Crossover operators to control size growth in linear GP and variable length GAs

Dominique Chu and Jonathan E. Rowe

Abstract— **In various nuances of evolutionary algorithms it has been observed that variable sized genomes exhibit large degrees of redundancy and corresponding undue growth. This phenomenon is commonly referred to as "bloat." The present contribution investigates the role of crossover operators as the cause for length changes in variable length genetic algorithms and linear GP. Three crossover operators are defined; each is tested with three different fitness functions. The aim of this article is to indicate suitable designs of crossover operators that allow efficient exploration of designs of solutions of a wide variety of sizes, while at the same time avoiding bloat.**

I. Introduction

A common problem of variable sized genomes is that solutions tend to get infested with non-functional parts that hitchhike with fit solutions, a phenomenon commonly known as *bloat*. Bloat might lead to substantial increases of the genome size relative to what would actually be required.

This phenomenon is well known from many varieties of evolutionary computation and has received particular attention in the field of genetic programming (see for example Langdon *et al.*[11] or Banzhaf and Langdon[1]). In GP the most common method to limit bloat is so-called *maximal depth restriction*[10]; basically this method sets a limit to the allowed maximal depth of individual candidate solutions. A number of variants of this method exist. A problem common to all approaches that in some way cap the size of individuals is that the best solutions might require encodings that are longer than the chosen cap-size. Another possibility is to allow bigger individuals as long as their increased size is somehow counterbalanced by an increased fitness; one method that achieves this is *parsimony pressure* (see for example [14], [20]). Again, the problem with these methods is that it needs to be specified precisely how fitness can counter-balance size; in the context of parsimony pressure this problem has been recognized and addressed by Luke[13]. Finally there have also been attempts to control bloat by introducing and limiting a "resource" that is required to construct candidate solutions. Then bloat is controlled by the scarcity of this resource; see for example Silva *et al.*[18], [19].

This contribution will specifically focus on the problem of bloat in the context of linear GP[3] and variable length genetic algorithms (GA). Standard GAs[6] are traditionally used with a fixed length genome. This is suitable for many optimization tasks but can also be restrictive in certain applications. Specifically in situations where the optimal size of the system to be evolved might not be known. One example where this could be the case is the evolution of chemical systems, such as for example cell signaling networks[5]. Furthermore, there is strong evidence that growth phenomena are important in the evolution of real genomes[9].

There are relatively few attempts to use variable length GAs. Harvey introduced the species adaptation genetic algorithm[8] which allowed certain variations of genome sizes in GAs; this work was recently further developed by Bull[4]. Other applications include Grefenstette *et al* who constructed a GA to learn tactical decision rules[7], Wu and Garibay introduced the "Proportional Genetic Algorithm"[21]; this is a more biologically motivated version of classical GAs that uses explicit genes to encode information about the genome.

As far as bloat reduction in linear GP and variable length GAs is concerned, similar concerns apply as described above: Limiting the maximum allowed genome length excludes all (potentially very good) solutions beyond the chosen limit. Applying a fitness penalty to the genome size introduces an additional arbitrary parameter. This might again lead to good (but long) solutions being missed if this parameter is not set correctly. A second possibility is to reduce redundancy at run-time by removing junk-entries in the genomes. The problem here is that it might not be obvious whether or not a particular part of the genome is actually redundant or not. Precisely how feasible this approach is will depend on the specific circumstances of the application.

Instead of focusing on various ways to *control* bloat once it arises, this article will concentrate on the designs of crossover operators (for variable length GAs and linear GP) that minimize its occurrence in the first place, more specifically the design of the cross-over operator: With respect to the cross-over operator, the main difference between fixed length GAs and variable length GAs/linear GP is that in the latter the operator needs to be defined for chromosomes of un-equal length. Precisely how the cross-over points are chosen will determine the possible range of the length of the offspring. In general there are two conflicting requirements that need to be satisfied by the cross-over operator. Firstly, as stated above, the operator should be designed in such a way as to minimize bloat. This requirement would be best satisfied by a fixed-length representation. Hence, the second requirement is that the GA/linear GP has some mechanism to explore solutions from a range of different sizes. Again, if this second requirement is given too much weight, then code bloat is the inevitable result. The ideal operator would enable exploration without causing extensive code bloat.

D. Chu is with the Computing Laboratory, University of Kent, UK (email: D.F.Chu@kent.ac.uk); J. Rowe is with the School of Computer Science, University of Birmingham, UK (email: J.E.Rowe@cs.bham.ac.uk).

In this article we will test a number of crossover operators with respect to these formulated criteria. The aim is to find a design that strikes a good balance between exploration and code bloat. The focus of this paper on the crossover operator is not meant to imply that careful design of the crossover operator can/should be the only way to reduce bloat in variable length GAs/linear GP. Also crossover may not be the only source of size variation in a specific variable length GA/linear GP. Such additional sources of length variation and their impact on bloat will then need to be considered separately. This article will ignore any additional such sources and concentrate on investigating the effects of possible crossover operators on bloat.

