

EMERGENT PHENOMENA IN GENETIC PROGRAMMING¹

LEE ALTENBERG

Institute of Statistics and Decision Sciences
Duke University, Durham, NC 27708-0251 USA

ABSTRACT

Evolutionary computation systems exhibit various emergent phenomena, primary of which is adaptation. In genetic programming, because of the indeterminate nature of the representation, the evolution of both recombination distributions and representations can emerge from the population dynamics. A review of ideas on these phenomena is presented, including theory on the evolution of evolvability through differential proliferation of subexpressions within programs. An analysis is given of a model of genetic programming dynamics that is supportive of the “Soft Brood Selection” conjecture, which was proposed as a means to counteract the emergence of highly conservative code, and instead favor highly evolvable code.

1. Introduction

The ability of complex systems to exhibit emergent properties is one of their main attractions as subjects of study. By “emergent” I here mean behaviors that are not described in the specification of the system, but which become evident in its dynamics. In conventional computer programming, the ability of the program to carry out a desired behavior must be specified by the programmer. The field of evolutionary computation includes the study and design of algorithms in which adaptation is an emergent property achieved by the Darwinian process of selection on heritable variation for adaptations.

But adaptation is not the only emergent property that can be exhibited by evolutionary algorithms. The phenomenon of hybrid inviability between subpopulations, which implies coadaptation within the evolved genomes, has been noted for GAs and GP. Evolutionary algorithms can be designed so that two other properties can emerge as well: the evolution of the genetic transmission function and the evolution of the representation. In classical, fixed length genetic algorithms (GAs), and in the simplest evolution strategies, the genetics and representation are specified and fixed at the outset. But early on in the development of evolution strategies and later with GAs modifier genes controlling the genetic operators were included in the genotype, allowing adaptation of the genetic operators as an emergent property.

Representations have been made to adapt as an emergent property in GAs by using “adaptive parameter encoding” and “messy GAs”.

Selection is usually a specified property of evolutionary algorithms, but in some artificial life programs such as *Tierra*, selection emerges from the dynamics of self-replication.

A rough categorization of different dynamical systems by whether the properties such as

¹pp. 233-241 in *Proceedings of the Third Annual Conference on Evolutionary Programming: 24-25 February 1994, San Diego, California*, Anthony V. Sebald and Lawrence J. Fogel, ed., World Scientific, Singapore.

Table 1: A rough categorization of emergent and specified properties of different adaptive systems. E = emergent, S = specified, — = not applicable.

SYSTEM	PHENOMENON			
	Adaptation	Selection	Genetics	Representation
Analytic design	S	—	—	—
Gradient ascent	E	—	—	—
Classical GA, EP	E	S	S	S
Evolution Strategies	E	S	E/S	S
GA w/ modifiers	E	S	E/S	S
GP	E	S	E/S	E/S
Artificial-Life	E	E	E/S	E/S
Life	E	E	E	E

adaptation, selection, genetics, and representation are specified, emergent or a combination is given in Table 1.

In genetic programming (GP), because of the indeterminate size, structure and algorithmic properties of the tree-structured representation, both the genetics and representation can evolve as an emergent property of the dynamics. Thus GP has an intermediate combination of emergent and specified features [Walter Tackett, personal communication].

In this paper I review current ideas about emergent phenomena in GP, including the evolution of “junk code”, and the evolution of representations. The issue I have focused on in the evolution of representations is their *evolvability*, i.e. the probability that alterations in a program can produce a fitness increase. Previous work on this is summarized. I then take up in greater detail the analysis of a conjecture I made on a selection operator — soft brood selection — designed to increase the evolvability of code that is produced in GP. The results are supportive of the conjecture.

