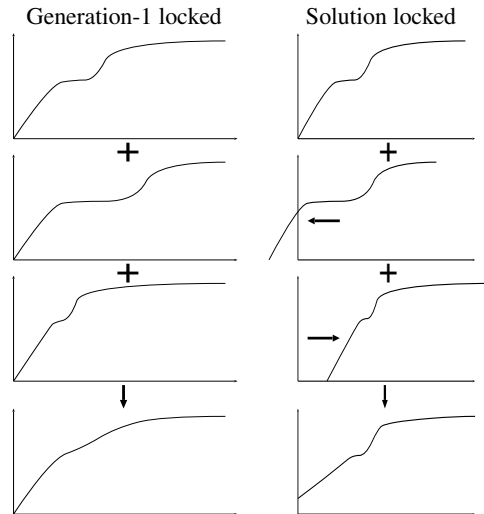


Fig. 1. Example of distortions produced by averaging: standard (generation-1-locked) averaging (left) and solution-locked averaging (right)

period which eventually led to discovering a solution and a final rapid convergence of the population towards it. This is shown in Fig. 1(left). The initial rapid growth is phase locked with the start of the run while the final rapid growth occurs at different generations in different runs. Averaging the data in the traditional way (bottom left) correctly represents the initial rapid growth present in the three runs starting from generation 1. Averaging, however, turns the second phase of rapid increase which occurs at different times in different runs into an inconspicuous slope which is not present in any of the runs and which may be misinterpreted as a slow-down of the initial rapid growth.

This phenomenon is present whenever the temporal data being averaged contain components with different latencies (delays). To understand the nature of the deformations produced by averaging in these conditions, let us consider the simplest possible case: let us imagine that there are only two components in the data recorded in an experiment: a component, $s(t)$, which always starts at generation 1 and continues throughout a run, and a variable-latency component, $r(t)$, which is associated with the final phases of a successful run. For simplicity, let us imagine that the component $r(t)$ starts at random times during a run. However, once it starts, within exactly n generations a solution is found. So, this component is phase-locked with the final generation of a run. Because the conditions which determine the start of $r(t)$ is stochastic, the latency of $r(t)$ is a stochastic variable. Let $\rho(t)$ be the latency distribution of $r(t)$. Let us also assume that $r(t)$ and $s(t)$ are additive, i.e., the statistics of interest is simply the sum of these two components. Under these assumptions it is easy to show [2,7] that the ordinary average of multiple run data is $a_s(t) = s(t) + r(t) \star \rho(t)$, where \star is the convolution operation. Given that latency distribution $\rho(t)$ is non-negative (and typically unimodal), this means that the ordinary average can only show a smoothed (low-pass filtered) version of the variable-latency component $r(t)$. So, averaging introduces systematic errors in the

analysis of events that precede (and are related to) the solution of a problem. This blurring is likely to prevent the detection of important phenomena, e.g., related to factors that determine whether a run is successful.

Extending the work in [5] and the response-locked averaging technique used in the ERP literature [4], in this paper we propose and analyse two simple countermeasures one can use to improve the analysis of GP statistics: solution locking and solution-time binning. Both techniques pre-process the data before any generation-by-generation averaging (or other aggregation) is performed.

2 Solution-Locked Averages and Solution-Time Binning

Solution locking involves time-shifting the statistics collected in different runs in such a way that the data corresponding to the generation in which a solution was found are aligned, instead of aligning stats at the first generation.¹ The situation is illustrated in Fig. 1(right). Let us see what averaging these shifted data produces.

For the simple model described above, the solution-locked average is given by $a_r(t) = r(t) + s(t) \star \rho(-t)$ [2,7]. In other words, the solution-locked average will smooth out the component $s(t)$ which is phased locked with the start of the run, but it will not alter $r(t)$. So, this form of averaging can provide very useful information on the late stages in a run, which, as indicated in the previous section, are never well represented in ordinary averages. So, this simple trick can reveal important events which immediately precede (and contribute to) the discovery of solutions. The price to pay is that the early phases of runs will be represented with a low effective resolution.