Previous work on bloat in variable length GA/linear GP includes theoretical predictions of the length distribution of genomes. Rowe and McPhee [17] considered an infinite population model and a flat fitness function. For various operators they derive the limiting length distribution of the population. While the results of this work are exact, they assume very simplified scenarios.

The present article is organized as follows: In section II the 3 fitness functions that are used as test functions are defined; each of these functions has different characteristics in terms of the length of solutions it favors; furthermore none of the functions used here represents particularly hard problems. This allows the GA to find the optimal solution quickly thus providing a good basis for comparing the tendency for bloat among the optimal solutions. These operators are also introduced in section II.

Section III presents the main results of our simulations. Since the chosen fitness functions are relatively easy to solve, in all experiments discussed below the optimal solution has been found within very short time (except for one case; see below). Hence, what varies throughout the simulations is not the fitness of the solutions but rather the length of the genomes as the evolutionary system explores neutral mutants of the optimal solutions. Throughout this contribution we will therefore concentrate on this aspect of genome length rather than on the fitness of the solution.

What is absent from all experiments below is mutation. In all simulations the mutation rate was set to zero. The reason for neglecting mutations is that they introduce a number of second order effects. Our studies have shown that these do lead to interesting effects; yet it is not clear how these effects are to be interpreted in a more general context. We decided therefore not to include the role of mutations into this report.

II. Description of the model

The model is a simple implementation of a variable length GA. In all the experiments reported here we used a population size of 1000 and a tournament selection with tournament size 10. In all simulations we performed 5 million tournaments. For practical reasons it was necessary to set an upper limit for the length of the genome; this was necessary in order to prevent the occurrence of too large genomes that would exhaust the available computational resources. This limit was kept constant at 200000 for all experiments. The population was initialized with random strings of 1's and 0's. The initial length of strings was randomly chosen between 3 and 2000. The figures illustrating the changes of genome length over time were produced as follows: At every 1000 time steps the sizes of all genomes in the population at this time were recorded. This resulted in a data file with 5 million entries. The figures in this article were produced by plotting every tenth point in these files. This reduction did not qualitatively change the graphs but substantially reduced the computational resources needed to produce and handle the relevant figure files.

A. *Fitness functions*

We experimented with three different fitness functions. The first fitness-function, ff_1, was inspired by the Ising model. If s_i is the i-th entry on the genome string of length L, then ff_1 is

$$ff_1 = -\sum_{i}^{L-1} \frac{diff(s_i, s_{i+1})}{L-1}$$

$$diff(x,y) = \begin{cases} 0 & \text{if } x \neq y \\ 1 & \text{otherwise} \end{cases}$$

Note that we follow the convention that small fitness values are better than high fitness values.

In the case of ff_1 the fitness is independent of the length, in the sense that the fitness contribution is averaged over the length of the genome. Hence, this fitness function has no inherent length bias. The maximum fitness can be reached by candidate solutions of all allowed sizes. Any solution longer than 2 constitutes bloat.

The second fitness function simply measures the distance of the solution from a target contents of exactly n 1's in the genome. In this report n was chosen to be 18. Thus a string entirely consisting of zeros would have a fitness of 18, whereas a string that is 100 long with exactly 18 1's would have the optimal fitness of 0. The second fitness function is:

$$ff_2 = \sqrt{\left(\sum_{i}^{L} s_i - 18\right)^2}$$

Other than ff_1 this fitness function does have a certain length bias. There is a minimum length (here: 18) which is required for a solution to obtain the maximal fitness. Once this maximum fitness is reached, there is no necessity for solutions to grow any further in size, as it will not lead to better fitness. If there are candidate solutions that are substantially longer than 18, then this must be regarded as an indication for bloat.

Finally the last fitness-function ff_3 that will be investigated here equates fitness with the number of leading 1's multiplied by -1. For example the string `0111111111` has a fitness of zero, whereas `11011111011` has a fitness of -2 (which is better). Unlike the first fitness function, in ff_3 the fitness is not normalized by length, so there is a strong length bias in the sense that the length of a solution sets a hard limit for the best possible fitness value it can obtain.

So, an increase of length beyond a certain limit would not be an indication of bloat; instead one could take the value of $ff_3(L) + L$ as an indication of bloat; whenever it is greater than zero, then bloat is present. As it will turn out, bloat, however is not an issue with ff_3. Instead, in the present context experiments with this fitness-function will be taken as an indicator whether or not a given crossover operator can efficiently explore solutions of various lengths to find the optimal solution (given the imposed upper limit of the length of the genomes, the optimal solution would be of length 200000).

In summary, we have chosen three different fitness functions each with a different length bias and thus corresponding potentials to exhibit bloat.