1.1. “Junk code” and Recombination Modification

The phenomenon of “bloating” has been observed in typical GP runs, in which the average program size swells as the population of programs matures under repeated action of selection and recombination operators. Much of this swelling appears to be due to the accumulation of “junk code” — code that can be eliminated without changing the behavior of the program. The emergence of junk code may be analogous to the evolution of “junk DNA” found in eukaryotic genomes, such as introns, which in recent GA experiments have been introduced by hand into the representation. Junk code does not contribute to fitness, but it modifies the recombination operator by increasing the recombination rates between the code flanking the junk code, and decrements recombination elsewhere by absorbing recombination events.

Several ideas have been offered to explain the evolution of junk code. The “defense against crossover” hypothesis poses that junk code absorbs crossover events, and their neutral outcome on fitness is an advantage over recombination events that disrupt program function. As such, “defense against crossover” is in keeping with the heuristic “Principle of Minimum Genetic Load” offered as a guiding principle for the evolution of genetic systems. It is quite similar to

the “mutation-absorption” model for the evolution of smooth adaptive landscapes², and suffers from the same mechanistic problem: In order for selection to systematically increase the frequency of junk code, the junk code, which is selectively neutral, has to generate associations with selectively advantageous code. But as a target for crossover, crossover itself severs the association. Thus, junk code would not behave as allelic modifiers of recombination rate, nor be subject to the theory developed for such modifiers.

The hitchhiking hypothesis [Walter Tackett, personal communication] suggests that junk code is hitchhiking along with useful code and this causes program bloating. As described in item 1 below, subexpressions can emerge as a level of replicator in GP, and this hitchhiking would have to be at this level of selection. Junk code that happened to be associated with code that is proliferating *within* programs because of its effects on fitness would also proliferate.

The hitchhiking hypothesis has a sound mechanistic underpinning. But I would suggest further that program bloating with junk code may be simply driven by the recombination operator, a neutral diffusion process with upward drift. In other words, even in the absence of selection on programs, programs might be expected to bloat because of the asymmetry between addition and deletion of code at the lower boundary of program size. This conjecture can be readily tested.

1.2. Evolution of the Representation

The way changes in the data structure of a program map to changes in its performance comprises the *variational* properties of its representation. In order for evolution in genetic programming to achieve performance increases, the representations must give a non-vanishing likelihood that variation produces performance improvements, and this property I refer to as “evolvability”. The indeterminate nature of the structure of programs in GP allows the evolution of evolvability as an emergent property. My main hypotheses regarding the evolution of representations in GP are summarized:

1. Because the recombination operator can produce multiple copies of subexpressions within programs, subexpressions emerge as a new level of replicator through their possible differential proliferation.
2. The proliferation rate of a subexpression, i.e. its “constructional” fitness, is determined by the distribution of fitness changes caused by its insertion or deletion in the programs in the population, in concert with the population dynamics:
 - (a) In conditions typical of biological evolution, in which new copies of a subexpression go to fixation before recombination changes it again, subexpressions increase in number proportional to their chance of increasing fitness when added to programs in the population (their evolvability value). Thus evolvability increases in the population in proportion to the variance in evolvability values among the subexpressions (Theorem 4 in).

²Conrad, it should be noted, is among the first to consider the evolution of the genetic representation as a means to increase evolvability, mechanistic difficulties notwithstanding.

- (b) In conditions typical of GP runs, with recombination rates and selection rates on the same order, neutral or near-neutral (“conservative”) subexpressions may have the highest constructional fitness. This would be especially true when a mature population has reached a recombination-selection balance. Thus the representation may evolve away from high evolvability toward robustness of fitness in the presence of recombination, reducing the rate of adaptation.
3. In order to control the evolution of the representation toward greater evolvability, a number of modifications of the population dynamics are proposed: these include estimating the evolvability value of subexpressions and adapting the recombination operator to use them more often, and a reproductive operator: soft brood selection.

The idea behind soft brood selection is to shield reproduction from the costs of producing deleterious offspring. Soft brood selection accomplishes this by holding a tournament between the members of a brood of two parents, and using the winner as the offspring contributed by that mating. It is “soft” selection because one offspring is contributed regardless of the offspring fitnesses. The production of deleterious offspring is one of the costs of exploration. When recombination rates are high, and populations near a recombination-selection balance, it is more of a premium to have high *average* offspring viability than high likelihood of offspring being fitter than parents when this concurrently depresses the average offspring viability. Soft brood selection should shift constructional selection away from an emphasis on subexpressions that yield high average offspring viability to those that yield high evolvability. It should be able to alter the variational properties of the code that emerges from the GP run.

