Chapter 14

EVOLUTION ON NEUTRAL NETWORKS IN GENETIC PROGRAMMING

Wolfgang Banzhaf1 and Andre Leier1
1Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, A1B 3X5, CANADA

Abstract We examine the behavior of an evolutionary search on neutral networks in a simple linear GP system of a Boolean function space problem. To this end we draw parallels between notions in RNA-folding problems and in Genetic Programming, observe parameters of neutral networks and discuss the population dynamics via the occupation probability of network nodes in runs on their way to the optimal solution.

Keywords: Neutrality, Linear GP, Networks, Population Dynamics

1. Introduction

For more than a decade now, neutrality has been observed to play an important role in Genetic Programming (GP) runs. This was originally believed to be an atypical phenomenon, perhaps related to the choice of representation (Koza, 1992; Altenberg, 1994a; Angeline, 1994). It was later realized that introns or non-effective code, as it became to be called, constitute the bulk of material generating neutrality in GP and that this type of code would appear in most representations of GP systems (Nordin and Banzhaf, 1995). For a long time the debate centered around questions of reasons for the emergence of this type of code which certainly was unintended by the designers of GP systems, and originally deemed disadvantageous (Soule et al., 1996; Langdon and Poli, 1998; Soule and Heckendorn, 2002).

During the same time, it was proposed that the theory of neutral mutations as put forward in the seventies and eighties for natural evolution (Kimura, 1983),

could be understood in terms of the existence of neutral networks (Schuster, 1995; Forst et al., 1995; Reidys et al., 1997). Subsequent to that proposal various natural evolutionary systems have been examined, and the existence of neutral networks has been confirmed (Huynen et al., 1996; Babajide et al., 1997). Its benefits for evolution were gradually reveiled (Nimwegen et al., 1998; Schultes and Bartel, 2000), and thus it was natural to ask what neutral networks would have to offer for evolutionary search.

Barnett proposed to adopt a search paradigm different from a population-based GA search in landscapes with considerable neutrality (Barnett, 2001). Smith et al (T. Smith and O'Shea, 2001) argue that, due to higher evolvability, GA systems with neutrality in search behave more aptly in difficult search landscapes.

Recently, the confluence of both lines of inquiry can be observed in Genetic Programming as well. Early observations (Banzhaf, 1994) spoke to the advantage of using plenty of neutrality. In the context of circuit design using Cartesian GP Miller and coworkers argued for search efficiency as one characteristic of representations with neutrality (Vassilev and Miller, 2000b; Vassilev and Miller, 2000a; Vassilev et al., 2003). Ebner (Ebner et al., 2002) pointed out how neutral networks can influence evolvability and Yu (Yu and Miller, 2001) studied the interaction between neutral and adaptive mutations in the context of search in Boolean function landscapes.

In this contribution we shall discuss neutrality and the benefit of neutral networks in the context of a simple Boolean search problem using a linear GP representation, that consists of registers and logic operators. We shall show the relation between genotype and phenotype networks, discuss how the search benefits from neutrality as offered by non-effective code, and demonstrate the population dynamics of a search process. In a final section we shall put our eyes on robustness of the evolutionary solutions, and ask ourselves how evolvability of the search process can be improved if the observations put forward here can be generalized.

2. Problem, GP representation and Search Operators

In order to be able to examine the effects we are interested in, we have chosen a small problem instance of a Boolean problem space. While it can be argued that this space is not suitable to solve real problems, the emphasis here is on trying to understand the influence of neutrality, notably its benefits.

The problem space under consideration is the NAND space where two binary inputs x_1 and x_2 are used and the output x_3 is studied under various NAND-combinations of inputs.

$$x_3 = f_{NAND}(x_1, x_2) \tag{14.1}$$

This follows work done by (Langdon and Poli, 1999) where it was shown, for tree-based GP, that there is a complexity threshold above which all Boolean functions can be reached by a combination of Boolean operators on inputs.

We use a linear GP representation because it is much easier to analyse in terms of non-effective code (Banzhaf et al., 1998; Brameier and Banzhaf, 2001), and because it is easier to understand. The representation consists of a set of instructions in a register machine language, interpreted by the CPU as a program. As content of the registers we only allow Boolean values ”0” and ”1”, as operators of these programs only the logical NAND operation.

Even with so small a set of elements, combinatorics is at play, forcing us to quickly relinquish the plan to depict everything exhaustively. One choice we have is whether we want to have only a single type of register (read-and-write) which can act both as source and destination register of the programs executed, or two types of registers (input and calculation) which differ in that input registers hold the input values constantly, i.e. are only acting as source registers, and calculation registers can act both as source and destination registers.