B. Crossover operators

We will now describe the crossover operators used. If the crossover points on both parents are chosen randomly then the length of the offspring will typically be different to the length of both parents. Precisely how different the length of the offspring is (in a statistical sense) will determine the tendency of the corresponding operator to cause bloat. In this article three different crossover operators are considered.

The first operator O_1 is a straightforward extension of the fixed length case. For each of the two parent strings p_1 and p_2 the crossover point is chosen between 1 and the length of the string $L(p_x)$; denote the respective crossover points by p_{1,n_1} and p_{2,n_2}. The offspring will then be the new string composed of the left part of p_1 and the right part of p_2: $p_{\text{off}} = p_{1,1} \cdot p_{1,2} \cdots p_{1,n_1} \cdot p_{2,n_2} \cdot p_{2,(n_2+1)} \cdots p_{2,L(p_2)}$.

One property of O_1 is that the offspring p_{off} might substantially differ in length from its parents. While some variation in length is desirable in variable length GAs/linear GP, too much of it may not be. An alternative crossover operator, O_2, works according to the same principle as O_1 but the choice of the crossover points is constrained so that the length of the offspring does not differ from $L(p_1)$ by more than a fixed number; in all experiments reported here this number was kept fixed at 10. The third crossover operator, O_3, works in the same way as O_2 but the difference between $L(p_1)$ and $L(p_{\text{off}})$ may be up to 10 percent of p_1.

III. Results

A. Flat fitness

We performed a number of simulations to understand the behavior of the variable length GA. In what follows we are primarily interested in the lengths of the solution rather than in their fitness. In order to understand the inherent biases of the crossover operators we performed a number of simulations with a flat fitness function (that is all genomes have equal fitness).

Figure 1 & 2 show example-runs for the time evolution of the GA in a flat fitness-landscape under the three crossover operators. Under the operator O_1 the system shows strong quantitative variations both between runs and over the course of a single simulation, in the sense that the mean

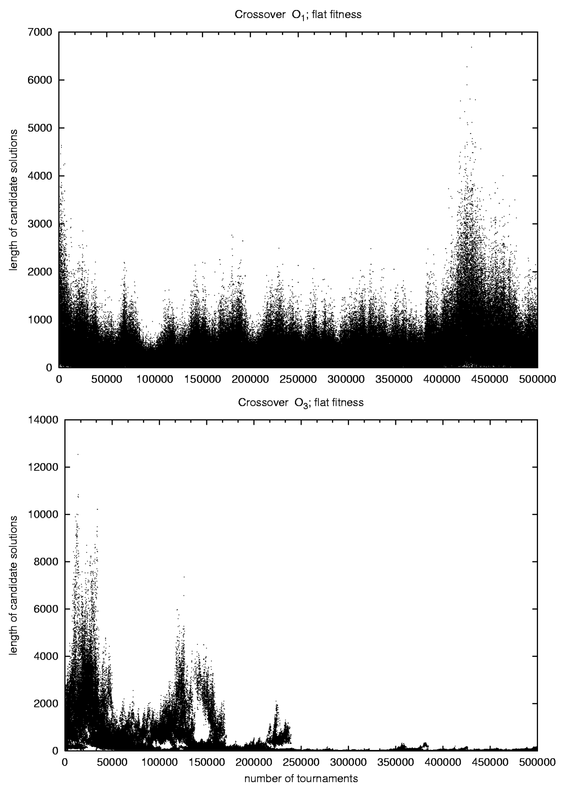

Fig. 1. *Flat fitness-function:* Time evolution of the length of candidate solutions under the crossover operators O_1 (top) and O_3 (bottom). Snapshots of the entire population are plotted at regular intervals; see main text for a precise explanation. The x-axis represents time (or more precise the number of tournaments) and the y-axis is the length of the candidate solutions. The figures represent single runs. The plot shows that the operator O_1 causes strong variations of genome sizes over time without any apparent trend to settle on a specific length. In contrast, operator O_3 settles on very short genome sizes after a transient period.

and maximum length vary strongly. Qualitatively, however, different runs are similar to one another, in the following sense: The standard deviation is typically close to the mean; for example in the particular run shown in figures 1 the mean length of genomes taken over the entire simulation is just under 380 with a standard deviation of 346. This large deviation of the actual behavior from the mean behavior is a consequence of intermittent explosions of the genome size; these are clearly visible in figure 1. Larger genome-sizes are distributed roughly exponentially (over the course of a simulation). Figure 3 shows the histogram of the distribution of the genome length in a simulation of O_1 in a flat fitness landscape (note that data plotted is taken from a different run to that in figure 1). It is in agreement with a theoretical prediction by Rowe and McPhee[17] of the behavior of an infinite population.

