Reassembling Operator Equalisation - A Secret Revealed

Sara Silva [1,2]
[1] INESC-ID Lisboa, IST / UNL, Portugal
[2] CISUC, University of Coimbra, Portugal
sara@kdbio.inesc-id.pt

ABSTRACT

The recent Crossover Bias theory has shown that bloat in Genetic Programming can be caused by the proliferation of small unfit individuals in the population. Inspired by this theory, Operator Equalisation is the most recent and successful bloat control method available. In this work we revisit two bloat control methods, the old Brood Recombination and the newer Dynamic Limits, hypothesizing that together they contain the two main ingredients that make Operator Equalisation so successful. We reassemble Operator Equalisation by joining these two ingredients in a hybrid method, and test it in a hard real world regression problem. The results are surprising. Operator Equalisation and the hybrid variants exhibit completely different behaviors, and an unexpected feature of Operator Equalisation is revealed, one that may be the true responsible for its success: a nearly flat length distribution target. We support this finding with additional results, and discuss its implications.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming

General Terms

Algorithms, Performance, Experimentation

Keywords

Genetic Programming, Bloat, Operator Equalisation, Crossover Bias, Brood Recombination, Dynamic Limits, Regression

1. INTRODUCTION

The most recent theory concerning bloat is the Crossover Bias theory introduced by Dignum, Poli and Langdon [5, 6, 11]. It explains code growth in tree based GP by the effect that standard subtree crossover has on the distribution of tree sizes, or program lengths, in the population. Whenever subtree crossover is applied, the amount of genetic material removed from the first parent is the exact same amount inserted in the second parent, and vice versa. The mean tree size, or mean program length, remains unchanged. However, as the population undergoes repeated crossover operations, it approaches a particular Lagrange distribution of tree sizes where small individuals are much more frequent than the larger ones [6]. For example, crossover generates a high amount of single-node individuals. Because very small individuals are generally unfit, selection tends to reject them in favor of the larger individuals, causing an increase in mean tree size. According to the theory, it is the proliferation of these small unfit individuals, perpetuated by crossover, that ultimately causes bloat. Strong theoretical and empirical evidence supports the Crossover Bias theory. It has been shown that the bias towards smaller individuals is more intense when the population mean tree size is low, and that the initial populations resembling the Lagrange distribution bloat more easily than the ones initialized with traditional methods [11]. It was also found that the usage of size limits may actually speed code growth in the early stages of the run, as it promotes the proliferation of the smaller individuals [6]. Along with further theoretical developments, it has also been shown that smaller populations bloat more slowly [14], and that elitism reduces bloat [12, 13].

Inspired by the Crossover Bias theory, Operator Equalisation [7, 17] is the most recent and successful bloat control method available today. It can bias the population towards a desired program length distribution by accepting or rejecting each newly created individual into the population. Operator Equalisation can easily avoid the small unfit individuals resulting from the crossover bias, as well as the excessively large individuals that are no better than the smaller ones already found. Preventing the larger individuals from entering the population is a common bloat control practice; preventing the smaller ones is not, however it has been done non explicitly. We revisit two bloat control methods, the old Brood Recombination [21] and the newer Dynamic Limits [16], hypothesizing that together they contain these two key ingredients that seem to make Operator Equalisation so successful. We reassemble Operator Equalisation by joining them in a hybrid method, and test it in a hard real world regression problem, revealing surprising results.

In the next section we describe Operator Equalisation with some detail. Sections 3 and 4 describe Brood Recombination and Dynamic Limits, explaining why they contain the necessary ingredients to reassemble Operator Equalisation. Section 5 describes the data, techniques and parameters used for the experiments, while Section 6 reports and discusses all the results obtained. Finally, Section 7 summarizes and draws conclusions, also suggesting future work.

2. OPERATOR EQUALISATION

Developed alongside the Crossover Bias theory (see Section 1), Operator Equalisation is a recent technique to control bloat that allows an accurate control of the program length distribution inside a population during a GP run. Already used a number of times in benchmark and real world problems (e.g. [18–20, 22]), it is however still fairly new, so we provide a detailed explanation of how it works.

2.1 Distribution of program lengths

We use the concept of a histogram. Each bar of the histogram can be imagined as a bin containing those programs (individuals, solutions) whose length is within a certain interval. The width of the bar determines the range of lengths that fall into this bin, and the height specifies the number of programs allowed within. We call the former *bin width* and the latter *bin capacity*. All bins are the same width, placed adjacently with no overlapping. Each length value, l, belongs to one and only one bin b, identified as follows:

$$b = \left\lfloor \frac{l-1}{bin\_width} \right\rfloor + 1 \qquad (1)$$

For instance, if $bin\_width = 5$, bin 1 will hold programs of lengths 1,..,5, bin 2 will hold programs of lengths 6,..,10, etc. The set of bins represents the distribution of program lengths in the population.

Operator Equalisation biases the population towards a desired target distribution by accepting or rejecting each newly created individual into the population (and into its corresponding bin). The original idea of Operator Equalisation [7], where the user was required to specify the target distribution and maximum program length, rapidly evolved to a self adapting implementation [17] we here designate as OpEq, where both these elements are automatically set and dynamically updated to provide the best setting for each stage of the evolutionary process. Other developments of Operator Equalisation were also made [18] but we do not use them here.

