# Why introns in Genetic Programming grow exponentially

Wolfgang Banzhaf[1] Peter Nordin[13] and Frank D. Francone[2]

[1] LS11, Dept. of CS, University of Dortmund, 44221 Dortmund, Germany
[2] RML Inc., 360 Grand Ave., Oakland, CA, 94610 USA
[3] DaCapo AB, Kronhusgatan 9, 41105 Goeteborg, Sweden

**Abstract.** We argue that introns or non-coding segments of GP programs are a means for genomes to protect themselves from the destructive effects of crossover and other operators. There are, however, both advantages and disadvantages to these protection devices.

Various studies have shown that introns in Genetic Programming Systems grow heavily if not exponentially towards the end of runs [3, 12, 5, 8, 1, 11], with some partially contradictory explanations that have been put forward to date (see also [7, 6, 10]). The matter is far from being settled though. This field is very young and it may be that all of these studies, some seemingly inconsistent, just represent different spots on the very complex surface that represents the effect of introns. Much more study is necessary before anything more than tentative conclusions may be stated.

Before we suggest a possible explanation for the problem of explosive intron growth in GP, a few qualifications are necessary:

- Introns may have differing effects before and after exponential growth of introns begins. After exponential growth occurs, the exponential effect surely overwhelms whatever effect the introns had previously, if any.
- Different systems may generate different types of introns with different probabilities. It may, therefore be harder to generate introns in some GP systems than in others.
- The extent to which genetic operators are destructive in their effect is likely to be a very important initial condition in intron growth. This underlines the importance of measuring and reporting on destructive, neutral and constructive crossover or mutation figures when doing intron research.
- Mutation and crossover may affect different types of introns differently.
- Finally, it is important to distinguish between emergent introns and artifical intron equivalents [1, 9, 2]. In most systems, the existence and growth of artifical intron equivalents is more or less free to the system — a gift from the programmer so to speak. This may well make artificial intron equivalents much more likely to engage in exponential growth than emergent introns.

We believe that the reason for the growth behavior is that introns can provide very effective *global protection* against the destructive effects of operators (mainly crossover in GP). By that, we mean that the protection is global to the entire individual. This happens because toward the end of a run, the individuals are at or close to their best performance. It is difficult for them to improve their fitness by solving the problem better. Instead, their best strategy for survival changes. Their strategy becomes to prevent destructive genetic operators from disrupting the good solutions already found.

One can reformulate the equation for effective fitness given in [4] as follows:

$$f_j^e = f_j \cdot [1 - \sum_r p_j^{D,r} \cdot \left(1 - (L_j^{i,r}/L_j^a)\right)] \tag{1}$$

where $p^{D,r}$ now lumps together both the probability of application and of destructiveness of an operator $r$, and $L_j^{i,r}$ is the corresponding intron length.

When fitness $(f_j)$ is already high, the possibility of improving effective fitness by changing actual fitness is much lower than at the beginning of the run. But an individual can continue to increase its effective fitness, even late in a run, by increasing the number of introns $(L_j^{i,r})$ against $r$ in its structure.

Further, there is no end to the predicted growth of introns, apart from reaching the maximally allowed length. Because the number of introns in an individual is always less than the absolute size of an individual, the ratio (supressing index $r$), $L_j^i/L_j^a$ is always less than one. So introns could grow infinitely and continue to have some effect on the effective fitness of an individual as long as $p^D > 0$.

The effect of explosive intron growth is dramatic. Normally, crossover is destructive more than seventy-five percent of the time. But after introns occupy most of the population, destructive crossover is replaced almost completely with neutral crossover. Individuals are merely swapping introns with higher and higher probability each generation. Swapping code that has no effect between two individuals by definition has no effect. Hence neutral crossover comes to dominate a run after the explosive growth of introns.

Introns in GP are always a mixed blessing. On the one hand they show how variable size genomes in EAs reflect the biological paradigm of evolution. With them, we also observe some unwanted/unsuspected aspects of evolution. Furthermore, they could be put to work for compressing or for structuring effective code. On the other hand, introns slow down search processes even to the point of inhibiting evolution altogether. How they can be controlled or otherwise made useful for the search process is presently a question of utmost importance.

## References

1. David Andre and Astro Teller. A study in program response and the negative effects of introns in genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 12–20, Stanford University, CA, USA, 28–31July 1996. MIT Press.
2. Peter J. Angeline. Two self-adaptive crossover operators for genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 5, pages 89–110. MIT Press, Cambridge, MA, USA, 1996.

3. Peter John Angeline. Genetic programming and emergent intelligence. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 4, pages 75–98. MIT Press, 1994.
4. W. Banzhaf, F. Francone, and P. Nordin. On some emergent properties of variable size evolutionary algorithms. In *Workshop on Variable Size Genomes at this conference*, 1997.
5. W. B. Langdon. Evolving data structures using genetic programming. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 295–302, Pittsburgh, PA, USA, 15-19July 1995. Morgan Kaufmann.
6. W.B. Langdon and R. Poli. Fitness causes bloat. In *2nd Online World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2)*.
7. Nicholas Freitag McPhee and Justin Darwin 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, 15-19July 1995. Morgan Kaufmann.
8. Peter Nordin, Frank Francone, and Wolfgang Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In Justinian P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 6–22, Tahoe City, California, USA, 9July 1995.
9. Peter Nordin, Frank Francone, and Wolfgang Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 6, pages 111–134. MIT Press, Cambridge, MA, USA, 1996.
10. J. Rosca. Analysis of complexity drift in genetic programming. unpublished manuscript, 1997.
11. Terence Soule, James A. Foster, and John Dickinson. Code growth in genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 215–223, Stanford University, CA, USA, 28–31July 1996. MIT Press.
12. Walter Alden Tackett. *Recombination, Selection, and the Genetic Construction of Computer Programs*. PhD thesis, University of Southern California, Department of Electrical Engineering Systems, 1994.