Table 2 shows the combinatorics in these two different systems, depending on the length of programs allowed. In the following, we shall concentrate on $C = I = 2$. The first calculation register also works as the output register.

Table 14-1. Comparison of number of programs for different number of registers. C: Number of calculation registers; I: number of input registers; L: Length of programs in number of instructions. The number of programs is calculated by $(I + C)^{2L} C^L$.

C Registers	*I Registers*	$L = 2$	$L = 3$	$L = 4$	$L = 5$	...	$L = 10$
2	0	64	512	4.1×10^3	3.3×10^4		1.1×10^9
3	0	729	19,683	5.3×10^5	1.4×10^7		2.1×10^{14}
1	2	81	729	6.6×10^3	5.9×10^4	...	3.5×10^9
2	2	1,024	32,768	1.0×10^6	3.4×10^7		1.1×10^{15}
3	2	5,625	421,875	6.3×10^6	2.4×10^9		5.6×10^{18}

A typical program (for $R0, R1$ calculation registers and $R2, R3$ input registers, output in register $R0$) looks like this:

```
R0 = R1 NAND R2
R1 = R1 NAND R0
R0 = R3 NAND R1
R1 = R1 NAND R2 (*)
R1 = R1 NAND R0 (*)
R1 = R2 NAND R1 (*)
```

which we code as the following genotype

```
012 110 031 112 110 121
```

This is different from the phenotype of that program which results after removing the introns[1] ((*)-marked code, above) to yield

```
R0 = R1 NAND R2
R1 = R1 NAND R0
R0 = R3 NAND R1
```

which we code as the following phenotype

```
012 110 031
```

Figure 14-1 depicts which functions can be reached with programs of different length up to $L = 8$. As can be seen from the figure, there is a large discrepancy between the presence of different Boolean functions, with some like "Identity" being frequently found and thus being easy, and others like "Equivalence" being seldomly found and thus being difficult. Note the complexity threshold again: Below program length 5 there is no solution to the Equivalence function.

In the following, our GP system will be set up to find the most difficult function, the "Equivalence" function, and we shall study how the system achieves this solution and what can be said about the neutral networks it uses to find it.

After introducing the representation, we have to say a few words about the search operator(s) we shall employ in our GP runs. In this contribution we decided again for the operator easiest to analyse, mutation. Whereas it can again be argued that this is not an efficient way to traverse the problem space at hand, we would counter, that at least we can understand what is going on in the system.
For illustration purposes, suppose a mutation would change a bit in the above mentioned genotype.

```
012 110 031 112 110 121 -> 012 110 031 012 110 121
```

This would mean, that the phenotype now changes, too:

```
012 110 031 -> 012 110 012
```

In other words, by switching one bit, one of the instructions has been rendered non-effective, whereas a previously non-effective one has become effective.

The evolutionary dynamics we have chosen is again a very simple one, we observe and examine runs with a population of $\mu(1 + \lambda)$ searchers, where the notation is borrowed from Evolutionary Strategies. There are μ independent searchers (providing for statistics), each one acting in an elitist way ($+$-strategy), and exploring the neighborhood with λ trials (in our case, $\lambda = 10$). If one of these neighbor states is equal or better in fitness, the searcher assumes the new state, if not, it remains where it was.

[1]The last three instructions only affect register $R1$ and not the output register $R0$.

