To appear as:

Spector, L., and J. Klein. 2005. Trivial Geography in Genetic Programming. In *Genetic Programming Theory and Practice III,* edited by T. Yu, R.L. Riolo, and B. Worzel, pp. 109–124. Boston, MA: Kluwer Academic Publishers.

Chapter 8

# TRIVIAL GEOGRAPHY IN GENETIC PROGRAMMING

Lee Spector[1] and Jon Klein[1,2]

[1]*Cognitive Science, Hampshire College, Amherst, MA, 01002-3359 USA.;* [2]*Physical Resource Theory, Chalmers University of Technology & Göteborg University, Göteborg, Sweden.*

**Abstract** Geographical distribution is widely held to be a major determinant of evolutionary dynamics. Correspondingly, genetic programming theorists and practitioners have long developed, used, and studied systems in which populations are structured in quasi-geographical ways. Here we show that a remarkably simple version of this idea produces surprisingly dramatic improvements in problem-solving performance on a suite of test problems. The scheme is trivial to implement, in some cases involving little more than the addition of a modulus operation in the population access function, and yet it provides significant benefits on all of our test problems (ten symbolic regression problems and a quantum computing problem). We recommend the broader adoption of this form of "trivial geography" in genetic programming systems.

**Keywords:** geography, locality, demes, symbolic regression, quantum computing

## 1. Geography

All biological populations are distributed in space, with the result that some organisms are close neighbors while others live at great distances from one another. It has long been recognized that such geographical distribution, even in uniform environments, can influence evolutionary dynamics in significant and complex ways (Mayr, 1942, Wright, 1945, Avise, 2000, Lieberman et al., 2005). In particular, positive influences of geographical distribution on the evolution of individuals with certain desirable features (e.g. altruistic behavior) have been demonstrated in both analytical models and simulations (Eshel, 1972, Nowak and May, 1992, Axelrod et al., 2004, Spector and Klein, 2005a).

It is therefore not surprising that many evolutionary computation systems also model some form of geography, locating their evolving individuals within grid-based or continuous virtual spaces. This is a particularly natural move for systems that are designed to model aspects of natural ecosystems (Ray, 1991, Holland, 1995, Ofria and Wilke, 2004). But it is also a popular move in problem-solving evolutionary computation systems, in the context of which geography is often justified by the ways in which it can be used to maintain population diversity.

Standard genetic algorithms and genetic programming techniques are non-spatial in their most common formulations (Holland, 1992, Koza, 1992, Banzhaf et al., 1998). However, many researchers and practitioners routinely divide their populations in to discrete or overlapping sub-populations, often called *demes*, that provide a form of geography (Collins and Jefferson, 1991). In these systems selection and competition takes place locally but selected individuals occasionally mate or migrate across demes. Because the computations taking place in different demes are generally independent—particularly when the demes are non-overlapping, in which case they are sometimes called "islands"—one can often run them on independent processors and reap benefits both of parallelism and of the diversity maintenance supported by geographical distribution (Maruyama et al., 1993, Nowostawski and Poll, 1999, Andre and Koza, 1996).

Demes have been demonstrated, in certain cases, to improve problem solving performance (see e.g. (Collins and Jefferson, 1991, Fernandez et al., 2003)). A wide range of connectivity patterns and migration regimes has been discussed in the literature, and there are initial results linking specific connectivity patterns to expected performance on specific problems (Bryden et al., 2005).

In this chapter we present a form of geography that is considerably simpler than those generally used in genetic programming. Our *trivial geography* model is a 1-dimensional "overlapping neighborhoods" model that implements a concept of geography similar to that used in many artificial life simulations (Ray, 1991, Ofria and Wilke, 2004, Axelrod et al., 2004). It is also similar in many respects to the "local selection" genetic algorithm of Collins and Jefferson (1991); although their work is often cited as inspiration for the use of isolated demes with migration, the individuals in their model were actually distributed across 1-dimensional or 2-dimensional grids, with one individual per grid location, and selection and mating were performed in local areas of the grid. For example, short random walks through the grid were used to pair mates. A more recent genetic programming model, known as "cellular" or "diffusion" genetic programming, locates individuals on a 2-dimensional grid and allows interactions only between immediate neighbors (Pettey, 1997, Folino et al., 1999, Folino et al., 2003). Several other models involving related notions of locality have been used in other genetic programming work, often in the con-

text of additional innovations (e.g. co-evolution or autoconstructive evolution) (D'haeseleer and Bluming, 1994, Spector, 2001).