While this technique is very useful, a researcher will still be presented with two alternative and potentially conflicting representations of the same data: one based on standard averaging and one based on solution-locked averaging. Both for $a_s(t)$ and for $a_r(t)$, the severity of the smoothing effectively depends on how narrow the distribution $\rho(t)$ is. A key problem is that running more experiments and *averaging more data does not help increase the fidelity of the reconstructed statistics* because the convolution with $\rho(t)$ or $\rho(-t)$ introduces a systematic error in the averaging process.

Fortunately, as suggested in [5] for ERPs, a very simple technique can improve the situation: *binning runs based on their recorded solution time and then computing bin averages*. This has the potential of radically reducing the blurring problems of ordinary and solution-locked averages. In particular, solution-time binning can significantly improve the resolution with which variable-latency components of run statistics can be recovered via averaging. The qualitative reason for this is simple to understand.

The idea is that if one selects out of a dataset all those runs which were successful within approximately the same amount of time, it is more likely that similar GP dynamics will have taken place within those runs than if one looked at the whole dataset. So, within those runs, components of statistics that would normally have a widely variable latency might be expected, instead, to present a much narrower latency distribution. In other words, we can expect more homogeneity in binned data. Thus, we should find that, for the runs within a bin, fixed- and variable-latency components are much more

¹ Runs where a solution wasn't found can be treated in whatever way they are treated in relation to ordinary averages.

synchronised than if one did not divide the dataset. Averaging such data should, therefore, allow the rejection of noise while at the same time reducing also the undesirable distortions and blurring associated with averaging.

We look at these qualitative ideas more formally in the next section.

3 Analysis of Resolution of Bin Averages

Let us consider in what ways binning by solution time and then averaging would alter the resolution with which fixed latency and variable-latency components of runs statistics can be recovered. Here we will specifically look at the case where runs are aligned at the first generation. However, there are symmetric properties for binned solution-locked averages. For space limitations we omit many intermediate calculations and extensions.

Let us define a function $\delta(x)$ that returns 1 if x is true, and 0 otherwise. It is simple to show that the (ordinary) average of the data in the bin corresponding to the solution-time interval $[r_1, r_2]$ is given by $a_s^{[r_1, r_2]}(t) = s(t) + r(t) \star \rho^{[r_1, r_2]}(t)$ where $\rho^{[r_1, r_2]}(t) = \delta(r_1 \leq t < r_2) \rho(t) / \int_{r_1}^{r_2} \rho(t) dt$ is the convolution kernel responsible for $r(t)$ appearing blurred in the average. So, in order to see whether $a_s^{[r_1, r_2]}(t)$ provides a better (less blurred) representation of $r(t)$ than $a_s(t)$, we need to analyse the differences between the distributions $\rho^{[r_1, r_2]}(t)$ and $\rho(t)$.

The difference between the two is that $\rho^{[r_1, r_2]}(t)$ is the product of $\rho(t)$ and a rectangular windowing function, $\delta(r_1 \leq t < r_2)$. In the frequency domain, therefore, the spectrum of $\rho^{[r_1, r_2]}(t)$ is $\mathcal{R}^{[r_1, r_2]}(f) = \mathcal{R}(f) \star \Delta(f)$, where $\Delta(f)$ is a scaled and rotated (in the complex plane) version of the *sync* function. Therefore, $\mathcal{R}^{[r_1, r_2]}(f)$ is a smoothed and widened version of $\mathcal{R}(f)$. In other words, while $\rho^{[r_1, r_2]}(t)$ is still a low-pass filter, it has a higher cut-off frequency than $\rho(t)$. Thus, *binning by solution time and then averaging improves the resolving power on averages*. Indeed, it is possible to prove that in the limit of very small bins the bin average is an unbiased estimator of the true statistics.

It is possible to generalise these results to the case of multiple additive components in the statistics recorded in a run. The generalisation shows that whenever there is a non-zero correlation between the latency of such components and the solution time, then binning by solution time produces bins where the variability in the latency of all such components is reduced. As a result averaging the data in a bin leads to averages that are less blurred and more representative of reality.