There are two tasks involved in OpEq: calculating the target (in practical terms, defining the capacity of each bin) and making the population follow it (making sure the individuals in the population fill the set of bins).

2.2 Calculating the Target Distribution

In OpEq the dynamic target length distribution simply follows fitness. For each bin, the average fitness of the individuals within is calculated, and the target is proportional to these values. Bins with better average fitness will have higher capacity, because that is where search is proving to be more successful. Formalizing, the capacity, or target number of individuals, for each bin b, is calculated as:

$$bin\_capacity_b = round(n \times (\bar{f}_b / \sum_i \bar{f}_i)) \qquad (2)$$

where $\bar{f}_i$ is the average fitness in the bin with index i, $\bar{f}_b$ is the average fitness of the individuals in b, and n is the number of individuals in the population. Equation 2 is used for maximization problems where higher fitness is better (so the fitness values must suffer a transformation for minimization problems, for example a sign inversion and mapping back to positive values).

Initially based on the first randomly created population, the target is updated at each generation, always based on the fitness measurements of the current population. This creates a fast moving bias towards the areas of the search space where the fittest programs are, avoiding the small unfit individuals resulting from the crossover bias, as well as the excessively large individuals that are no better than the smaller ones already found. Thus the dynamic target is capable of self adapting to any problem and any stage of the run.

2.3 Following the Target Distribution

In OpEq every newly created individual must be validated before eventually entering the population, and the ones who do not fit the target are rejected. Each offspring is created by genetic operators as in any other GP system. After that, its length is measured and its corresponding bin is identified using Equation 1. If this bin already exists and is not full (meaning that its capacity is higher than the number of individuals already there), the new individual is immediately accepted. If the bin still does not exist (meaning it lies outside the current target boundaries) the fitness of the individual is measured and, in case we are in the presence of the new best-of-run (the individual with best fitness found so far), the bin is created to accept the new individual, becoming immediately full. Any other non-existing bins between the new bin and the target boundaries also become available with capacity for only one individual each. The dynamic creation of new bins frees OpEq from the fixed maximum program length that was present in the original idea. The criterion of creating new bins whenever needed to accommodate the new best-of-run individual is inspired by the Dynamic Limits bloat control technique [16].

When the intended bin exists but is already at its full capacity, or when the intended bin does not exist and the new individual is not the best-of-run, the individual is evaluated and, if we are in the presence of the new best-of-bin (meaning the individual has better fitness than any other already in that bin), the bin is forced to increase its capacity and accept the individual. Otherwise, the individual is rejected. Permitting the addition of individuals beyond the bin capacity allows a clever overriding of the target distribution, by further biasing the population towards the lengths where the search is having a higher degree of success. In the second case, when the bin does not exist and the individual is not the best-of-run, rejection always occurs.

3. BROOD RECOMBINATION

Brood Recombination, also called Greedy Recombination, was popularized by Tackett in 1994 [21] as a new recombination operator to serve as a substitute for the standard subtree crossover. Instead of recombining two parents once to produce one pair of offspring, Brood Recombination recombines two parents n times, each time selecting different crossover points, to produce n pairs of offspring, where n is called the *brood size factor*. Then only two offspring are selected, the best of the brood, and the rest discarded. This idea was originally introduced by Altenberg as Soft Brood Selection [1], to which Tackett added the use of a reduced-cost fitness evaluation for members of the brood. The primary motivation for developing Brood Recombination was to improve the efficiency of GP systems:

> *"The fitness evaluation of brood members is performed with a 'culling function' which is a fractional subset of the fitness evaluation function for full-fledged population members. A significant result is that large reductions in the cost of the culling function produce small performance degradation of the population members."* [21]

The secondary motivation was to reduce bloat, based on the early and long lasting theory that bloat emerges as a protection against the destructive effects of crossover (e.g. [1, 4, 8, 10], for a review of bloat theories see [16]). But Tackett refutes this theory based on the fact that Brood Recombination, being a much less destructive recombination operator, was not able to reduce code growth. However, according to the Crossover Bias theory, Brood Recombination should help control bloat. In practical terms, creating several pairs of offspring and then choosing only the best may reduce the crossover bias to create many small individuals. If it is verified that the smaller offspring are indeed the most unfit, they will not be selected from among the brood members, and not introduced into the population. The larger the brood, the larger the reduction of bias.

Therefore, we designate Brood Recombination as the first key element for assembling a hybrid method that recreates the successful behavior of Operator Equalisation. We do not, however, use the "culling function" for brood member selection, instead using the same fitness function used for full-fledged population members. This means we are in fact using the original Soft Brood Selection [1], however we decide to keep the most popular name of Brood Recombination. We also introduce a variant of Brood Recombination which we call Batch Recombination. The only difference is that, instead of repeatedly selecting two offspring from the $2n$ brood members produced by each single couple, all the offspring needed to form a new generation are now selected only once from among the several broods produced by all the couples. This should reduce the crossover bias ever further.

4. DYNAMIC LIMITS

Tree-based GP traditionally uses a static depth limit to avoid excessive growth of its individuals. When an individual is created that violates this limit, one of its parents is chosen for the new generation instead [9].