Trivial geography requires no explicit representation of demes, connectivity patterns, or migration rates. It requires only minimal changes to a standard genetic programming system and a single new parameter. The question we set out to investigate was whether such a minimal form of geography could make much of a difference with respect to problem-solving performance, and if so what that difference might be. Our data show that trivial geography does indeed appear to make a substantial positive difference, improving problem-solving performance.

In the next section we describe our concept of trivial geography and its simple implementation. This is followed by two sections demonstrating the utility of trivial geography, first on a suite of ten symbolic regression problems and then on a difficult problem in quantum computing. We follow these demonstrations with a general discussion and a recommendation that trivial geography be incorporated into genetic programming systems more broadly.

## 2. Trivial Geography

In our trivial geography scheme the population is viewed as having a 1-dimensional spatial structure—actually a circle, as we consider the first and last locations to be adjacent. The production of an individual for location $i$ is permitted to involve only parents from $i$'s local neighborhood, where the neighborhood is defined as all individuals within distance $R$ (the neighborhood radius) of $i$. Aside from this restriction no changes are made to the genetic programming system.

This scheme can be applied to most standard genetic programming systems with very little effort. Since most systems store their populations in 1-dimensional data structures (arrays or lists) anyway, all that is required is that one restrict the selection of parents relative to the index of a child.

To avoid conflation of geography and genetic operators we assume that genetic operators are chosen independently of location. Presumably the operators are chosen randomly, with biases incorporated into the random choice to achieve desired operator ratios. This is indeed a common implementation strategy (used, for example, in ECJ[1]), although in some implementations (e.g. that described in (Koza, 1992)) a particular genetic operator is applied to produce the first segment of the population, another operator is applied to produce the next segment, and so on. Under such an implementation operators would be restricted to certain geographic areas and one can imagine that strange dynamics

[1] `http://cs.gmu.edu/~eclab/projects/ecj/`

*Table 1-1.* Symbolic regression problems used for tests of trivial geography.

| # | *Problem* |
|---|---|
| 1 | $y = 8x^3 + 3x^2 + x$ |
| 2 | $y = 7x^3 - 3x^2 + 17x$ |
| 3 | $y = 5x^3 + 12x^2 - 3x$ |
| 4 | $y = x^3 + x^2 + x$ |
| 5 | $y = x^3 - 2x^2 - x$ |
| 6 | $y = 8x^5 + 3x^3 + x^2 + 6$ |
| 7 | $y = 7x^4 - 6x^3 + 3x^2 + 17x - 3$ |
| 8 | $y = 5x^6 - 2x^5 - 5x^3 + 3x^2 + 5$ |
| 9 | $y = x^4 + x^3 + x^2 + x - 8$ |
| 10 | $y = x^6 - 2x^4 + x^2 - 2$ |

would result; one would probably want to convert first to location-independent operator selection, which is itself usually a simple modification.

While trivial geography can be used with various selection schemes it is particularly simple to describe in terms of tournament selection. In this context it can be implemented simply by changing the function that chooses a random individual to participate in a tournament. Whereas the standard scheme chooses each such individual randomly from the entire population, in trivial geography we choose each such individual from the neighborhood of the location for which we are creating a new individual. In particular we choose only from individuals with indices in the range $(i - R, i + R)$, where $i$ is the index of the location for which we are creating an individual, $R$ is a *radius* parameter, and we "wrap around" from the bottom to the top of the range and vice versa.[2] The modification to restrict the range of choices is indeed often trivial, involving only one or a few lines of code.

## 3. Trivial Geography Applied to Symbolic Regression

We tested trivial geography on the ten arbitrarily chosen symbolic regression problems listed in Table 1-1. We used the PushGP genetic programming system, which evolves programs in the Push language (Spector, 2001, Spector and Robinson, 2002, Spector et al., 2005).[3] Push is a multi-type, stack-based programming language that supports the evolution of novel control structures through explicit code and control manipulation, but none of these novel fea-

[2]In some programming languages this "wrapping around" can be accomplished with a single call to the modulus function.