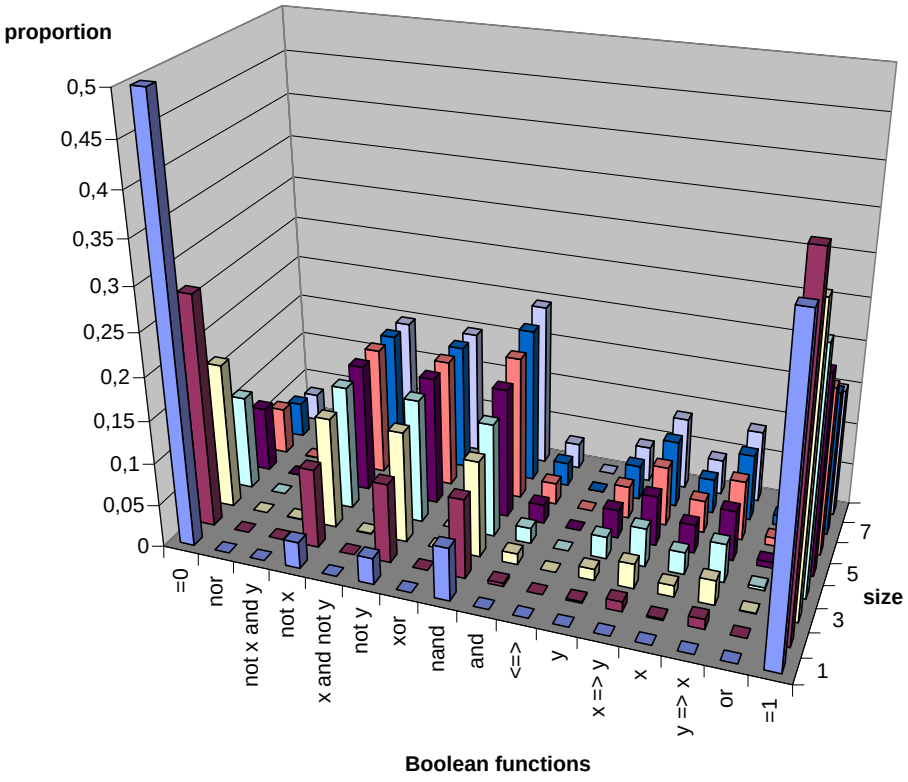

Figure 14-1. Boolean function space for various length of programs. For $L = 5$ "$\Leftrightarrow$" has a 0.00114 % share of the search space, in contrast to "$= 1$" with a share of 23.4 %. For $L < 5$ "$\Leftrightarrow$" is not present at all.

3. Non-effective Code, Neutral Networks, and the Genotype-Phenotype Map

As we have mentioned in the beginning, we expect that neutrality should play an important role in the search process in our Boolean function landscape. Neutrality is provided by non-effective code. This is unintentionally generated by a sequence of instructions if a later instruction simply overwrites what has been computed before. It might even happen that all instructions are non-effective. This is the case, if no data is written into the predetermined output register of the GP system. We refer to the corresponding phenotype as the "empty phenotype".

The Genotype-Phenotype-Mapping function is provided through removing the non-effective code. This is analogous to the neutrality provided in RNA

folding (Gruener et al., 1996). By analysing a program's code, beginning from the last line we identify those instructions which are not effective (it could be an entire block of instructions). All other instructions which will have an influence on the result of the calculation, are subsequently copied and treated as the phenotype of the program.

Table 3 shows, for an exhaustive examination of all possible genotypes in a small example, the frequency of corresponding phenotypes. This is precisely the sort of picture one encounters in RNA folding landscapes: Many very uncommon phenotypes, and few highly common phenotypes, if looked at from the point of view of enumeration of all genotypes.

Table 14-2. Redundancy of genotypes mapping into phenotypes for $C = 2; I = 2; L = 5$. The last line shows total number of genotypes and phenotypes. G: Number of genotypes; P: number of phenotypes; R=G/P: Redundancy. The fitness value relates to $\Leftrightarrow$ as the reference function.

G	*P*	*R*	*Best Fitness*	*Worst Fitness*
1,192,960	1,192,960	1	0	4
87,808	5,488	16	2	2
415,744	12,992	32	1	3
749,568	15,616	48	1	3
948,224	14,816	64	1	4
1,030,400	12,880	80	1	3
384,000	4,000	96	1	2
100,352	392	256	2	2
657,408	856	768	1	2
1,413,120	920	1,536	1	2
2,560,000	1,000	2,560	1	2
405,504	144	2,816	1	3
1,753,088	428	4,096	2	3
917,504	56	16,384	2	2
4,096,000	100	40,960	2	2
131,072	2	65,536	2	2
4,259,840	40	106,496	1	2
3,276,800	10	327,680	2	2
1,048,576	1	1,048,576	2	2
8,126,464	4	2,031,616	2	3
33,554,432	1,262,705			

Each genotype can be considered a node in a graph. A mutation would then provide a link between nodes in the graph, allowing evolution to move if this step is actually allowed by selection. Due to the genotype-phenotype mapping, however, there is also a graph of nodes constituting the network of phenotypes. Each of these nodes has a particular fitness depending on how the fitness function was defined for the problem. A movement on the genotype

network driven by mutation now induces a corresponding movement on the phenotype network. Figure 14-2 shows the graph of phenotypes in a Boolean problem small enough that all phenotypes can be enumerated and drawn (length of programs: 2 instructions only).