> *"This effectively avoids the growth of trees beyond a certain point, but it does nothing to control bloat until the limit is reached. The static nature of the limit may also prevent the optimal solution to be found for problems of unsuspected high complexity."* [16]

These unsolved problems lead Silva et al. to create a bloat control technique called Dynamic Maximum Tree Depth [15, 16]. It also imposes a depth limit on the individuals accepted into the population, but this one is dynamic, meaning that it can be changed during the run. The dynamic limit is initially set with a low value, usually the same as the maximum depth of the initial random trees. Any new individual who breaks this limit is rejected and replaced by one of its parents (as with the traditional static limit), unless it is the best individual found so far. In this case, the dynamic limit is raised to match the depth of the new best-of-run and allow it into the population. Dynamic Maximum Tree Depth can coexist with the traditional depth limit.

First published in 2003 [15], the original Dynamic Maximum Tree Depth was then extended to include two variants: a heavy dynamic limit, called heavy because it falls back to lower values whenever allowed, and a dynamic limit on size instead of depth. The entire concept has later been collectively designated as Dynamic Limits [16]. The heavy limit is one that accompanies the depth of the best individual, up or down, with the sole constraint of not going lower than its initialization value; a very heavy option allows it to fall back even below its initialization value. As expected, whenever the limit falls back to a lower value, some individuals already in the population immediately break the new limit. These are allowed to remain in the population but, when breeding, the limit that applies to their children is the depth of the deepest parent. The second variation is the usage of a dynamic size limit, where size is the number of nodes of the tree. The dynamic size limit also includes a modified version of the Ramped Half-and-Half initialization procedure that replaces the concept of depth with the concept of size.

Since Operator Equalisation itself was inspired by the Dynamic Limits for the decision on when to open new bins (see Section 2.3), it is only natural to assume that this is the second key element for its success. Although a size limit would probably mimic the decisions made by Operator Equalisation more accurately (because they are based on solution length, which is exactly the same thing), this variant was never as successful as using depth [16] and has the additional burden of a modified initialization procedure, so we decided to use the dynamic limit on depth. We have, however, chosen the very heavy option that allows the limit to fall back as much as possible, since in Operator Equalisation it is also possible to eliminate any bins from the target, in case they remain empty.

5. EXPERIMENTS

To perform our experiments we chose to use the first real world problem that was tackled by Operator Equalisation, the prediction of the human oral bioavailability of a set of candidate drug compounds on the basis of their molecular structure [18,19]. We briefly describe the problem and then specify how Operator Equalisation was reassembled using Brood Recombination and Dynamic Limits, specifying the techniques and parameters used in the experiments.

5.1 Test Problem

Human oral bioavailability is the pharmacokinetic parameter that measures the percentage of the initial orally submitted drug dose that effectively reaches the systemic blood circulation after passing through the liver. This parameter is particularly relevant in the drug discovery process, and this problem has already been approached by several machine learning methods, with GP providing the best results so far [2, 3].

We have used the same dataset as [18, 19], which is freely available[1]. The dataset consists of a matrix composed by 260 rows and 242 columns, where each row is a vector of molecular descriptors of a particular drug, and each column represents a molecular descriptor, except the last one that contains the known target values of the bioavailability parameter.

Following [18, 19], from this dataset training and test sets were obtained by random splitting: at each different run, 70% of the molecules were randomly selected with uniform probability and inserted into the training set, while the remaining 30% were used for the test set.

5.2 Techniques and Parameters

To reassemble Operator Equalisation we began by implementing a Standard GP system (StdGP). Then we joined Brood Recombination (Brood), alternatively Batch Recombination (Batch), using different Brood/Batch sizes (2,5,10). To assess how much the Brood/Batch Recombination differs from Standard GP, and how much it pushes the behavior towards Operator Equalisation (OpEq), we compared all of them to each other.

On a second phase we joined Dynamic Limits (Dyn) to all the previous variants to create the hybrid techniques, and once again performed comparisons among them, to assess how much they are able to approximate the behavior of OpEq. Finally we implemented OpEq with a flat target distribution (FlatOpEq) to verify some of our results.

Table 1 shows the numbers and acronyms of the 16 different techniques used. Some of the plots of Section 6 use the numbers for lack of space for the acronyms. Table 2 shows the parameter settings common to all the techniques. Regarding the parameters specific to each technique, both Operator Equalisation techniques use a bin width of 1, and none uses the maximum depth limit.

[1]http://personal.disco.unimib.it/Vanneschi/bioavailability.txt

Table 1: Numbers and acronyms of the 16 techniques used.

Number	Acronym	Number	Acronym
1	StdGP	9	DynBrood2
2	DynStdGP	10	DynBrood5
3	Brood2	11	DynBrood10
4	Brood5	12	DynBatch2
5	Brood10	13	DynBatch5
6	Batch2	14	DynBatch10
7	Batch5	15	OpEq
8	Batch10	16	FlatOpEq

Table 2: Parameter settings common to all techniques.

Parameter	Setting
Number of runs	30
Population size	500
Function and terminal sets	$\{+,-,*,/\}$, $\{x_1,...,x_{241}\}$
Tree initialization and growth	ramped max depth 6, limit 17
Fitness function	root mean squared error
Selection for reproduction	lexicographic tournament, size 10
Replication rate	0.1
Genetic operators	crossover 0.9, mutation 0.1
Selection for survival	non elitist, replace all
Stop criterion	50 generations

6. RESULTS AND DISCUSSION

Some of the plots presented in this section are still somewhat unconventional. They plot the evolution of fitness against length of the solution, completely disregarding generations, evaluations, or time spent in the search process. These plots have been first used by Silva and Dignum [17] and we consider them to be an intuitive way of visualizing the bloating behavior of any given technique.