Crossover O_3 shows qualitatively different behavior. As shown in figure 1 (right) there are large genomes at early stages of the simulation. Over time the maximum sizes go down, although variations persist. Roughly in the second half of the run the genome sizes have substantially reduced. Closer inspection shows that in this area the largest sizes are around 200, apparently remaining stable over time from then

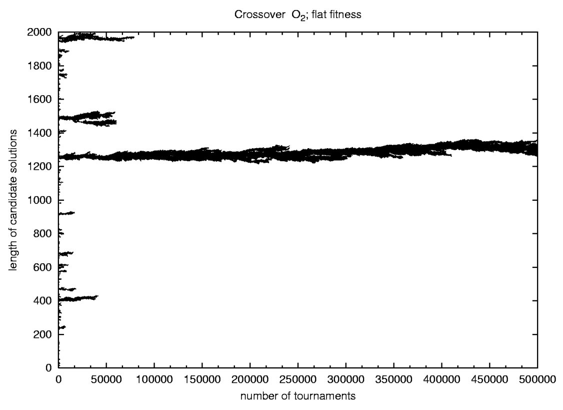

Fig. 2. *Flat fitness-function:* Time evolution of the crossover operator O_2 in a flat fitness landscape. The figures display the lengths of the entire populations recorded at regular time intervals; see main text for a precise explanation. The x-axis represents time and the y-axis is the length of the candidate solutions. The figures are single runs. The initial lengths of the genomes are chosen at random between 2 and 2000 (just about discernible in the left hand side of the plot). After a short time, most of the genome lengths have "died" out and eventually only a narrow band of possible lengths remains. This is due to the "hoovering" effect described in the main text. This qualitative effect is shown by all runs that use the same settings but the location of the band of genome lengths on which the system settles varies from run to run.

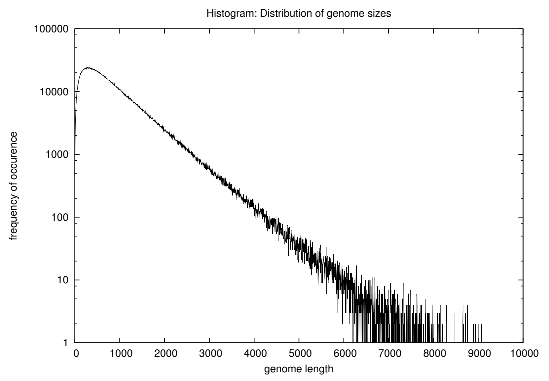

Fig. 3. Histogram of the genome length recorded in a simulation of O_1 in a flat fitness landscape; note that this is a linear-log plot. Large genome lengths are roughly exponentially distributed. This distribution is in agreement with a theoretical prediction by Rowe and McPhee.

on. Note again that figure 1 shows a single run, yet repeated simulations show the same qualitative behavior (data not shown).

The observed behavior can be explained by the properties of O_3. The possible size of the offspring is limited to be within 10 percent of the length of the parent. Hence the longer the genomes in the population the more variation one would expect; this variation can go in both directions, towards longer and shorter genomes, but once the population consists of short genomes only, the possible variations per crossover event are smaller; as a results there will be less growth in absolute terms; short genomes thus act as sinks. For the particular parameters chosen the systems eventually settles into an exponential distribution with a maximum observed genome size of about 200.

Finally, the second crossover operator shows a similar effect yet with a different outcome. One genome size "hoovers" up all others. A closer inspection of figure 2 shows that the simulation with O_2 starts with the lengths well distributed over the initially allowed range between 0 and 2000. After a short time, only 4 relatively narrow bands of genome sizes remain; after about a fifth of the simulation time all but one of them have died out and all genomes are in one single narrow size band.

This effect is explained by a process similar to size related growth. Note that independent of the length of a genome, it's offspring can always only differ from parent 1 by at most 10. Furthermore, note that the offspring created replaces a randomly chosen member of the population. After inserting the offspring into the population, the number of genomes that are within 10 of parent 1 has either stayed the same or increased by 1. Remember that parent 1 is also chosen at random. The more genomes there are in a particular size band the more likely it becomes that the next chosen parent 1 is from this band and hence the more likely it is that the number of genomes in this band grows by 1. The effect observed here is closely related to well known examples of spontaneous symmetry breaking in complex systems[2].

The width of the observed band depends on the allowed absolute length change between parent 1 and offspring. In the limit of very large allowed changes, the system would approach the behavior of operator O_1. By the same token, smaller allowed variations lead to narrower bands.

In summary, experiments with the three crossover operators show their different characteristics. Operator O_1 approaches an exponential distribution. Operator O_3 on the other hand does have a bias for shorter solutions, in the sense that once there are only short genomes in the population, genome lengths will remain short. Finally, operator O_2 has a limited ability to explore various genome sizes, particularly once the population has converged.

B. Introducing Fitness

In this sections we will describe results obtained with the three crossover operators and the three fitness functions.