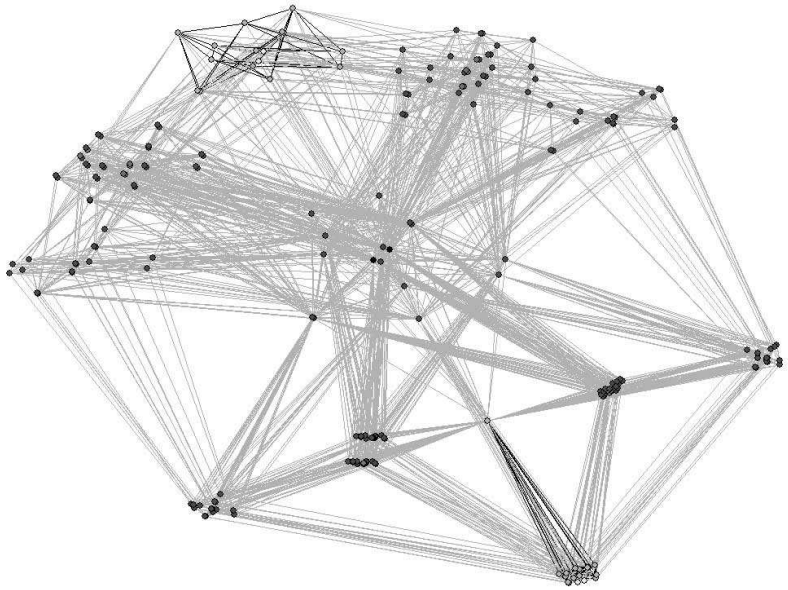

Figure 14-2. Phenotype network graph for a Boolean function problem with $C = 2; I = 2; L = 2$. Nodes have different colors, depending on the particular fitness they represent which is calculated as the difference to the AND function. Two neutral networks are shown with black edges. Self-connections of nodes are not shown.

The links between nodes correspond, as we said, to mutations, except that we have not shown self-connections which may still have a substantial impact on evolutionary search. These links are distributed unequally between nodes, induced by the genotype-phenotype mapping (GPM).

Neutral networks are constituted by those nodes in the network which have the same fitness and are connected by mutations. Note that there is a difference between this definition of neutrality and the definition used by e.g. (Ebner et al., 2002). Here we consider all phenotypes with the same fitness to be in the same neutral network, provided there is a mutational link. Ebner et al. consider neutral networks only between the same phenotypes (which surely will have the same fitness). There are two disconnected components of the neutral network to the second-best fitness level.

Strictly speaking, the phenotype network has no *direct* meaning for the evolutionary search. Our GPM is a simple many-to-one projection and the connectivity of nodes on a path in the phenotype network is not necessarily related to the path in the genotype network. That is to say, some phenotype nodes are hiding the fact that the genotypes represented by them are actually not connected at all. Therefore, the connectivity distribution of the phenotype network seems to be only of minor interest. We shall address this problem later again by suggesting another way of forming phenotypes.

4. Connectivity of Neutral Networks and Population Dynamics

It is interesting to study the connectivity of neutral networks, and relate it to the dynamics of a population of searchers on the network. The reason is that, as is well known from the study of random walks on graphs, those nodes in the network which have the highest connectivity tend to be visited the most. This is a simple Markov chain result (Lovacz, 1993; Noh and Rieger, 2004), and it leads to the following prediction: The search in the neutral network will not be a pure random drift. It will have a bias, and will concentrate on those nodes of the network where connectivity is highest. If in the mutation neighborhood of those nodes a node with a better fitness can be found, it will be discovered quickly. This can be captured by saying that the nodes of the neutral network have a different effective fitness (Nordin and Banzhaf, 1995; Banzhaf et al., 1998; Stephens and Vargas, 2000; Banzhaf and Langdon, 2002), and those nodes with a higher connectivity will have a higher effective fitness.

As has been pointed out (Schuster et al., 1994), it can be safely assumed that neutral networks for different levels of fitness are strongly intertwined. I.e. it will not be difficult to encounter transition nodes from one of these networks to another with a higher fitness. These so-called portal nodes (Nimwegen et al., 1998) are spread throughout the network and provide ample chance to jump off a neutral network onto one with better fitness. The only problem in our Boolean example is that in fact the problem is so easy (only 5 different fitness values) that it is difficult to observe all the phenomena. By looking at Figure 14-3 we can compare an exhaustive mapping of the search space in terms of connectivity characteristics with a mapping based on 100,000 GP runs. With this amount of sampling, the GP runs are already approaching full knowledge of the search space.