Indeed, we are not interested in measuring the performance of the techniques in terms of how much computational effort is required to achieve a given fitness. Operator Equalisation is recognized to be inefficient, an issue discussed at length in [17], however it can find solutions with a fitness/length ratio that other techniques do not seem to be able to reach. In the real world this is usually one of the most important quality factor of a solution, regardless of the more or less lengthy search process that ultimately found it. Therefore, we also present a few plots showing the evolution of length along the generations, knowing perfectly well that one generation represents enormously different computational efforts to different techniques.

Although the issue of overfitting is not central to this work, we are using a real world problem where the generalization ability is important. Therefore, we present some results obtained in the test set, to show that none of the modified techniques suffers from a decreased generalization ability that would prevent it from being successfully used in the real world.

Finally, in most plots and related text we do not discriminate between the different Brood/Batch sizes except when we consider the differences to be important.

6.1 Comparing fitness and solution length

Figure 1a shows the best training fitness plotted against the average length of the solutions in the population, for Standard GP, the different sizes of Brood/Batch Recombination, and Operator Equalisation. Figure 1b is similar to 1a except that instead of plotting the best training fitness, it plots the test fitness of the best training individual. It is immediately apparent that, although some variants of Brood/Batch Recombination exhibit a more desirable bloating behavior (fitness/length ratio) than Standard GP, most of the differences seem to be caused by the simple fact that producing more offspring allows for more search, since the general trend is the same. Also Operator Equalisation is allowed more search due to the number of rejected individuals, however its behavior is not even remotely approximated by any of the Brood/Batch variants. The stabilization of average solution length exhibited by Operator Equalisation had already been observed in other real world problems (e.g. [20, 22]). Figure 1c shows the evolution of the average solution length plotted against the generations, where the differences in code growth are more easily observed. For a visualization of the distribution of training and test fitness and average solution length in the final generation consult Figure 5.

Figure 2 is analogous to 1 except that it refers to Standard GP and Brood/Batch Recombination using Dynamic Limits. Like Brood/Batch Recombination, Dynamic Limits is a somewhat helpful modification by itself (see Figure 4 for a direct comparison), but the effect of joining both elements is not cumulative, and once again Operator Equalisation remains a far better technique in terms of bloating behavior. The usage of Dynamic Limits in Brood/Batch Recombination causes larger differences in the behavior of different Brood/Batch sizes, with size 2 (DynBrood2 and DynBatch2) resulting in less code growth and less learning (see also Figure 5), which may simply be a result of less search.

6.2 Exploring the length distributions

Given the previous plots we are forced to conclude that the hybrid techniques using Brood/Batch Recombination and Dynamic Limits do not contain the same ingredients as Operator Equalisation. By design, the hybrid techniques should emulate the decisions of Operator Equalisation on whether to accept or reject the individuals (large or small) that fall outside the limits of the target, so the difference must lie within the target itself. Therefore, we now focus our attention on the distribution of solution lengths during the evolution, looking for an explanation to the unique behavior of Operator Equalisation. In principle, given the same limits the solution lengths should follow similar distributions in all the techniques, since Operator Equalisation enforces a distribution that is "proportional" to fitness (see Section 2.2), which is exactly what selection is supposed to do.

Figure 3 contains some actual and target length distributions of different techniques. The three plots in the first row (a,b,c) show typical length distributions obtained by Standard GP, Dynamic Brood of size 2, and Operator Equalisation. The height of the peaks is not important for this discussion. DynBrood2 was chosen for being the hybrid technique with the lowest expected difference to Standard GP, and it is interesting to compare the length distributions of both. There seems to be no substantial difference in terms of the larger individuals, at least nothing typical was evident among the 20 runs. However, in terms of the smaller individuals the effect of Brood Recombination can be clearly seen. While Standard GP keeps producing (and consequently accepting) small individuals long after the distribution is centered on larger lengths, Brood Recombination has the ability to prevent them from entering the population, exhibiting an almost clean cut between zero frequency and high frequency. This effect is stronger as the brood size increases, and further so when using Batch instead of Brood (not shown). We naturally assume that, given the similarity of both distributions, like Standard GP Brood Recombination is also producing small individuals. Therefore we conclude that they are indeed the most unfit ones and are thus rejected in favor of the best of the brood. According to the Crossover Bias theory (see Section 1),