[3]`http://hampshire.edu/lspector/push.html`

*Table 1-2.* Parameters for symbolic regression tests of trivial geography. The instruction set is limited to simple integer manipulation and integer stack manipulation. The `INPUT` instruction pushes the current input ($x$) value onto the integer stack.

| | |
|---|---|
| Problems | Symbolic regression problems listed in Table 1-1. |
| Input ($x$) values | 0–9 |
| Fitness | Sum of absolute values of errors for all inputs. |
| Runs per problem | 115 with trivial geography, 115 without trivial geography. |
| Radius ($R$) | 10 |
| Population size | 2000 |
| Crossover rate | 40% |
| Mutation rate | 40%, fair mutation (Crawford-Marks and Spector, 2002) |
| Duplication rate | 20% |
| Tournament size | 7 |
| Maximum generations | 200 |
| Initial program size limit | 100 |
| Child program size limit | 100 |
| Program evaluation limit | 100 |
| Ephemeral random constants | integer $(-10, 10)$ |
| Instructions | `INTEGER.+`, `INTEGER.-`, `INTEGER.*`, `INTEGER./`, `INTEGER.POP`, `INTEGER.DUP`, `INTEGER.SWAP`, `INTEGER.SHOVE`, `INTEGER.YANK`, `INTEGER.YANKDUP`, `INPUT` |

tures were used in the present study. For the experiments reported here we used only a minimal integer-oriented instruction set, so that PushGP was acting here much like any standard genetic programming system.[4] We have no reason to believe that the remaining differences between PushGP and other genetic programming systems contributed to our results in any significant way. The full set of parameters used for our runs is presented in Table 1-2.

We examined the results in two ways, looking both at the "computational effort" required to find a solution (Koza, 1994) and the mean best fitness across all runs on a particular problem. Computational effort was computed as described by Koza (pp. 99–103), first calculating $P(M, i)$, the cumulative probability of success by generation *i* using a population of size *M* (this is just the total number of runs that succeeded on or before the *i*th generation, divided by the total number of runs conducted). $I(M, i, z)$, the number of individuals that must be processed to produce a solution by generation *i* with probability greater than *z* (by convention, $z$ =99%) is then calculated as:

[4]We used the version of PushGP distributed with the Breve simulation environment (Klein, 2002). Breve is available from `http://www.spiderland.org/breve`.

*Table 1-3.* Successes/runs and computational efforts for the symbolic regression problems with and without trivial geography.

| # | *Successes/runs without trivial geography* | *Effort without trivial geography* | *Successes/runs with trivial geography* | *Effort with trivial geography* |
|---|---|---|---|---|
| 1 | $67/115$ | $600,000$ | $113/115$ | $316,000$ |
| 2 | $24/115$ | $3,024,000$ | $64/115$ | $2,176,000$ |
| 3 | $8/114$ | $12,566,000$ | $50/115$ | $3,160,000$ |
| 4 | $115/115$ | $36,000$ | $115/115$ | $30,000$ |
| 5 | $106/115$ | $132,000$ | $115/115$ | $66,000$ |
| 6 | $17/115$ | $5,928,000$ | $76/113$ | $1,840,000$ |
| 7 | $2/114$ | $54,810,000$ | $6/114$ | $38,406,000$ |
| 8 | $0/113$ | $\infty$ | $1/113$ | $144,282,000$ |
| 9 | $73/113$ | $848,000$ | $113/113$ | $276,000$ |
| 10 | $101/113$ | $280,000$ | $113/113$ | $164,000$ |

$$I(M,i,z) = M * (i+1) * \left\lceil \frac{\log(1-z)}{\log(1-P(M,i))} \right\rceil$$

The minimum of $I(M,i,z)$ over all values of $i$ is defined to be the "computational effort" required to solve the problem.

The computational efforts calculated from our $2,283$ runs ($115$ runs for each of the $2$ conditions for each of the $10$ problems, with $17$ runs lost to miscellaneous system problems) are shown in Table 1-3. Lower efforts are, of course, better, so this data demonstrates that trivial geography provides a considerable benefit on all of the symbolic regression problems.

Because the problems vary widely in difficulty we also show, in Figure 1-1, a graph of these results normalized independently for each problem, with the effort for the standard configuration (without trivial geography) set to $100$; the values for the runs with trivial geography therefore indicate the computational effort as a percentage of that in the standard configuration. From this graph it is clear that the benefits provided by trivial geography are indeed substantial.

The mean best fitness values from our runs are shown in Table 1-4. Lower fitness values are better, so this data also demonstrates that trivial geography provides a considerable benefit for all of the symbolic regression problems. We also show, in Figure 1-2, a graph of these results normalized for each problem, with the mean best fitness for the standard configuration (without trivial geography) set to $100$; the values for the runs with trivial geography therefore indicate the mean best fitness as a percentage of that in the standard configuration. For problems #5, #9 and #10 trivial geography achieved a 100% solution rate (best fitness = 0 for all runs). Problem #4 was exceptionally