Connectivity characteristics lends itself as a new way of observing the system, and allows an alternative definition of phenotypes. The only condition of these phenotypes will be that the fitness of an individual should be carried by the phenotype. So our alternative phenotypes look like this: $(fitness, N, I)_i$ for

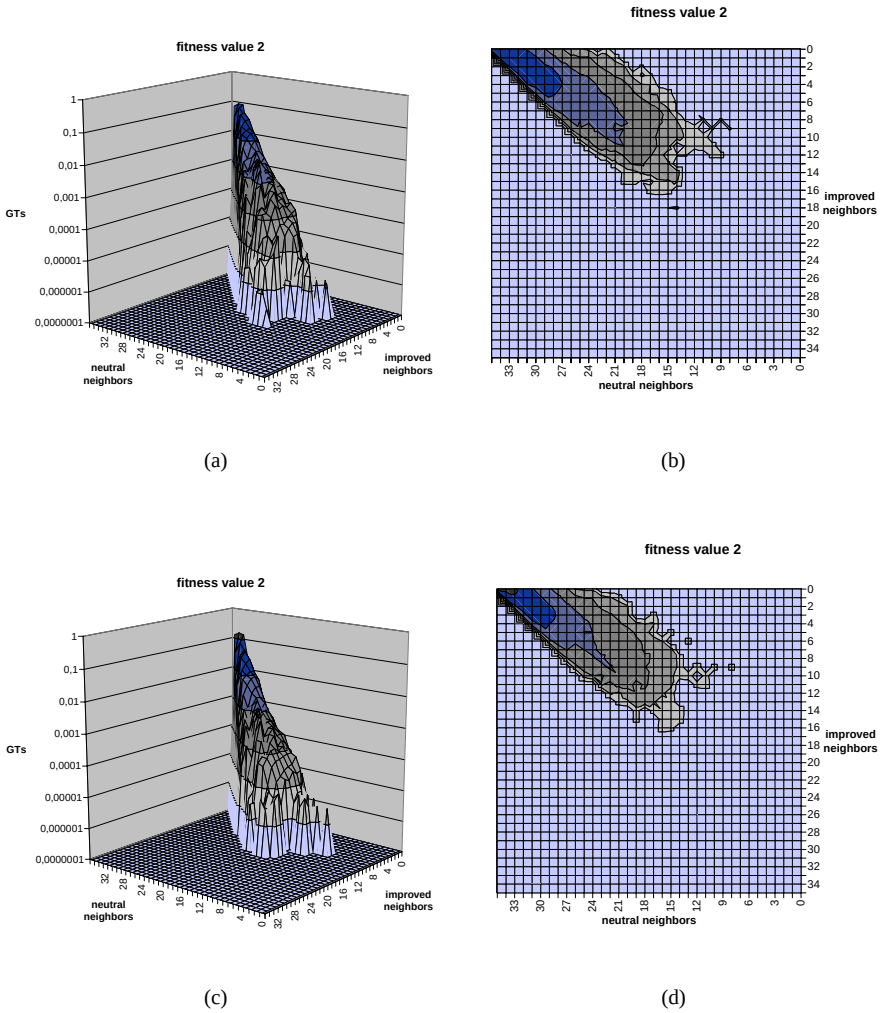

Figure 14-3. Distribution according to connectivity characteristics: A genotype's connectivity characteristics is given by a triplet of values (I, N, D) where I (D) is the number of neighbors with improved (deteriorated) fitness and N the number of neutral neighbors. Since the total number of neighbors is constant (35), two values (here: I and N) are sufficient for characterization. The 3D/2D plots show the proportions of connectivity for all genotypes of fitness 2 in the genotype network (Figures (a) and (b)) and for all visited nodes of fitness 2 within $100,000$ GP runs (Figures (c) and (d)).

individual i, where N is the number of neutral connections and I is the number of improving connections of the individual node.