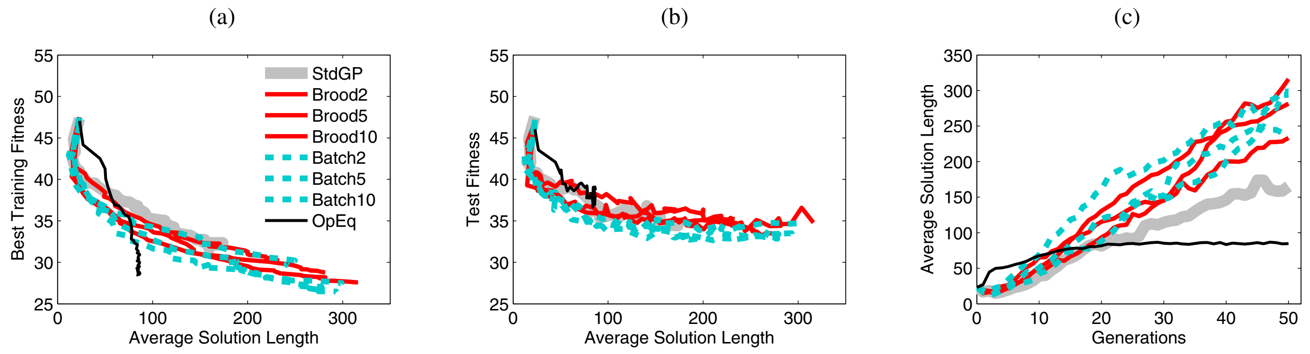

Figure 1: Standard GP versus Brood/Batch Recombination versus Operator Equalisation. (a) Best training fitness versus average solution length; (b) Test fitness (of the best training individual) versus average solution length; (c) Average solution length versus generations.

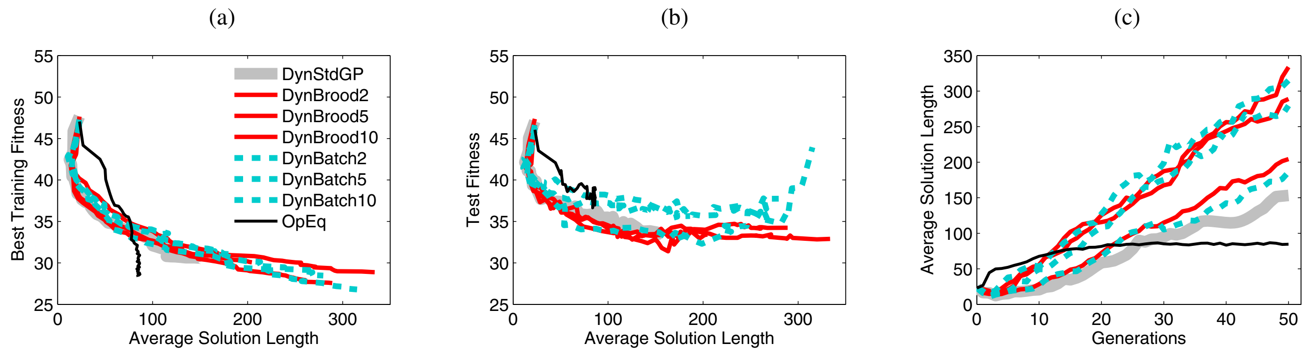

Figure 2: Dynamic Standard GP versus Dynamic Brood/Batch Recombination versus Operator Equalisation. (a) Best training fitness versus average solution length; (b) Test fitness (of the best training individual) versus average solution length; (c) Average solution length versus generations.

this should act against bloat, but our results show only a minimal effect, at least when compared to Operator Equalisation.

The length distribution of Operator Equalisation (Figure 3c) is completely different from the previous ones. Instead of showing a clear preference for given lengths, this distribution spreads the individuals across most of the available bins, including the bins of the smaller lengths that presumably contain the worst individuals. This surprising "flatness" was found to be the main characteristic of all the 20 distributions of Operator Equalisation. Further investigation revealed that these distributions are the result of following targets that are mostly uniform, a truly unexpected finding. The plot of Figure 3d shows the target that originated the distribution shown on plot c. Had the target been faithfully followed, the actual distribution would be even flatter, but the several rejections and overrides gave it a less artificial look.

The explanation for such a flat uniform target became clear after realizing that the diversity and amplitude of fitness values that occur in this real world problem almost guarantees the presence of very unfit outliers in the population, practically every generation. These fitness values are so much worse than the others that, compared to them, all the rest looks the same, and all the bins end up getting the same capacity. Removing these outliers before calculating the target may prove to be a difficult task, since once we remove the first lot others will appear in the new distribution. An obvious improvement is to use the best or median fitness of each bin, instead of the average, to calculate the target, but even so the problem persists when some bins contain only very unfit individuals.

But is this really a problem that needs to be solved? Certainly an unintended feature, but also the most probable reason why Operator Equalisation exhibits a behavior so different and so much better than all other techniques, at least where bloat is concerned. No matter where the best individuals are found, Operator Equalisation maintains an almost uniform search across the entire set of explored lengths, thus increasing the probability of finding smaller solutions. In the limit, the number of individuals in the population will not be enough to ensure one individual per bin, but let us concentrate on the most immediate issues for now.

6.3 Enforcing a flat target

Assuming the nearly flat length distribution target is the true responsible for the success of OpEq in this symbolic regression problem, we wonder what improvements we can achieve if we enforce a truly flat distribution. So we implemented an Operator Equalisation variant that does not calculate the capacity of the bins with Equation 2, but instead gives the same capacity to all of them. All the rest, in particular the decision to create new bins, did not suffer any changes. Note that for most problems the actual length distribution is never exactly equal to the target, because when bins get full the target begins to be overridden. In our particular problem this is aggravated by the fact that all the arithmetic operators in the function set are binary, making it impossible to create solutions of even length. This means that half of the bins of our perfectly flat target are never filled, and half or the individuals of the population are guaranteed to override the target. In fact, because of this limita-