4 Test Problems and Algorithms

To test the ideas proposed in this paper we used a tree-based GP system [3], namely a version of TinyGP [6, appendix B]. The system used tournament selection with a steady state replacement strategy where individuals are replaced with negative tournaments. Both for selection and replacement we used tournaments of size 2. We evolved populations for 100 generations. Programs were initialised using the grow method (see, for example, [6]). Offspring were generated using either sub-tree crossover (with uniform selection of crossover points) or point mutation. When mutation was chosen the nodes in the selected parent program were replaced with random nodes of the same

arity with a probability of 5% (per node). Runs were not stopped when a solution was found, but the first time a solution was discovered was recorded.

We considered two problems. The first is the well-known Even-6 parity problem. We used the primitive set: $\{X1, X2, X3, X4, X5, X6, \text{AND}, \text{OR}, \text{NOR}, \text{NAND}, \text{XOR}, \text{EQ}\}$. The fitness of a program was the number of entries of the truth table of the even-6 parity function that the program could correctly predict. So, program fitnesses are integers between 0 and 64. We performed 1,000 independent runs of TinyGP with this problem. We used populations of 2,000 individuals. Initial programs had a maximum depth of 6 (counting one-node programs as having a depth of 0). The maximum size programs were allowed to grow during evolution was 1,000 nodes. Crossover was applied with 80% probability, while mutation was applied with 20% probability.

The second problem is the Sine symbolic regression problem, which requires programs to fit the sine function over a full period of oscillation. We used 63 fitness cases obtained by sampling $\sin(x)$ for $x \in \{0.0, 0.1, 0.2, \dots, 6.2\}$. We define a *success* to be a run where the best fitness was less than 3.15, or an average error of less than 0.05 over the 63 test cases. Here we performed 500 independent runs with populations of size 10,000. The primitive set included the functions ADD, SUB, MUL, DIV (protected) plus 100 random constants uniformly distributed in the range $[-5, 5]$ and one input variable. The max initial depth for trees was 5. Crossover and mutation rates (per individual) were 10% and 90%, respectively. Fitness was the sum of absolute errors.

The parameters for each problem were chosen in such a way to obtain approximately a split of one third and two thirds between successful and unsuccessful runs.

5 Results

To exemplify the benefits of our averaging techniques, we will look at best-of-generation and mean fitnesses averaged across multiple runs for the Even-6 parity problem and best-of-generation fitnesses for the Sine symbolic regression (in many symbolic regression problems means vary so much that they are rather uninformative). We will compute averages both with and without solution-locking and solution-time binning.

Before we analyse the data, we would like to stress a few important details about the averaging procedures we adopted. Firstly, when computing solution-locked averages, one needs to *shift runs relatively to one another* in such a way to align their solution time. The method, however, leaves unspecified the *absolute shift* of the solution time with respect to the reference system used to collect or generate the data (which in our case is a Cartesian reference with generations along the abscissas and fitness along the ordinates). By convention, here we have aligned the solution time to the abscissa corresponding to last generation (generation 100). Since most successful runs lasted less than 100 generations, to ensure every point in a plot was an average of the same number of runs, we extended runs as if they started before generation 1, freezing their fitness stats in generation 0, -1, -2, etc. to the value they had at generation 1. Secondly, we should also note that below we will distinguish between successful and unsuccessful runs (see Tab. 1). Finally, note that when computing solution-locked averages involving a mix of successful and unsuccessful runs or only unsuccessful runs, we used the convention that the *solution time* for unsuccessful runs was their last generation.

Table 1. Success rate statistics for the Even-6 parity and the Sine regression problems

Problem	Successful Runs	Total	Success Rate
Even-6 Parity	351	1,000	35.1%
Sine Symbolic Regression	144	500	28.8%

5.1 Ordinary Averages vs. Solution-Locked Averages

Let us first compare ordinary averages and solution-locked averages in the absence of binning. These are shown at the top of Fig. 2. Note that these averages do not distinguish between successful and unsuccessful runs, as it is typically done when reporting fitness statistics in the GP literature.