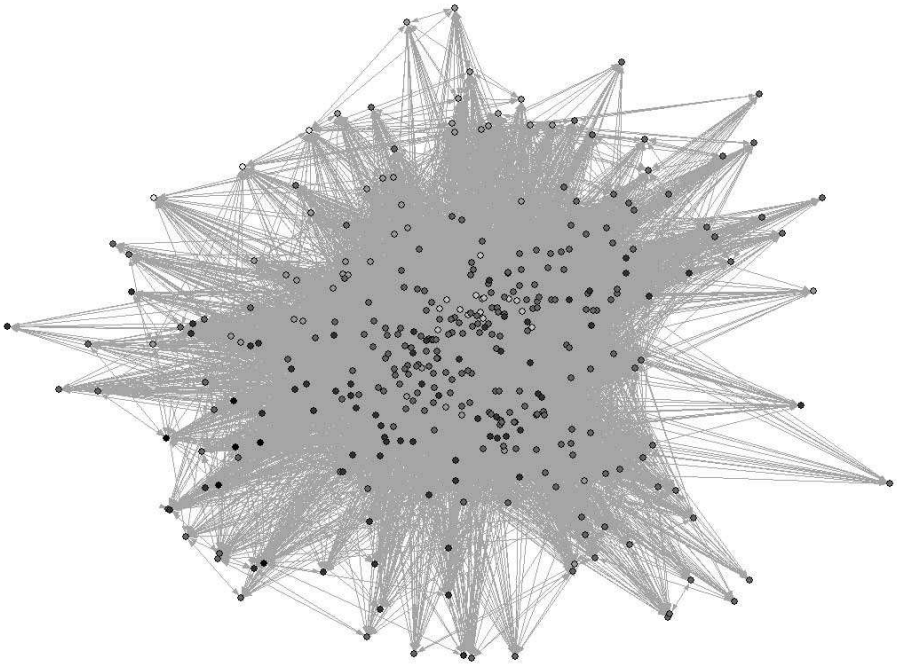

Figure 14-4. The alternative phenotype definition allows to visualize a PT network. Node colors reflect the fitness levels from high fitness (white) to low fitness (dark gray). Pale nodes in the network center correspond to nodes in the subnetwork depicted in next figure. Three nodes with fitness 0 (perfect solutions) lie in the upper left corner of the network. Fruchterman-Reingold algorithm (2D) was used to create graphs.

5. Robustness and Evolvability

Two of the main functions of neutrality in biological systems are considered to be (i) robustness of phenotypes against mutation and (ii) evolvability. For (i) to work, a viable genotype would try to locate itself in the center of a neutral network such as to make sure that any mutation that might happen to it still allows it to stay on the neutral network. In the absence of neutrality, a viable genotype/phenotype pair might always stand a high probability to produce deleterious mutations.

The other function is to provide more potential for evolvability. Following Kirschner and Gerhardt (Kirschner and Gerhart, 1998) evolvability can be defined as the capacity of an organism to generate heritable variation. It is interesting to note that modern metazoa seem to have developed in that direction.

In the context of evolutionary computation this would come about by allowing genotype/phenotype pairs to escape local optima through higher di-

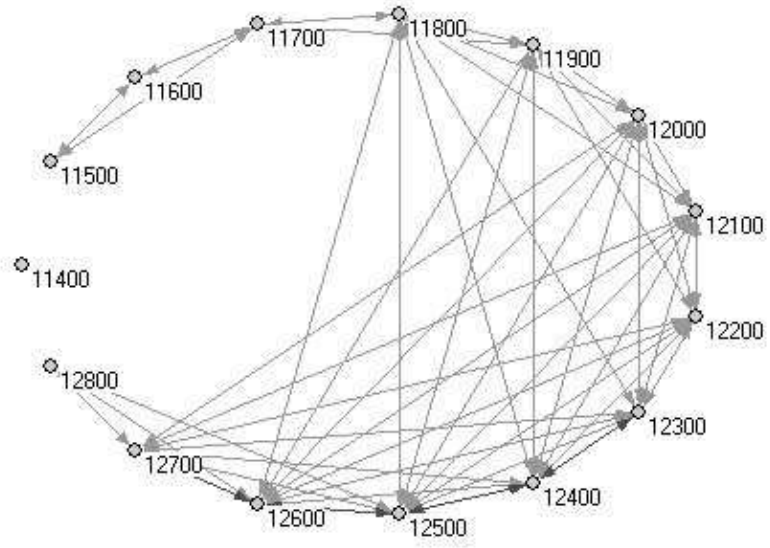

Figure 14-5. Neutral network of the most frequently visited nodes. More than 95% of all edges in the PT network passed during 1,000 GP runs belong to this subnetwork. Node labels specify fitness value (one digit), number of neutral neighbors (two digits) and number of improved neighbors (two digits). Self-connections are not shown, although they contribute over 50% .

mensional saddles, produced by neutral changes to the pair. Furthermore, if the network provides a clear guide via effective fitness, it could accelerate evolution even in the case of not being caught in a local minimum. Evolution would most probably be attracted to genotypes/phenotypes which are highly connected in the network, and thus have a better chance to be connected to higher-fitness states.

Another aspect of evolvability - not discussed here - is modularity (Altenberg, 1994b; Wagner and Altenberg, 1996). For this to work, a clearer picture of what building blocks are should be developed. We feel that more research needs to be done on the question of building blocks in GP before this question can be approached. For recent progress in this field, see (Langdon and Banzhaf, 2005).