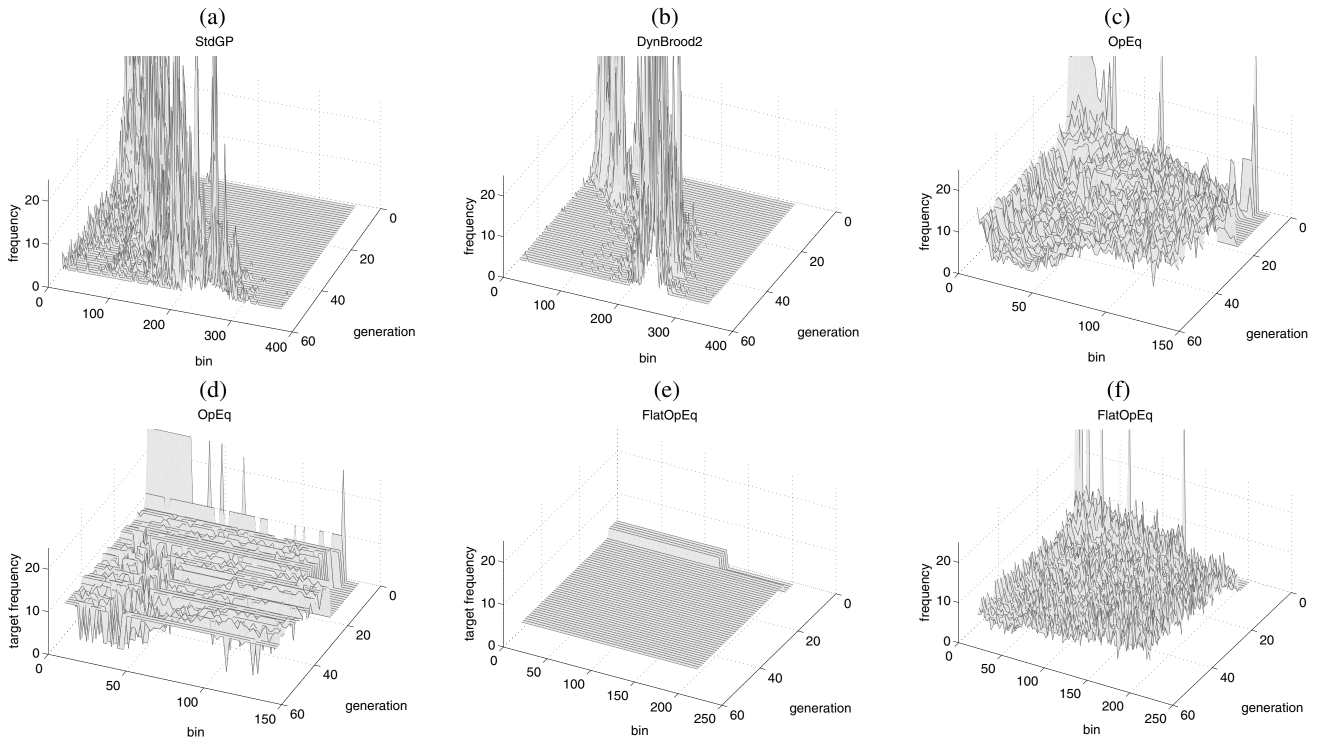

Figure 3: Examples of actual and target length distributions. (a) Typical length distribution of Standard GP; (b) Typical length distribution of DynBrood2; (c) Typical length distribution of Operator Equalisation; (d) Target length distribution that originated the distribution in plot c; (e) Typical target length distribution of Operator Equalisation with flat target; (f) Length distribution originated by the target in plot e. All frequencies above 25 are not shown.

tion caused by the set of exclusively binary functions, to facilitate the visualization none of the plots of Figure 3 shows the bins of even length.

Figure 3e shows a typical target distribution of the new variant of Operator Equalisation, that we call FlatOpEq, while Figure 3f shows the actual distribution obtained by using this target. It is not completely flat for the reasons stated above, but it is typically much flatter than the actual distributions of OpEq, like the one in Figure 3c. Next, we compare FlatOpEq with the remaining techniques, in terms of fitness and solution length.

Figure 4 shows a direct comparison between Standard GP with and without Dynamic Limits, Operator Equalisation with and without flat target, and another choice of a Brood/Batch technique, in this case Batch5 for being the one with the more desirable bloating behavior among all the Brood/Batch variants, with or without Dynamic Limits. Figure 5 shows boxplots of the training and test fitness values, and the solution lengths, obtained in the final generations of the 20 runs. The FlatOpEq technique can reach significantly better fitness values than all other techniques (determined by non-parametric ANOVA with $p = 0.05$), but once again the explanation may simply be that more rejections mean more search, hence more learning. In terms of test fitness there are no statistically significant differences (Figure 5b), however it is worth noting that somewhere along the evolution FlatOpEq is able to reach better test fitness than the other techniques, before overfitting occurs (Figure 4b). The average solution length is basically the same for both Operator Equalisation variants at the end of the run, for despite doing more search FlatOpEq stabilizes the average solution length at around the same values as OpEq.

All comparisons made, there seems to be no disadvantage in artificially flattening the target distribution of Operator Equalisation.

7. CONCLUSIONS AND FUTURE WORK

We have hypothesized that the two key ingredients that make Operator Equalisation such a successful bloat control method can be found in older methods such as Brood Recombination and Dynamic Limits. With Brood Recombination, the usually very unfit small individuals frequently produced by the parents are rejected in favor of the best of the brood. According to the Crossover Bias theory, eliminating the bias to introduce small unfit individuals in the population helps control bloat. With Dynamic Limits, the individuals larger than any others already found in the population are only accepted if they prove to be the best ever found during the run. This prevents unnecessarily large individuals to enter the population, thus controlling bloat.