2. A Model of Genetic Programming Dynamics

To analyze the effect of soft brood selection, a simple model of genetic programming dynamics can be formulated. Here I assume that the only genetic operator is recombination. The population dynamics assume generational reproduction (discrete, non-overlapping generations), the infinite population size limit, and frequency-independent selection. The GP algorithm will thus be:

1. Pairs of parents are picked through roulette wheel selection;
2. With probability α (the probability of no recombination), one of the parents is chosen as the contribution of this mating to the next generation; with probability $1 - \alpha$, an offspring produced by applying the recombination operator, with or without soft brood selection, is contributed to the next generation.
3. In the case of soft brood selection, a brood of size b is produced by each mated pair and a tournament held between the offspring; the winner of the tournament is the contribution of this mated pair to the next generation.

This is described with these definitions. Let

\mathcal{P} be the space of programs, and \mathcal{S} be the space of different subexpressions extractable from programs in \mathcal{P} by the recombination operator;

x_i be the frequency of program i in the population;

w_i be the fitness of program i , and $\bar{w} = \sum_i w_i x_i$ be the mean fitness of the population;

α be the probability of recombination during reproduction;

$C(s \leftarrow k)$ be the probability that the recombination operator picks out a particular subexpression s from a program k for recombination into the other parent. So $\sum_{s \in \mathcal{S}} C(s \leftarrow k) = 1$, for all $k \in \mathcal{P}$;

$P(i \leftarrow j, s)$ be the probability that when subexpression s is recombined into program j it produces program i . So $\sum_{i \in \mathcal{P}} P(i \leftarrow j, s) = 1$, for all $j \in \mathcal{P}$. Note that $P(i \leftarrow j, s)$ is not symmetrical in arguments j and s . In I erroneously noted that $C(s \leftarrow i) > C(s \leftarrow j)$, given that subexpression s is added to program j to make program i ; clearly it is possible that a single s could replace a subtree containing several copies of s .

This yields the following recursion for the change in frequency of program i in the absence of soft brood selection:

Recursion 1 (Genetic Programming Model)

The frequency of program i in the next generation is:

$$x'_i = (1 - \alpha) \frac{w_i}{\bar{w}} x_i + \alpha \sum_{j, k \in \mathcal{P}} \frac{w_j w_k}{\bar{w}^2} x_j x_k \sum_{s \in \mathcal{S}} P(i \leftarrow j, s) C(s \leftarrow k). \quad (1)$$

2.1. The “Constructional Fitness” of a Subexpression

In analyzing the evolution of the representations in GP, we would like to know how the composition of subexpressions within the programs changes as a function of the effect each subexpression has on fitness when recombined into a program. The rate of proliferation of a subexpression in the population is measured by the change in the average value of $C(s \leftarrow i)$:

$$\bar{u}_s = \sum_{i \in \mathcal{P}} C(s \leftarrow i) x_i,$$

the average chance that the recombination operator would pick subexpression s from a randomly chosen program. Using Eq. (1) we have

$$\begin{aligned} \bar{u}'_s &= \sum_{i \in \mathcal{P}} C(s \leftarrow i) x'_i \\ &= (1 - \alpha) \sum_{i \in \mathcal{P}} C(s \leftarrow i) \frac{w_i}{\bar{w}} x_i + \alpha \sum_{i, j, k \in \mathcal{P}} C(s \leftarrow i) \frac{w_j w_k}{\bar{w}^2} x_j x_k \sum_{r \in \mathcal{S}} P(i \leftarrow j, r) C(r \leftarrow k). \end{aligned} \quad (2)$$

Elimination of the term times α gives a “Schema Theorem” for genetic programming:

$$\bar{u}'_s \geq (1 - \alpha) \frac{v_s}{\bar{w}} \bar{u}_s,$$

where $v_s = \sum_{i \in \mathcal{P}} C(s \leftarrow i) w_i x_i / \sum_{i \in \mathcal{P}} C(s \leftarrow i) x_i$ is the marginal or “schema” fitness of subexpression s . But this is not useful for understanding the evolution of the representations,

since it is a pure selection effect without reference to the genetic operator, which is where brood selection acts.