6. Suggestions for Future Work, Summary and Conclusions

We have shown, in the context of a very simple linear GP system, that neutral mutations play an important role in setting the system up for exploration. We argue that the situation in this type of a GP system is analogous to what can be found in RNA-folding and optimization: There are many uncommon phenotypes, and just a few very common ones. From this we concluded that neutral networks must be highly intertwined such as to allow a quick transition from one neutral network to the next, through certain portal nodes.

By exhaustively enumerating solutions for a small Boolean logic problem we have demonstrated these ideas. The problem space is by no means considered to be difficult. Yet, by choosing the most difficult Boolean function to be realized in the system, we have at least made every effort possible to make it "relatively" difficult.

Unfortunately, systems like the present are combinatorial and do not lend themselves to exhaustive search very easily, except for the smallest choice of parameters. It would be interesting, for example to analyse the networks of $C, I > 2$. As Table 2 illustrates, however, this becomes quickly infeasible.

Notwithstanding the problem of exhaustive examination, we plan to analyse networks locally, around local optima or best fitness phenotypes found so far. We also want to provide more thorough statistical measures of network characteristics, such as centrality of neutral networks etc. It would be most interesting to be able to pinpoint the nodes which most searchers have to pass through and to manipulate the search in order to either lead it towards these nodes or away from them.

Acknowledgments

The authors wish to thank NSERC for support under Discovery grant RGPIN 283304-04. Software used to visualize our networks: Pajek 1.0 by Vladimir Batgelj & Andrej Mrvar

`http://vlado.fmf.uni-lj.si/pub/networks/pajek`

References

Altenberg, Lee (1994a). Emergent phenomena in genetic programming. In Sebald, Anthony V. and Fogel, Lawrence J., editors, *Evolutionary Programming — Proceedings of the Third Annual Conference*, pages 233–241, San Diego, CA, USA. World Scientific Publishing.

Altenberg, Lee (1994b). The evolution of evolvability in genetic programming. In Kinnear, Jr., Kenneth E., editor, *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press.

Angeline, Peter John (1994). Genetic programming and emergent intelligence. In Kinnear, Jr., Kenneth E., editor, *Advances in Genetic Programming*, chapter 4, pages 75–98. MIT Press.

Babajide, A., Hofacker, I.L., Sippl, M.J., and Stadler, P.F. (1997). Neutral networks in protein space. *Fold. Des.*, 2:261–269.

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

Banzhaf, W., Nordin, P., Keller, R., and Franconce, F. (1998). *Genetic Programming - An Introduction*. Morgan Kaufmann, San Francisco, CA.

Banzhaf, Wolfgang (1994). Genotype-phenotype-mapping and neutral variation – A case study in genetic programming. In Davidor, Yuval, Schwefel, Hans-Paul, and Männer, Reinhard, editors, *Parallel Problem Solving from Nature III*, volume 866 of *LNCS*, pages 322–332, Jerusalem. Springer-Verlag.

Barnett, Lionel (2001). Netcrawling-optimal evolutionary search with neutral networks. In *Proceedings of the 2001 Congress on Evolutionary Computation, 2001*, pages 30 – 37. IEEE Press.

Brameier, Markus and Banzhaf, Wolfgang (2001). A comparison of linear genetic programming and neural networks in medical data mining. *IEEE Transactions on Evolutionary Computation*, 5(1):17–26.

Ebner, M., Shackleton, M., and Shipman, R. (2002). How neutral networks influence evolvability. *Complexity*, 7:19—33.

Forst, C.V., Reidys, C., and Weber, J. (1995). Evolutionary dynamics and optimization: Neutral networks as model-landscapes for rna secondary-structure folding-landscapes. In *Advances in Artificial Life, Proc ECAL 1995*. Springer-Verlag, LNAI Vol 929.

Gruener, W., Giegerich, R., Strothmann, D., Reidys, C.M., Weber, J., Hofacker, I.L., Stadler, P.F., and Schuster, P. (1996). Analysis of dna sequence structure maps by exhaustive enumeration - part i: Neutral networks. *Monatsh. hemie*, 127:355 – 377.

Huynen, M., Stadler, P.F., and Fontana, W. (1996). Smoothness within ruggedness: The role of neutrality in adaptation. *Proc. Natl. Acad. Sci. USA*, 93:397—401.

Kimura, Motoo (1983). *The Neutral Theory of Molecular Evolution*. Cambridge University Press.

Kirschner, M. and Gerhart, J. (1998). Evolvability. *Proc. Natl. Acad. Science (USA)*, 95:8420—8427.