We reassembled Operator Equalisation by taking a Standard GP system and coupling it with these two ingredients, obtaining a hybrid method which we tested in a hard real world regression problem. None of the several variants tested was able to produce a bloating behavior remotely similar to the one of Operator Equalisation. We took a deeper look at the dynamics of the search and found that, for previously unsuspected reasons, the target length distribution used by Operator Equalisation is typically nearly flat, contrasting with the peaky and well delimited targets of all the other approaches. Finally we introduced a new Operator Equalisation variant that enforces an artificially created flat target, and verified that the results were even better than the previous version.

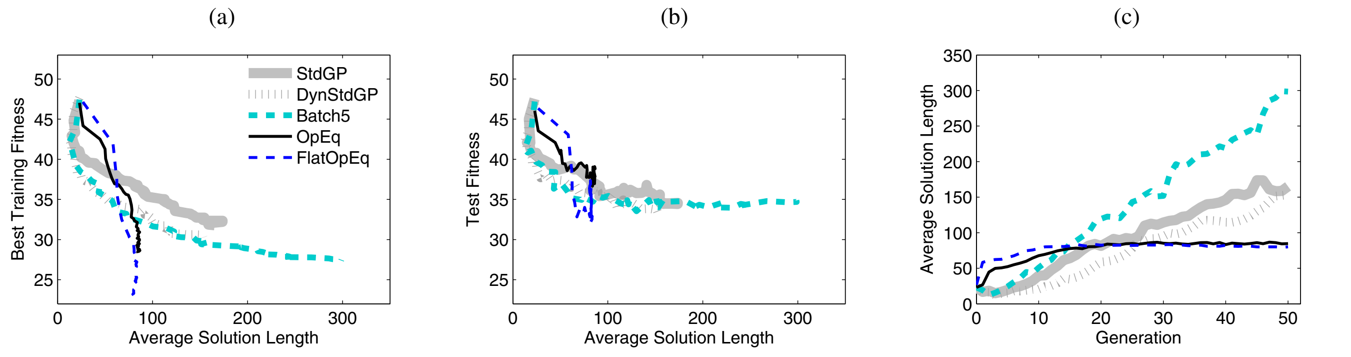

Figure 4: Standard GP with and without Dynamic Limits versus one of the Brood/Batch techniques (Batch5) versus Operator Equalisation with and without flat target. (a) Best training fitness versus average solution length; (b) Test fitness (of the best training individual) versus average solution length; (c) Average solution length versus generations.

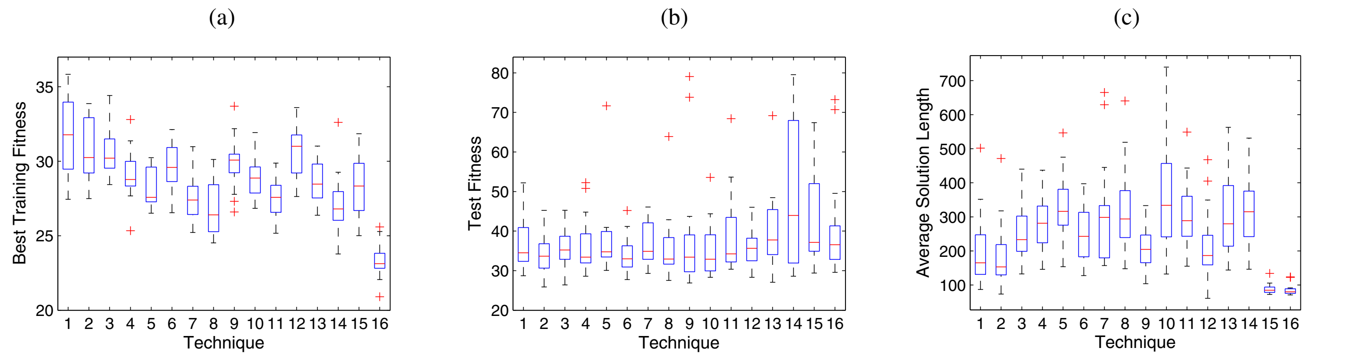

Figure 5: Boxplots of all the 16 techniques used (see Table 1 for their acronyms). (a) Best training fitness; (b) Test fitness (of the best training individual); (c) Average solution length. Values obtained in the last generation of each of the 20 runs. Many outliers not shown in plot b.

It seems like the flatter the target, the most success is achieved in bloat control. Instead of avoiding small unfit individuals, the flat target actually prevents the search from moving away from the shorter lengths, even long after better and larger solutions have been found. It simply spreads individuals across all the previously visited lengths, ensuring that search does not abandon any of them.

After absorbing these results it becomes quite trivial that, to avoid bloat and reach smaller solutions, we must keep searching among the shorter lengths. The success of Operator Equalisation is undeniable, but the current results force us to look back at its previous successes and check if they were simply the result of an unintended flat distribution target, or if the Crossover Bias theory actually plays a significant role in the process. We realize just now that a flat target may appear as a consequence of, not only extremely high, but also extremely low, phenotypic diversity, and the benchmark parity problems immediately come to mind as cases to check. We leave this as future work. We also intend to provide results based on some measure of computational effort, for example the number of evaluations performed, instead of the number of generations, to make the comparison between techniques more objective and fair.