Analysis of Eq. (2) is difficult due to the complex possibilities relating $C(s \leftarrow i)$ and $C(s \leftarrow j)$ when j is a parent of i . Instead, I examine here the frequency with which subexpression s is used to *produce* the offspring i , with the expectation that greater use of a subexpression in producing the next generation would be correlated with an increase in \bar{u}_s .

2.2. Analysis of Soft Brood Selection

We wish to know how different subexpressions contribute differentially to the creation of new programs in the population. We begin with the statistic

$$d(s|j, k) = \sum_i P(i \leftarrow j, s) C(s \leftarrow k) = C(s \leftarrow k),$$

the proportion of offspring of parents j and k that were produced by a recombination of subexpression s into program j . We want to know how soft brood selection changes the proportion, $d(s|j, k)^{(b)}$, of subexpression s contributed to the winning offspring of the brood of size b .

Define the measurement function:

$$F_i(w) = \begin{cases} 1 & w_i \leq w \\ 0 & w_i > w \end{cases}.$$

For any parental pair $\{j, k\}$, the proportion, $G_{jk}(w)$, of their offspring with fitness less than or equal to w is:

$$G_{jk}(w) = \sum_{i,s} F_i(w) P(i \leftarrow j, s) C(s \leftarrow k).$$

Let $g_{jk}(w)$ be the proportion of offspring of parents j and k with fitness equal to w . Thus, allowing for discrete measure in integration, $G_{jk}(w) = \int_0^w g_{jk}(u) du$.

Lemma 1

When tournament selection is applied to a brood of size b from parents j and k , the probability that the fitness of the winning offspring will be less than or equal to w is $G_{jk}^b(w)$. The proportion, $g_{jk}^{(b)}(w)$, of winning offspring with fitness equal to w will be

$$\begin{aligned} g_{jk}^{(b)}(w) &= G_{jk}^b(w) - [G_{jk}(w) - g_{jk}(w)]^b \\ &\approx b G_{jk}^{b-1}(w) g_{jk}(w) \quad \text{for small } g_{jk}(w). \end{aligned}$$

Lemma 2

The effect of soft brood selection is to boost the probability that parents j and k produce an offspring with fitness w by the factor

$$\omega(w|j, k)^{(b)} = g_{jk}^{(b)}(w)/g_{jk}(w) \approx b G_{jk}^{b-1}(w) \quad \text{for small } g_{jk}(w). \quad (3)$$

Using the lemmas we obtain:

Recursion 2

Under soft brood selection with a brood size b , the frequency of program i in the next generation is:

$$x'_i = (1 - \alpha) \frac{w_i}{\bar{w}} x_i + \alpha \sum_{j,k \in \mathcal{P}} \frac{w_j w_k}{\bar{w}^2} x_j x_k \omega(w_i | j, k)^{(b)} \sum_{s \in \mathcal{S}} P(i \leftarrow j, s) C(s \leftarrow k). \quad (4)$$

and:

Theorem 1 (Soft Brood Selection)

When tournament selection is applied to a brood of size b from parents j and k , the probability, $d(s | j, k)^{(b)}$ that the winning offspring will be created by the contribution of subexpression s is:

$$\begin{aligned} d(s | j, k)^{(b)} &= C(s \leftarrow k) \sum_i \omega(w_i | j, k)^{(b)} P(i \leftarrow j, s) \\ &\approx b C(s \leftarrow k) \sum_i G_{jk}^{b-1}(w_i) P(i \leftarrow j, s) \quad \text{with small } g_{jk}(w_i) \text{ for all } i. \end{aligned} \quad (5)$$