Looking at ordinary averages and comparing them across the two problems, we see the usual depiction of the dynamics of a GP system: there is a rapid improvement in fitness in the early generations of a run which is followed by a period where improvements are either rarer or smaller or both which eventually leads to a solution. In the Even-6 parity problem effectively the algorithm appears to be almost stagnating in the later phases of a run. What changes across the two problems is the relative slope of the final phases. Other than that, everything seems to confirm the dynamics of the fitness plots we have seen over and over again in the literature.

If we now focus on the solution-locked averages at the top of Fig. 2, however, we get a different picture. Firstly, the plots of such averages are significantly different from those of ordinary averages. For example, we see changes of curvature in the middle of a run. In the case of the Even-6 parity problem we also see an apparent period of stagnation followed by an acceleration in the late stages of runs.

The presence of flexes in the middle of runs and re-accelerations towards the end of runs go against almost every bit of published literature in GP. How is this possible? Which averages should we believe: ordinary or solution-locked? Neither and both.

As we emphasised above we have extended runs before their first generation for the purpose of realigning them at generation 100 and then computing a solution-locked average. So, while we can reasonably trust what's happening near generation 100 (by convention, the solution time), some of the effects we see many generations before generation 100 may be entirely due to our extending runs. This does not mean, however, that solution-locked averages are less reliable than ordinary averages. Indeed, symmetrically, in order to obtain ordinary averages, we have extended runs at the other end, by not stopping them when a solution was first found. So, we should also be very careful in interpreting what we see in ordinary averages when we are many generations away from the first. In some cases the effects we see in such averages in the late stages of a run, e.g., the saturation of best and mean fitnesses, may be entirely artefactual. If we are to believe solution-locked averages (which in principle have the best resolving power in the late stages of a run), in our two problems fitness is still growing significantly when solutions are first found.

To see things more clearly we really need to divide up the successful runs from the unsuccessful ones. This is done in Fig. 2 (middle and bottom, respectively). Note that, since for the unsuccessful runs we conventionally set their solution time to be their final