We finish with the ironic remark that the original meaning of equalization was, not surprisingly, flattening the signal along the entire spectrum.

Acknowledgments

This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds. The author thanks project PTDC/EIA-CCO/103363/2008 from FCT, Portugal. Thank you also to the anonymous reviewers for all their helpful comments.

8. REFERENCES

[1] L. Altenberg. The evolution of evolvability in genetic programming. In K. E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press, 1994.

[2] F. Archetti, S. Lanzeni, E. Messina, and L. Vanneschi. Genetic programming for human oral bioavailability of drugs. In M. Keijzer, et al., editors, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 255–262, Seattle, Washington, USA, 8-12 July 2006. ACM Press.

[3] F. Archetti, S. Lanzeni, E. Messina, and L. Vanneschi. Genetic programming for computational pharmacokinetics in drug discovery and development. *Genetic Programming and Evolvable Machines*, 8(4):413–432, Dec. 2007. special issue on medical applications of Genetic and Evolutionary Computation.

[4] M. Brameier and W. Banzhaf. Neutral variations cause bloat in linear GP. In C. Ryan, et al., editors, *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 286–296, Essex, 14-16 Apr. 2003. Springer-Verlag.

[5] S. Dignum and R. Poli. Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In D. Thierens, et al., editors, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, volume 2, pages 1588–1595, London, 7-11 July 2007. ACM Press.

[6] S. Dignum and R. Poli. Crossover, sampling, bloat and the harmful effects of size limits. In M. O'Neill, et al., editors, *Proceedings of the 11th European Conference on Genetic Programming, EuroGP 2008*, volume 4971 of *Lecture Notes in Computer Science*, pages 158–169, Naples, 26-28 Mar. 2008. Springer.

[7] S. Dignum and R. Poli. Operator equalisation and bloat free GP. In M. O'Neill, et al., editors, *Proceedings of the 11th European Conference on Genetic Programming, EuroGP 2008*, volume 4971 of *Lecture Notes in Computer Science*, pages 110–121, Naples, 26-28 Mar. 2008. Springer.

[8] S. Gelly, O. Teytaud, N. Bredeche, and M. Schoenauer. Universal consistency and bloat in GP. *Revue d'Intelligence Artificielle*, 20(6):805–827, 2006. Issue on New Methods in Machine Learning. Theory and Applications.

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

[10] N. F. McPhee and J. D. Miller. Accurate replication in genetic programming. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 303–309, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[11] R. Poli, W. B. Langdon, and S. Dignum. On the limiting distribution of program sizes in tree-based genetic programming. In M. Ebner, et al., editors, *Proceedings of the 10th European Conference on Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 193–204, Valencia, Spain, 11-13 Apr. 2007. Springer.

[12] R. Poli, N. F. McPhee, and L. Vanneschi. Analysis of the effects of elitism on bloat in linear and tree-based genetic programming. In R. L. Riolo, et al., editors, *Genetic Programming Theory and Practice VI*, Genetic and Evolutionary Computation, chapter 7, pages 91–111. Springer, Ann Arbor, 15-17May 2008.

[13] R. Poli, N. F. McPhee, and L. Vanneschi. Elitism reduces bloat in genetic programming. In M. Keijzer, et al., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1343–1344, Atlanta, GA, USA, 12-16 July 2008. ACM.

[14] R. Poli, N. F. McPhee, and L. Vanneschi. The impact of population size on code growth in GP: analysis and empirical validation. In M. Keijzer, et al., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1275–1282, Atlanta, GA, USA, 12-16 July 2008. ACM.

[15] S. Silva and J. Almeida. Dynamic maximum tree depth. In E. Cantú-Paz, et al., editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1776–1787, Chicago, 12-16 July 2003. Springer-Verlag.

[16] S. Silva and E. Costa. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10(2):141–179, 2009.

[17] S. Silva and S. Dignum. Extending operator equalisation: Fitness based self adaptive length distribution for bloat free GP. In L. Vanneschi, et al., editors, *Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009*, volume 5481 of *LNCS*, pages 159–170, Tuebingen, Apr. 15-17 2009. Springer.

[18] S. Silva and L. Vanneschi. Operator equalisation, bloat and overfitting: a study on human oral bioavailability prediction. In G. Raidl, et al., editors, *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1115–1122, Montreal, 8-12 July 2009. ACM.

[19] S. Silva and L. Vanneschi. Bloat free genetic programming: Application to human oral bioavailability prediction. *International Journal of Data Mining and Bioinformatics*, to appear.

[20] S. Silva, M. Vasconcelos, and J. Melo. Bloat free genetic programming versus classification trees for identification of burned areas in satellite imagery. In C. Di Chio, et al., editors, *Applications of Evolutionary Computation: EvoApplications 2010: Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP)*, volume 6024 of *LNCS*, Istanbul, 7-9 Apr. 2010. Springer.

[21] W. A. Tackett. *Recombination, Selection, and the Genetic Construction of Computer Programs*. PhD thesis, University of Southern California, Department of Electrical Engineering Systems, USA, 1994.

[22] L. Vanneschi and S. Silva. Using operator equalisation for prediction of drug toxicity with genetic programming. In L. S. Lopes, et al., editors, *Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence, EPIA 2009*, volume 5816 of *LNAI*, pages 65–76, Aveiro, Portugal, Oct. 12-15 2009. Springer.