Proof. Given Lemma 1, all we need include is $P(i \leftarrow j, s) / g_{jk}(w_i)$, the probability that the winner is i given the winner has fitness $w = w_i$. Summing over all possible offspring i gives us $d(s | j, k)$. ■

From Eq. (5), the factor

$$\bar{w}(s | j, k)^{(b)} = \sum_i \omega(w_i | j, k)^{(b)} P(i \leftarrow j, s) \approx b \sum_i G_{jk}^{b-1}(w_i) P(i \leftarrow j, s)$$

amounts to a selection coefficient on subexpression s due to brood selection, in terms of its participation in offspring production. This selection coefficient is a projection of the distribution $P(i \leftarrow j, s)$ onto $G_{jk}^{b-1}(w_i)$. In the case of no brood selection ($b = 1$), it can be seen that $\bar{w}(s | j, k)^{(b)} = 1$. As brood selection increases with larger b , the factors $G_{jk}^{b-1}(w_i)$ remain substantial only for higher and higher w_i . It is assumed that boosting a subexpression's participation in offspring production will enrich the offspring for this subexpression.

Hence, the subexpressions with the greatest probability of producing offspring with high fitness gain the largest brood selection coefficients. The probability of producing offspring with low fitness is irrelevant to the calculation of $d(s | j, k)^{(b)}$ under large brood size because the terms $G_{jk}^{b-1}(w_i)$ are negligible. Thus, reproduction is shielded from the effect of producing deleterious offspring, and only the evolvability value of the subexpression is relevant to its transmission.

The average participation of subexpression s in reproduction over the whole population is

$$\begin{aligned} \bar{d}(s)^{(b)} &= \sum_{j,k} d(s | j, k)^{(b)} \frac{w_j w_k}{\bar{w}^2} x_j x_k \\ &\approx b \sum_{j,k} \frac{w_j w_k}{\bar{w}^2} x_j x_k C(s \leftarrow k) \sum_i G_{jk}^{b-1}(w_i) P(i \leftarrow j, s) \quad \text{with small } g_{jk}(w_i) \forall i, j, k. \end{aligned} \quad (6)$$

This mean ‘‘prolificacy’’ of subexpression s thus depends on the frequencies, x_i , of programs in the population. From one generation to the next it is unlikely to change much, but over the course of a GP run, subexpressions may change in their relative prolificacy, as seen in and elsewhere.

3. Discussion

Genetic programming exhibits several emergent phenomena not found in classical genetic algorithms, including program bloating, the differential proliferation of subexpressions, and the evolution of the variational properties of the program representations. For an understanding of these phenomena one needs to turn to the population genetic dynamics of the algorithm. Program bloating may represent hitchhiking or simply neutral diffusion with upward drift. The evolution of representations has been hypothesized to result from the proliferation of subexpressions within programs as a consequence of the fitness distribution of recombinant offspring resulting from subexpression exchange, and depends on the magnitude of the recombination rate and the maturity of the population.

Soft brood selection has been proposed as a method of shifting the evolution of representations away from a conservative strategy toward an exploratory strategy. Here I have analyzed the effect of soft brood selection on the differential contribution of different subexpressions toward reproduction. The result shows that larger brood size gives an advantage to subexpressions with high evolvability value, regardless of their average effect on offspring fitness.

The production of large broods multiplies the computational effort in the GP run. However, by using reduced numbers of fitness cases during the brood selection tournament, which merely increases the variance in the effect of brood selection on subexpressions, the computational effort can be kept the same as in the absence of brood selection. Then the conjectured increase in the evolvability of the population could make soft brood selection an advantageous operator in genetic programming.

Acknowledgements

I thank Roger Altenberg and David Fogel for their consideration during the completion of this paper. Thanks to Pete Angeline and David Fogel for inviting this contribution, and Marcy Uyenoyama for computational support.