The **first fitness function** does not have a strong length bias as solutions of all sizes can acquire maximal fitness. Simulations with O_1 show that the optimal fitness is found within very short time (data not shown); from then on only individuals with the optimal solution appear in the population. At early stages of the simulation only short genomes are retained. This is readily explained by the fact that long genomes are very unlikely to have uninterrupted long stretches of either only 1's or only 0's (and therefore good fitness); very short random genomes are not only more likely to have good fitness but it is also easier to improve their fitness by a few crossovers only. This is reflected by the fact that initially genome sizes are very short in the simulations with O_1 (in figure 4). In due course the GA also explores longer solutions. Due to the particular characteristics of the Ising model fitness function once solutions are found they can easily be combined via crossover to give new optimal fitness solutions. Once all sub-optimal solutions have been

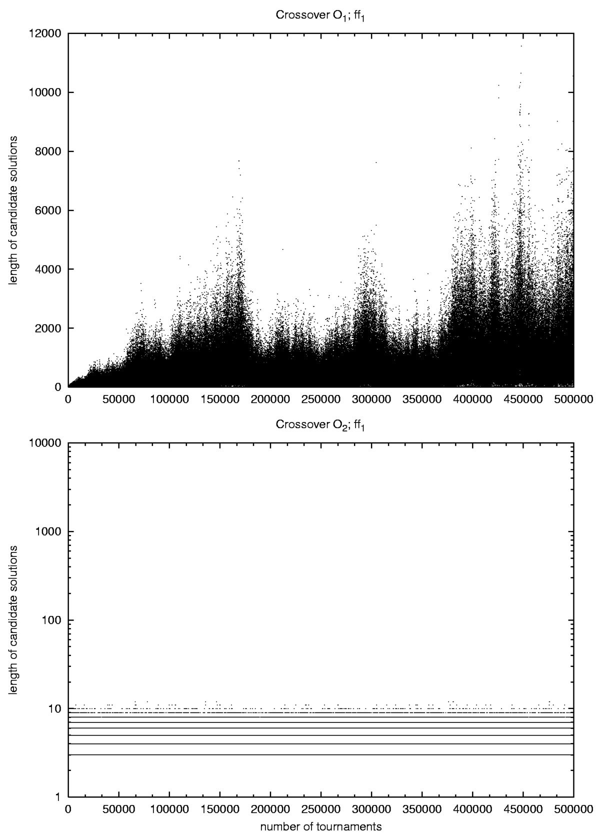

Fig. 4. *Fitness-function ff_1:* Time evolution of the genome sizes for the operators O_1 (top) and O_2 (bottom). These figures show that for the first fitness function operator O_1 leads to bloat whereas the second crossover operator quickly settles on very small genome lengths only. The third operator shows similar behavior (data not shown).

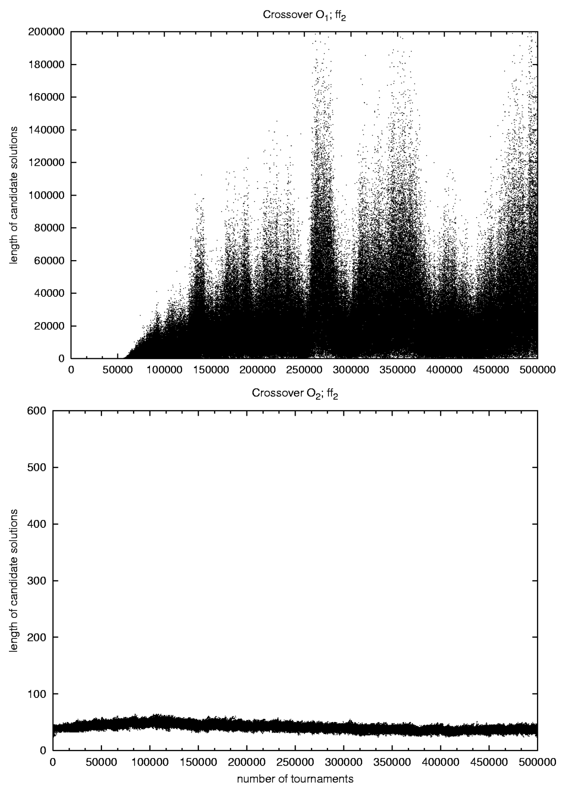

Fig. 5. *Fitness-function ff_2:* Time evolution of the genome sizes for the operators O_1 (top) and O_2 (bottom). The slight length bias of the second fitness function leads to an increase of the observed genome lengths for both simulations with O_1 and O_2. While for O_2 this increase is slight, O_1 shows a very substantial increased bloat if compared to figures 1 & 4. The behavior of simulations with O_3 are qualitatively similar to O_2 (data not shown).

removed from the population ff_1 is essentially the same as a flat fitness for the operator O_1. This is also confirmed by the distribution of the genome sizes; this distribution (data not shown) is (after an initial period) identical to the flat fitness case.