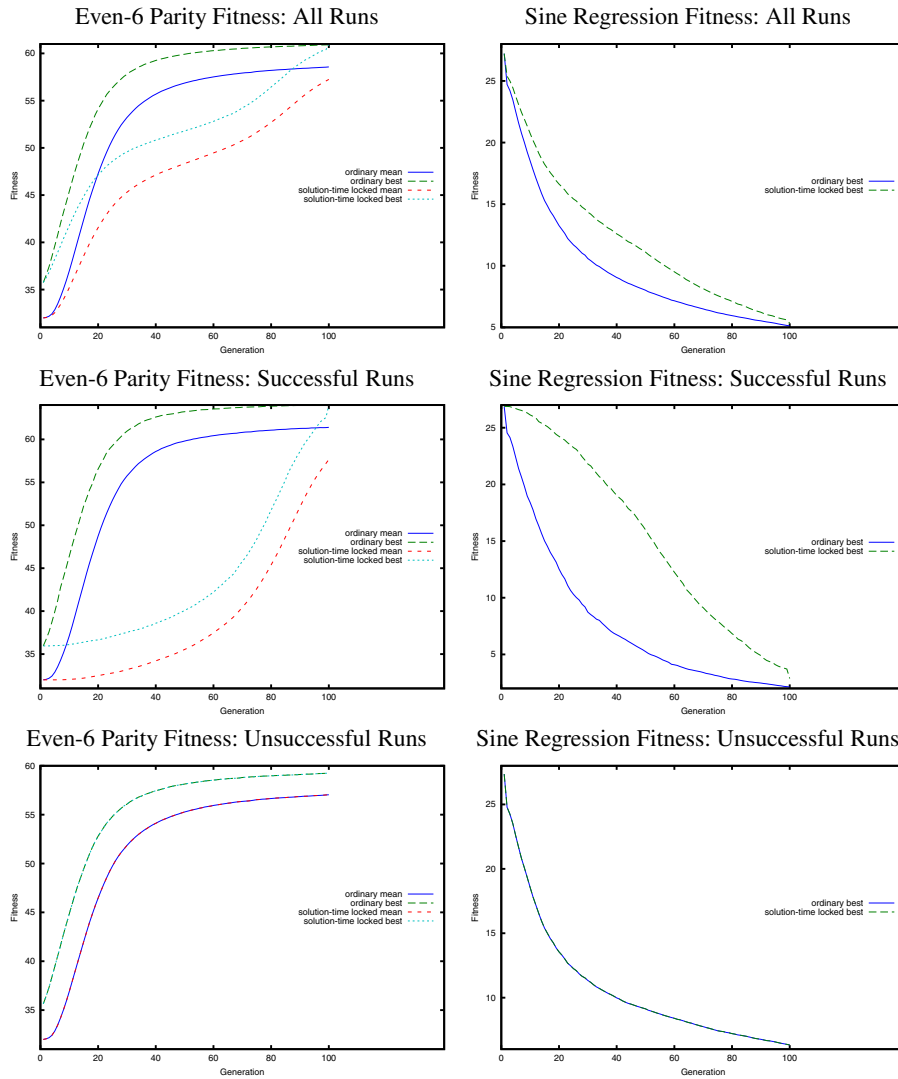


Fig. 2. Comparison between ordinary and solution-locked averages of mean and best population fitnesses for the Even-6 Parity problem (left) and of the mean fitness for the Sine symbolic regression problem (right) when all runs are averaged (top), when only the successful runs are averaged (middle) and when only the unsuccessful runs are considered (bottom)

generation, the plots of ordinary averages and solution-locked averages are on top of each other (see bottom of Fig. 2).

From a qualitative viewpoint the plots of *ordinary* averages of successful and unsuccessful runs are not very different, except, in the fitness values that are reached at the end of a run. Since the plots of averages across all runs are a blend of those for successful and unsuccessful runs, it is then not surprising to see that ordinary averages in the plots at the top of Fig. 2 are also qualitatively identical to the corresponding plots in the middle and bottom of the figure. However, now that successful runs are averaged separately, we can see that for them the differences between ordinary and solution-locked averages are further emphasised (see middle plots in the figure), with both problems showing much bigger end-of-run slopes in their solution-locked averages. In other words, towards the end of successful runs best and average fitnesses either don't saturate or saturate a lot less than we are used to see in ordinary GP averages.

The little step characterising the rightmost points in the solution-locked best fitness plots for the successful runs represents the time where a solution is first visited. So, such a step is present in all plots of this kind, irrespective of problem or system. In problems where fitness is discrete, the variation in best fitness between the last generation and the penultimate generation is an indication of how far (on average) the best of population was from solving the problem.

All of this is not visible in ordinary averages because of the smoothing biases of averaging. As we explained above, both ordinary averages and solution-locked averages are affected by a smoothing effect. The effect influences any component in the statistics which is not phase-locked with the beginning of the run or the end of the run, respectively. Elements which are phase-locked with the solution-time will be smoothed by the kernel $\rho(t)$ — the solution time distribution — in ordinary averages, while elements which are phase-locked with the start of the run will be smoothed by the reflection of that kernel, $\rho(-t)$, in solution-locked averages. Whether or not this smoothing has an effect depends on the width of the solution time distribution (narrower distributions producing less blurring than wider ones) and on whether or not the statistics contain components outside the band of the smoothing kernel. In other words, the blurring of statistics that are already smooth is unlikely to produce significant artifacts, while we should expect to see major deformations when statistics present rapid changes.

Thus, to ascertain the potential degree of blur introduced by averaging (whether ordinary or solution-locked), we need to look at the solution time distributions for our two problems. These are shown in Fig. 3. It is clear from these plots that both problems have a wide distribution which can potentially lead to significant blurring effects.