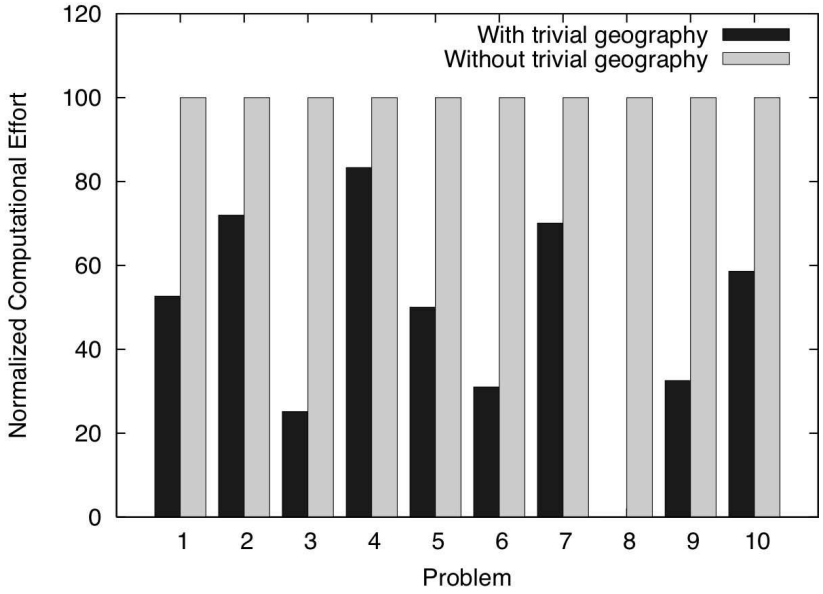


*Figure 1-1.* Computational efforts calculated for the symbolic regression problems with and without trivial geography. This plot is normalized independently for each problem, with the values for runs in the standard configuration (without trivial geography) shown as 100%. Problem #8 is anomalous because no solutions were found without trivial geography, producing an infinite computational effort.

easy, leading to 100% solution rates in both configurations; both are therefore plotted as 100%. From the mean best fitness values it is also clear that the benefits provided by trivial geography are indeed substantial.

For the mean best fitness values we conducted T tests to assess the statistical significance of the differences between the configurations with and without trivial geography. Aside from problem #4 (the problem on which both configurations achieved 100% solution rates) all differences are significant with $p < 0.01$.

## 4. Trivial Geography Applied to a Quantum Computing Problem

Quantum information technology is expected to provide revolutionary benefits for computing, but quantum computers are counter-intuitive and difficult to program. Genetic programming can be used to automatically develop quantum computing algorithms, and the resulting algorithms may be useful both for solving practical problems and for answering open questions in the the-

*Table 1-4.* Mean best fitness values (for which lower values are better) for the symbolic regression problems with and without trivial geography.

| # | *Mean best fitness without trivial geography* | *Mean best fitness with trivial geography* |
|---|---|---|
| 1 | 52.50 | 0.65 |
| 2 | 98.67 | 19.13 |
| 3 | 148.77 | 48.39 |
| 4 | 0 | 0 |
| 5 | 5.51 | 0 |
| 6 | 7,149.94 | 63.19 |
| 7 | 957.43 | 332.48 |
| 8 | 27,475.48 | 16,859.71 |
| 9 | 22.41 | 0 |
| 10 | 1.81 | 0 |

ory of quantum computing. A detailed discussion of the application of genetic programming to quantum computing problems can be found in (Spector, 2004).

The problem we set out to solve, like many quantum computation problems, involves determining how a "black box" computational gate called an *oracle* transforms the qubits to which it is applied.[5] In particular, we were interested in determining whether a given 2-input, 1-output Boolean oracle flips its output qubit under the conditions illustrated in Figure 1-3. That is, we are asked to determine if the cases for which the oracle flips its output qubit satisfy the logical formula $(I_{00} \vee I_{01}) \wedge (I_{10} \vee I_{11})$, where $I_{ab}$ indicates whether or not the output is flipped for the input $(a, b)$.

This problem, which is called the "AND/OR" oracle problem, has been the subject of several of our previous investigations (Spector et al., 1999, Barnum et al., 2000, Spector, 2004). We previously used genetic programming to find quantum algorithms that perform better than any possible classical algorithm (that is, they have lower probability of error) when restricted to a single oracle call. In our more recent work we have been investigating the two-oracle-call version of this problem. The lowest error probability obtainable by a probabilistic classical algorithm on the two-oracle-call version of this problem is $\frac{1}{6} = 0.1666...$, but in our recent work we have found, using genetic programming, quantum algorithms with an error probability of less than $0.11$ (Spector and Klein, 2005b).