Koza, John R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Langdon, W. B. and Poli, R. (1998). Fitness causes bloat: Mutation. In Banzhaf, Wolfgang, Poli, Riccardo, Schoenauer, Marc, and Fogarty, Terence C., editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391 of *LNCS*, pages 37–48, Paris. Springer-Verlag.

Langdon, W. B. and Poli, R. (1999). Boolean functions fitness spaces. In Poli, Riccardo, Nordin, Peter, Langdon, William B., and Fogarty, Terence C., editors, *Genetic Programming, Proceedings of EuroGP'99*, volume 1598 of *LNCS*, pages 1–14, Goteborg, Sweden. Springer-Verlag.

Langdon, William B. and Banzhaf, Wolfgang (2005). Repeated sequences in linear genetic programming genomes. *Complex Systems*. in press.

Lovacz, L. (1993). Random walks on graphs: A survey. Technical report, Department of Computer Science, Yale University, CT, USA.

Nimwegen, E.v., Crutchfield, J.P., and Huynen, M. (1998). Neutral evolution of mutational robustness. *Proc. Natl. Acad. Sci. USA*, 96:9716—9720.

Noh, J.D. and Rieger, H. (2004). Random walks on complex networks. *Phys. Rev. Lett.*, 92:118701–1–3.

Nordin, Peter and Banzhaf, Wolfgang (1995). Complexity compression and evolution. In Eshelman, L., editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, Pittsburgh, PA, USA. Morgan Kaufmann.

Reidys, C.M., Stadler, P.F., and Schuster, P. (1997). Generic properties of combinatory maps-neutral networks of rna secondary structures. *Bull. Math. Biol*, 59:339—397.

Schultes, E.A. and Bartel, D.P. (2000). One sequence, two ribozymes: Implications for the emergence of new ribozyme folds. *Science*, 289:448—452.

Schuster, P., Fontana, W., Stadler, P.F., and Hofacker, I.L. (1994). From sequences to shapes and back: A case study in rna secondary structures. *Proc. Roy. Soc. Lond. B*, 255:279—284.

Schuster, Peter (1995). Extended molecular evolutionary biology: Articial life bridging the gap between chemistry and biology. In Langton, C.G., editor, *Artificial Life: An Overview*, pages 39 – 60. MIT Press, Cambridge, MA.

Soule, Terence, Foster, James A., and Dickinson, John (1996). Code growth in genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 215–223, Stanford University, CA, USA. MIT Press.

Soule, Terence and Heckendorn, Robert B. (2002). An analysis of the causes of code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 3(3):283–309.

Stephens, C. R. and Vargas, J. Mora (2000). Effective fitness as an alternative paradigm for evolutionary computation I: General formalism. *Genetic Programming and Evolvable Machines*, 1(4):363–378.

T. Smith, Ph. Husbands and O'Shea, M. (2001). Neutral networks in an evolutionary robotics search space. In *Proceedings of the 2001 Congress on Evolutionary Computation, 2001*, pages 136 – 145. IEEE Press.

Vassilev, Vesselin K., Fogarty, Terence C., and Miller, Julian F. (2003). Smoothness, ruggedness and neutrality of fitness landscapes: from theory to application. In Ghosh, Ashish and Tsutsui, Shigeyoshi, editors, *Advances in evolutionary computing: theory and applications*, pages 3–44. Springer-Verlag New York, Inc.

Vassilev, Vesselin K. and Miller, Julian F. (2000a). The advantages of landscape neutrality in digital circuit evolution. In *Proceedings of the Third International Conference on Evolvable Systems*, pages 252–263. Springer-Verlag.

Vassilev, Vesselin K. and Miller, Julian F. (2000b). Embedding landscape neutrality to build a bridge from the conventional to a more efficient three-bit multiplier circuit. In Whitley, Darrell, Goldberg, David, Cantu-Paz, Erick, Spector, Lee, Parmee, Ian, and Beyer, Hans-Georg, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, page 539, Las Vegas, Nevada, USA. Morgan Kaufmann.

Wagner, G.P. and Altenberg, L. (1996). Complex adaptations and the evolution of evolvability. *Evolution*, 50:967—976.

Yu, Tina and Miller, Julian (2001). Neutrality and the evolvability of boolean function landscape. In Miller, Julian F., Tomassini, Marco, Lanzi, Pier Luca, Ryan, Conor, Tettamanzi, Andrea G. B., and Langdon, William B., editors, *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *LNCS*, pages 204–217, Lake Como, Italy. Springer-Verlag.