5.2 Solution-Time Binned Averages

To get a clearer picture of what is happening, we need to use solution-time binning on the successful runs. This does not only improve resolution of averages, it also clarifies whether different mechanisms are responsible for different run times.

Figs. 4 and 5 show the results of binning runs and then averaging, for our two problems, using 10 bins of 10 generations each. Naturally, we can align the runs in each bin at generation 1 (as in ordinary averaging) or at the generation where they solved the problem (as in solution-locked averaging). Let us analyse these figures.

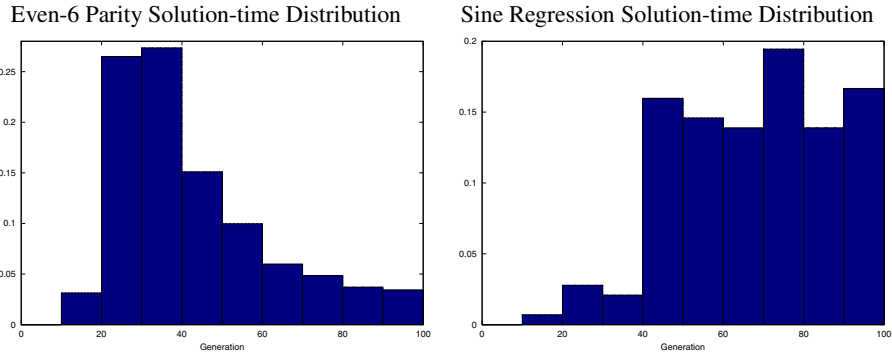


Fig. 3. Distribution of solutions times for the Even-6 Parity and the Sine symbolic regression problems

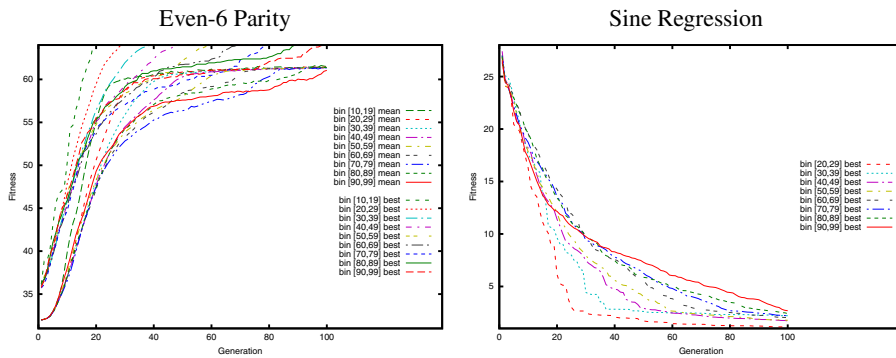


Fig. 4. Results of binning runs with the Even-6 Parity and Sine symbolic regression problems followed by *ordinary averaging* for successful runs

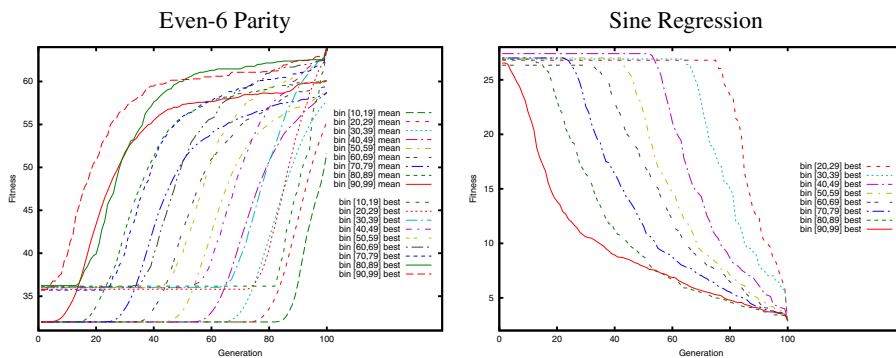


Fig. 5. Results of binning runs with the Even-6 Parity and Sine symbolic regression problems followed by *solution-locked averaging* for successful runs