The qualitative behaviors of O_2 and O_3 are very similar to each other but different to O_1. They lack a substantial exploration of longer optimal solutions and throughout the simulation remain essentially restricted to short solutions. This is caused by the "hoovering" effect described above (in the case of O_2) and the bias for short sequences (in the case of O_3).

The **second fitness function** has a minimum length required in order for the genome to acquire optimal fitness; above this minimal genome size there are many solutions with optimal fitness. When using ff_2 there is thus a minimum size for the genome below which a solutions cannot compete (at least after the short initial period required for the system to find one optimal solution). Using O_1 with this fitness-function leads again to an exponential distribution of the genome sizes, however, with substantially higher mean length and a maximum size that reached the cap-size of 200000. It is unclear whether or not the maximal solution would be bound in an un-capped version of the GA.

Operators O_2 (see fig. 5) and O_3 (data not shown) show qualitatively similar behavior. During the first half of the simulation they settle on a symmetric distribution around a mean of about 45. This then falls to somewhat lower mean lengths between 25 and 35 (depending on the run). Common to all simulations is that (after a transitional period) the system never shows genome sizes that are substantially longer than that.

The **third fitness function** has an inherent bias for long solutions; the optimal solutions to ff_3 must be the longest allowed in the system. In the present case this is a genome with the length equal to the cap size (200000). Figure 6 shows that both operators O_1 and O_3 quickly lead to this optimal fitness solution. Closer inspection shows that in both cases the population is dominated by genomes equal in length to the capsize (as expected); all other sizes are substantially less frequent; although not apparent from the figure, in fact the length of nearly all solutions is equal to the cap-size. In the case of O_3 explorations of alternative solutions is restricted to a relatively small band around the capsize; note that every solution shorter than 200000 will be immediately weeded out. So, the band represents solutions that are one crossover away from the optimal solution; the smallest genome length is thus 180000 which represents a

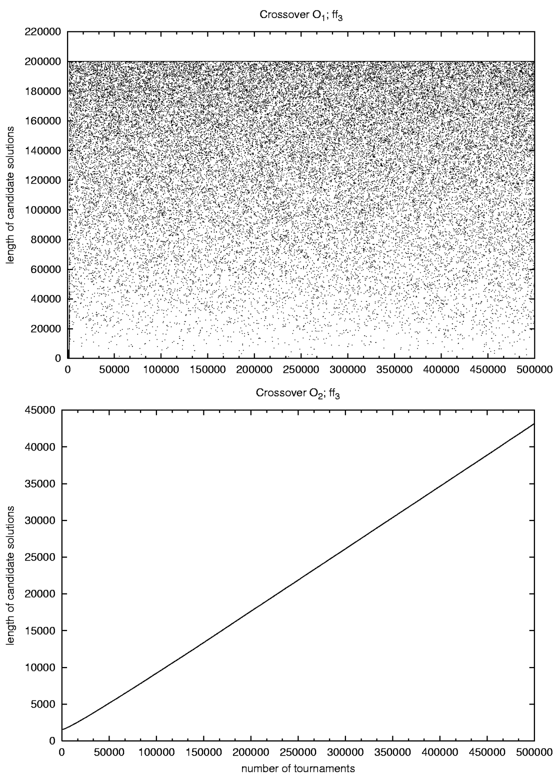

Fig. 6. *Fitness-function ff_3:* Time evolution of the genome sizes for the operators O_1 (top), O_2 (bottom). The first operator enables the system to find the fitness maximum and thus the maximum length solution. Closer analysis shows that nearly all genomes in the right figure are of length 200000. The operator O_2 on the other hand impedes the necessary growth to find the optimal solution. At the end of the simulation the genome lengths are still under 45000 long.

length change of 10 percent of 200000. Similarly, in the case of O_1 we also only see solutions that arise by one crossover, yet the possible change of length is greater in this case; this is reflected by the wider range of sizes in figure 6. A very different picture is offered by O_2. The population shows linear growth over the course of the simulation. Yet, the possible increases in size do not allow the system to find the best possible solution within the simulation time.

IV. Discussion

One of the foreseeable practical problems of variable length GAs/linear GP is bloat. Whether or not bloat will occur in a particular application of a variable size GA/linear GP will also depend on the specific circumstances, the fitness-function and the density of good solutions among longer genomes. As such, the present results are limited in their generality as would always be in a study of this kind. Despite those shortcomings, we believe that the chosen fitness functions and operators give at least some indications about their inherent tendency to cause bloat.

The experiments with a flat fitness function (see figure 1 & 2) indicate the main characteristics of the chosen operators. The first operator settles on a roughly exponential distribution confirming a theoretical result by Rowe and McPhee[17]. Even though the length distribution of O_3 is comparable to O_1 initially, after a transitional period the former settles onto smaller genomes in the flat fitness case; O_1 on the other hand continues to show strong fluctuations of the genome length for the entire simulation period. In practice this means that O_1 has an innate tendency for bloat.