References

- [1] Altenberg, L. 1994. The evolution of evolvability in genetic programming. In K. E. Kinnear, editor, *Advances in Genetic Programming*, Cambridge, MA. MIT Press.
- [2] Altenberg, L. and M. W. Feldman. 1987. Selection, generalized transmission, and the evolution of modifier genes. I. The reduction principle. *Genetics* 117: 559–572.
- [3] Angeline, P. J. and J. B. Pollack. 1994. Coevolving high-level representations. In C. G. Langton, editor, *Artificial Life III*, Menlo Park, CA. Addison-Wesley. In press.
- [4] Bergman, A. and M. W. Feldman. 1992. Recombination dynamics and the fitness landscape. *Physica D* 56(1): 57–67.
- [5] Conrad, M. 1979. Mutation-absorption model of the enzyme. *Bulletin of Mathematical Biology* 41: 387–405.
- [6] Dod, B., L. Jermin, P. Boursot, V. Chapman, and F. Bonhomme. 1991. Do co-adapted gene systems create barriers to gene flow in hybrid zones the case of the mus-musculus x chromosome? *Genetical Research* 58(1): 77.
- [7] Feldman, M. W. 1972. Selection for linkage modification: I. random mating populations. *Theoretical Population Biology* 3: 324–346.

- [8] Goldberg, D., K. Deb, and B. Korb. 1990. Messy genetic algorithms revisited: studies in mixed size and scale. *Complex Systems* 4(4): 415–444.
- [9] Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.
- [10] Kimura, M. 1960. Optimum mutation rate and degree of dominance as determined by the principle of minimum genetic load. *J. Genetics* 57: 21–34.
- [11] Kimura, M. 1967. On the evolutionary adjustment of spontaneous mutation rates. *Genetical Research* 9: 23–34.
- [12] Levenick, J. R. 1991. Inserting introns improves genetic algorithm success rate: Taking a cue from biology. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 123–127, San Mateo, CA. Morgan Kaufmann.
- [13] Lewes, G. H. 1874. *Problems of Life and Mind*, volume 1 of 1. J.R. Osgood, Boston, author's edition, page 98.
- [14] Liberman, U. and M. Feldman. 1986. A general reduction principle for genetic modifiers of recombination. *Theoretical Population Biology* 30: 341–371.
- [15] Mallet, J. 1989. The genetics of warning color in peruvian hybrid zones of *Heliconius erato* and *Heliconius melpomene*. *Proceedings Of The Royal Society Of London* 236(1283): 163–185.
- [16] Nei, M. 1967. Modification of linkage intensity by natural selection. *Genetics* 57: 625–641.
- [17] Ray, T. S. 1991. Is it alive, or is it GA? In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 527–534, San Mateo, CA. Morgan Kaufmann.
- [18] Rechenberg, I. 1965. Cybernetic solution path of an experimental problem. Technical report, Royal Aircraft Establishment Library, Hants, U.K.
- [19] Rechenberg, I. 1973. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart.
- [20] Schaffer, J. and A. Morishima. 1987. An adaptive crossover distribution mechanism for genetic algorithms. In J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 36–40, Hillsdale, NJ. Lawrence Erlbaum Associates.
- [21] Schwefel, H.-P. 1981. *Numerical optimization of computer models*. Wiley, Chichester.
- [22] Shaefer, C. G. 1987. The ARGOT strategy: adaptive representation genetic optimizer technique. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 50–58, Hillsdale, NJ. Lawrence Erlbaum Associates.
- [23] Singleton, A. Internet Message-Id: 2970958767.0.p00396@psilink.com, 1994. Personal communication.
- [24] Tackett, W. A. and A. Carmi. 1994. The unique implications of brood selection for genetic programming. Submitted to: IEEE 1994 World Congress on Computational Intelligence.
- [25] Wallace, B. 1975. Hard and soft selection revisited. *Evolution* 29: 465–473.
- [26] Zuckerkandl, E. 1992. Revisiting junk DNA. *Journal of Molecular Evolution* 34(3): 259–271.