Let us first focus on the results of solution-time binning in combination with ordinary averaging of the data in the bins shown in Fig. 4. We see here that for the Sine problem the plots of the averages of best fitness are effectively monotonic and featureless, except perhaps for the quickest runs. The plots for different bins are also quite similar, all sharing similar initial trajectories. The only significant differences between bin averages are slight differences in slope after this initial transient: runs which solved the problem sooner present bigger gradients than runs that took many generations to solve the problem. This suggests that GP solves this problem by a progressive and relentless march towards the solution. The difference between the runs that did it quickly and those that took longer appears to be simply a question of the rate at which the march proceeded, which in turn is likely to be due to the natural variability one would expect from stochastic search algorithms. Because run statistics are relatively smooth, the potential blurring effects of averaging (without binning) due to the convolution with the solution-time distribution in Fig. 3 had a limited influence on the averages shown in Fig. 2. This is the reason for their qualitative similarity with corresponding bin averages.

Let us look at the solution-locked averages for the Even-6 Parity problem. In Fig. 4 we see that actually during the generations that precede a solution evolution is still very rapid and only barely showing signs of slowing down. Solution-time bin averages clarify that essentially GP has alternative modes of solution. Firstly, all runs present a rapid initial evolution. During this there seems to be some essential preparation taking place for solving the problem. Indeed, no run solved the problem in less than 10 generations. After this initial transient of 10 to 20 generations, the runs in the first three non-zero bins essentially then continue their march towards the solution without delay (albeit at different speeds). Runs in other bins, instead, tend to slow down significantly (although never completely stagnating) after the transient eventually reaching a solution through a series of rare improvements. In other words, solution-time binning reveals the partial deceptiveness of this problem. Indeed, larger versions of the problem have been known for years to be extremely difficult to solve for GP.

All this makes sense in the light of what we know from a variety of studies with parity functions and continuous symbolic regression. There is, however, one surprising element in the ordinary averages of solution-binned statistics. Namely, in the bins collecting runs that took some time to solve the Even-6 Parity problems, the best fitness appears to speed up its upward motion in the very last stages of a run, which is counter-intuitive. Indeed this is an averaging artifact, as shown by the solution-time binned solution-locked averages shown in Fig. 5. As one can see there, all plots are essentially flat in the generations that precede the solution (which we positioned at generation 100, by convention). It is only at the last generation that a sudden increase in best fitness is present. This is simply associated with the discovery of a solution. The ramps shown in Fig. 4 in the late stages of a run are simply the result of convolving this solution-discovery peak with the kernel $\rho^{[r_1, r_2]}(t)$ associated with each solution-time bin. Although this kernel is typically much narrower than $\rho(t)$, it still has a non-zero width, leading to this residual form of blurring.

Symmetrically, the 10-generation wide sigmoid-like ramps shown in the early parts of each plot in Fig. 5 are artifacts. In fact, they are the result of blurring the initial rapid fitness increase shown in Fig. 4 with the convolution kernel associated with each bin.

So, even with solution-time binning, to get the most and best information (and the least artifacts) out of experimental data, it is essential to compare and analyse both solution-locked averages and ordinary averages. Without these tools, however, relatively little can be learnt from averages.

6 Discussion and Conclusions

Averaging data collected in multiple independent runs across generations is the standard method to study the behaviour of GP. However, as we have illustrated in this paper, standard averaging may suffer from what we could call a key-hole or a magnifying-glass effect. That is, while this technique is able to represent with good resolution the early behaviour of GP, it does so at the cost of putting everything else out of focus. When the dynamics of a GP system (as represented by a statistical descriptor that we want to average across runs) is simple and featureless, this lack of resolution may deform averages to a limited extent. However, when evolution is characterised by multiple rapid transients separated by periods of slow change, then we don't have a way of constructing a clear picture of what happens in a GP system. This may be particularly problematic for the late and all-important phases of evolution when a solution is finally constructed.

In this paper we have suggested that solution-locked averaging, i.e., temporally shifting the statistics of successful runs so as to ensure the times at which a solution is first visited are aligned across runs and then averaging, may be a partial solution to these problems. Solution-locked averaging, however, still suffers from a key-hole effect: while it shows the late stages of a run in full resolution, it blurs everything else.