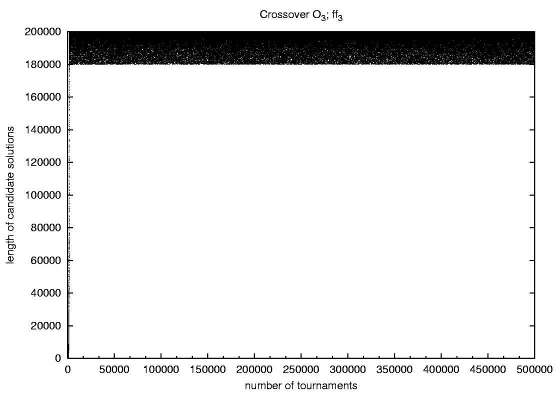

Fig. 7. *Fitness-function ff_3:* Time evolution of the genome sizes for the operator O_3. The operator O_3 immediately finds the fitness maximum with the associated maximum length genomes. As in figure 6 it is not apparent (but true) that nearly all solutions are of the maximum length 200000.

In the case of the flat fitness function the actually observed longest genomes in simulations of O_1 were in the order of 10000 long; the cap length of 200000 was never reached (see figure 2. If a bias for longer genomes is introduced the situation changes drastically: The fitness-function ff_3 has a very strong bias for long sequences, in that the optimal fitness can only be achieved by the longest possible solutions; in this case it is therefore not surprising that nearly all genomes actually take the maximal length (see figure 6). More surprising is the fact that much weaker biases are sufficient to cause substantial bloat. The fitness function ff_2 has a more subtle bias in the sense that there is a minimum required length of 18 in order to reach optimal fitness. This threshold of 18 is relatively short compared to the longest (and even the mean) genomes in the population observed in the flat fitness case. Yet this comparatively weak bias of ff_2 was sufficient to cause an increase of the sizes of the longest observed genomes by several orders of magnitude also increasing the maximum genome size to the capsize; this is evidenced in figure 5. (Note that this increase in size cannot be justified by a corresponding increase in fitness because the maximum fitness was reached at early stages of the simulation.) On the other hand, the potential of O_1 for quick growth enabled it to quickly find the optimal solution in the case of the third fitness function ff_3 (see figure 6).

Altogether this shows that the operator O_1 has a strong tendency to cause bloat even in the absence of a bias for longer solutions; this tendency is reinforced if the fitness function has an additional bias, even a weak one. Hence, operator O_1 is good at exploration of solutions of various sizes, but fairs poorly on the issue of bloat.

The simulations presented in this report suggest the opposite conclusion for the third operator O_2. The experiments show that the capability of this operator to explore solutions of various lengths is rather limited. In the case of a flat fitness functions O_2 locks itself into a narrow range of values (see figure 2); similar behavior is observed when fitness functions are introduced. Once the population has converged to a certain genome length no big length variations can happen any more. This has the effect that bloat is substantially reduced; but it also leads to an inflexibility in the case where optimal solutions are outside the range of initial values of the population and/or outside the range of an initial convergence of the population size. This is particularly well demonstrated by the simulations of O_2 with ff_3 (see figure 7); here the operator cannot keep up with the size changes required to find the best possible solutions within the given time. This operator thus seems to be fairly good at avoiding bloat, at least when compared to O_1, but does so at the expense of not being able to explore larger intervals of genome sizes.

Operator O_2 appears to strike a balance between those extremes. At least in the test problems investigated here it avoided bloat in the case of fitness functions ff_1 and ff_2 but was able to quickly find the best possible solution in the case of the fitness function ff_3.

V. Outlook and Conclusion

Altogether it thus appears that of the three operators investigated here, O_2 represents a useful combination between flexibility to explore solutions of various sizes and an inherent bias for shorter genomes that avoids bloat, at least in some circumstances. On the other hand, our experiments indicate that operators O_1 and O_3 are perhaps not useful except for applications that have very specific requirements. There may be applications where the user wants to restrict the length variations of the solutions or would like to explore a wide range of genome sizes.

Only real practical applications can show to what extent the results presented here will generalize to arbitrary fitness functions. These experiments however do indicate some broad characteristics of the operators under investigations; this will be useful as a general guideline for the practitioner who wishes to choose a crossover operator for a specific optimization problem.

There are several ways in which the current work can be extended. First of all it is desirable to mathematically formulate and prove properties of the behavior of the population under various operators and fitness functions. This is most likely only possible for the case of flat fitness and very simple fitness functions. At least for the case of an infinite population and the operator O_1 this has already been done[17].