In the paper we have also proposed a second extremely simple technique — binning runs based on solution times and then averaging — that can alleviate the problems of the standard form of averaging mentioned above. The technique is based on one simple and realistic assumption: that roughly the same dynamics takes place in a GP system which is able to solve a problem, when that problem is solved in approximately the same time. For this reason, in such runs, the distribution of latencies of all variable-latency components of the data (including those phase-locked with the solution) should be narrower than if one considered an undivided dataset. As a result, averaging the runs in a solution-time bin should provide a truer picture of the dynamics of GP.

We empirically validated the binning technique by applying a tree-based GP system to two radically different problems. We also provided a theoretical analysis of the resolution of averages with and without binning, which showed that there are benefits in applying solution-time binning even when there is still a substantial variability in the latency of variable-latency components after solution-time binning.

Empirical results and theory points unequivocally in one direction: averaging after solution-time binning produces clearer representations of GP dynamics. When this is further combined with solution-locked averaging, we obtain a much less ambiguous representation of the stages that eventually lead to a solution in successful runs. Much can be learnt from studying such aggregate statistics.

The degree to which ordinary averages deform late-run statistics and the degree to which solution-locked averages smooth the early-run statistics depend on the shape and width of that solution-time distribution. When the distribution is narrow, then effectively runs are already binned. So, solution-time binning cannot be expected to improve

resolution by much. Binning is still useful, however, since it clarifies if different run durations are associated with different modes of solution.

In the past, solution-time distributions have frequently been reported and used to compute the computational effort [3]. So, in principle we can now look back at those results to understand to what degree reported average statistics may have been blurred. In new experiments, it seems appropriate to always look at the solution time distribution to decide whether binning and solution-locking should be used. It may also be possible for researchers to retroactively reevaluate, shift and bin results of old experiments, and possibly discover new phenomena or further confirm existing theories.

Given the simplicity of the methods we have proposed, we would like to encourage researchers to adopt them as part of their set of data analysis routines to ensure the best possible use of the evidence contained in experimental results.

In relation to future work, in a sense, we can think of solution-time binning as a spot in the middle ground between single-run analysis and ordinary averages. In the future we would like to further explore this middle ground. For example, we would like to see if binning using gradual membership functions, as we did in [5] for ERPs, can provide even better reconstruction fidelity, if setting bin sizes on the basis of solution time quantiles and/or the noise in the data is beneficial to make best use of the available runs, and if solution-locked and ordinary averages can be jointly used (e.g., in the frequency domain as was done in [7] for ERPs) to further refine the reconstruction of GP's dynamics.

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council under grant "Analogue Evolutionary Brain Computer Interfaces" (EP/F033818/1).

References

1. Citi, L., Poli, R., Cinel, C.: High-significance averages of event-related potential via genetic programming. In: Riolo, R.L., O'Reilly, U.-M., McConaghy, T. (eds.) *Genetic Programming Theory and Practice VII, Genetic and Evolutionary Computation*, May 14-16, vol. 9, pp. 135–157. Springer, Ann Arbor (2009)
2. Hansen, J.C.: Separation of overlapping waveforms having known temporal distributions. *Journal of neuroscience methods* 9(2), 127–139 (1983)
3. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge (1992)
4. Luck, S.J.: *An introduction to the event-related potential technique*. MIT Press, Cambridge (2005)
5. Poli, R., Cinel, C., Citi, L., Sepulveda, F.: Reaction-time binning: a simple method for increasing the resolving power of ERP averages. *Psychophysiology* (Forthcoming, 2010)
6. Poli, R., Langdon, W.B., McPhee, N.F.: *A field guide to genetic programming* (With contributions by J. R. Koza) (2008), <http://lulu.com>, <http://www.gp-field-guide.org.uk>
7. Zhang, J.: Decomposing stimulus and response component waveforms in ERP. *Journal of neuroscience methods* 80(1), 49–63 (1998)