Future experimental work will need to explore the effects of various population sizes. The experiments presented here assume a rather large population size of 1000. Such population sizes might not be realistic in practical applications. Finally, and most importantly the present experiments need to be compared to harder problems. The fitness functions used here are very much toy-problems; they were chosen to investigate the specific aspects of bloat in variable length GAs/linear GP. Real problems will normally be very different in that good solutions will be rare. It is unclear to what extent this influences the present conclusions.

VI. Acknowledgment

This work was funded by the EU Framework 6 project "Digital Business Ecosystems."

References

[1] W. Banzhaf and W. Langdon. Some considerations on the reason for bloat. *Genetic Programming and Evolvable Machines*, 3(1):81–91, 2002.

[2] Y. Bar-Yam. *Dynamics of Complex Systems*. Addison-Wesley, Reading, 1997.

[3] M. Brameier and W. Banzhaf. Neutral variations cause bloat in linear GP. In C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, editors, *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 286–296, Essex, 14-16 Apr. 2003. Springer-Verlag.

[4] L. Bull. Coevolutionary species adaptation genetic algorithms: A continuing saga on coupled fitness landscapes. In M. Capcarrere, A. Freitas, P. Bentley, C. Johnson, and J. Timmis, editors, *Advances in Artificial Life : 8th European Conference, ECAL 2005, Canterbury, UK, September 5-9, 2005, Proceedings*, pages 845–853. Springer, September 2005.

[5] A. Deckard and H. Sauro. Preliminary studies on the in silico evolution of biochemical networks. *Chembiochem*, 5(10):1423–1431, 2004.

[6] D. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[7] J. Grefenstette, C. Ramsey, and A. Schultz. Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5:355–381, 1990.

[8] I. Harvey. Species adaptation genetic algorithms: a basis for a continuing SAGA. In F. J. Varela and P. Bourgine, editors, *Proceedings of the First European Conference on Artificial Life. Toward a Practice of Autonomous Systems*, pages 346–354, Paris, France, 11-13 1992. MIT Press, Cambridge, MA.

[9] L. Hsieh, L. Luo, F. Ji, and H. Lee. Minimal Model for Genome Evolution and Growth. *Physical Review Letters*, 90(5):101–104, 2003.

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

[11] W. Langdon, T. Soule, R. Poli, and J. Foster. The evolution of size and shape. pages 163–190, 1999. in: Advances in genetic programming: volume 3.

[12] W. B. Langdon and R. Poli. Fitness causes bloat. In P. K. Chawdhry, R. Roy, and R. K. Pan, editors, *Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing*, pages 13–22. Springer-Verlag London, 23-27 1997.

[13] S. Luke and L. Panait. Fighting bloat with nonparametric parsimony pressure. In J. J. Merelo-Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, number 2439 in Lecture Notes in Computer Science, LNCS, pages 411–421, Granada, Spain, 7-11 Sept. 2002. Springer-Verlag.

[14] S. Luke and L. Panait. Lexicographic parsimony pressure. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 829–836, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[15] R. Poli, N. McPhee, and J. Rowe. Exact schema theory and markov chain models for genetic programming and variable-length genetic algorithms with homologous crossover. *Genetic Programming and Evolvable Machines*, 5(1):31–70, 2004.

[16] R. Poli, J. Rowe, C. Stephens, and A. Wright. Allele diffusion in linear genetic programming and variable-length genetic algorithms with subtree crossover. In *EuroGP '02: Proceedings of the 5th European Conference on Genetic Programming*, pages 212–227, London, UK, 2002. Springer-Verlag.

[17] J. Rowe and N. McPhee. The effects of crossover and mutation operators on variable length linear structures. In L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 535–542, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[18] S. Silva and E. Costa. Resource-limited genetic programming: the dynamic approach. In H.-G. Beyer, U.-M. O'Reilly, D. V. Arnold, W. Banzhaf, C. Blum, E. W. Bonabeau, E. Cantu-Paz, D. Dasgupta, K. Deb, J. A. Foster, E. D. de Jong, H. Lipson, X. Llora, S. Mancoridis, M. Pelikan, G. R. Raidl, T. Soule, A. M. Tyrrell, J.-P. Watson, and E. Zitzler, editors, *GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation*, volume 2, pages 1673–1680, Washington DC, USA, 25-29 June 2005. ACM Press.

[19] S. Silva, P. J. N. Silva, and E. Costa. Resource-limited genetic programming: Replacing tree depth limits. In B. Ribeiro, R. F. Albrecht, A. Dobnikar, D. W. Pearson, and N. C. Steele, editors, *Adaptive and Natural Computing Algorithms*, Springer Computer Series, pages 243–246, Coimbra, Portugal, 21-23 Mar. 2005. Springer.

[20] T. Soule and J. A. Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309, Winter 1998.

[21] A. Wu and I. Garibay. The proportional genetic algorithm: Gene expression in a genetic algorithm, 2002.