[5]A qubit is the quantum analog of a classical "bit"; see (Spector, 2004) for a detailed description of qubits and the ways in which they are manipulated by quantum gates.

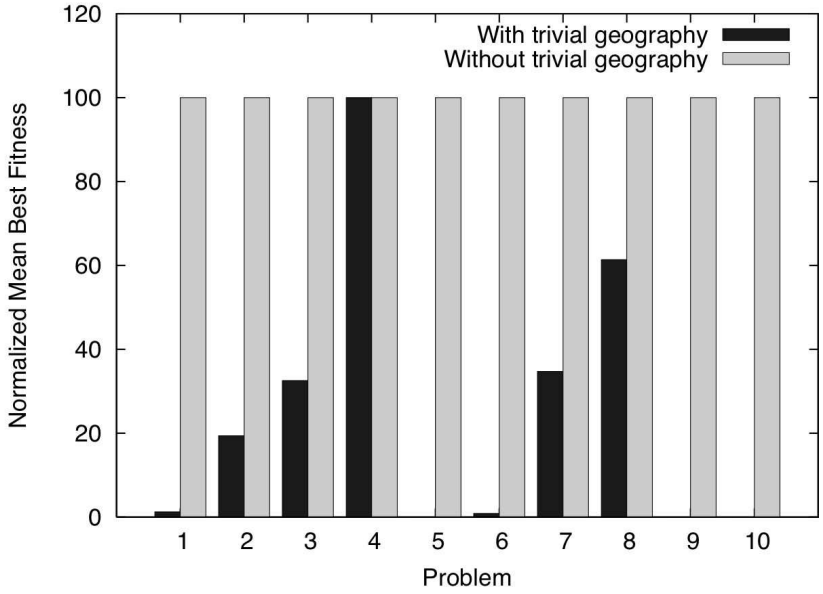


*Figure 1-2.* Mean best fitness values (for which lower values are better) for the symbolic regression problems with and without trivial geography. This plot is normalized independently for each problem, with the values for runs in the standard configuration (without trivial geography) shown as 100%. For problems #5, #9 and #10 trivial geography achieved a 100% solution rate (fitness = 0 for all runs). For problem #4 both configurations achieved a 100% solution rate.

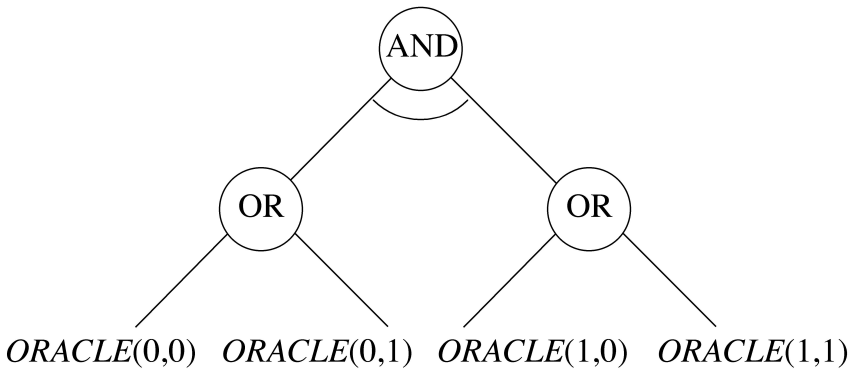


*Figure 1-3.* An "AND/OR" tree describing the property of interest in the AND/OR oracle problem (see text).

Our new results on the two-oracle-call AND/OR problem used trivial geography, and our anecdotal evidence led us to believe that trivial geography played an important role in our success. But this work also involved intensive runs with expensive fitness tests and large populations distributed across a 23-CPU computer cluster. It was not practical to replicate runs of this scale the hundreds of times that would be necessary to fully assess the contribution of trivial geography, so we opted instead to conduct many smaller-scale runs which, while they would not solve the problem of beating the classical error probability, would still produce significant improvements in fitness.

We conducted 92 runs with and 92 runs without trivial geography, using the parameters shown in Table 1-5 and, again, the version of the PushGP genetic programming system that is distributed with the Breve simulation environment. Fitness was assessed using the QGAME quantum computer simulator, a version of which is also distributed with Breve.[6]

Computational effort is meaningful and finite only in the context of a success criterion that is reached in at least some runs. But the difficulty of this problem, relative to the resources we employed, prevented us from finding any solutions that beat the classical error probability. Since there is no other obvious choice for a success criterion we report here only a comparison of mean best fitness values.

The mean best fitness for the runs without trivial geography was $0.51$, while the mean best fitness for the runs with trivial geography was better, at $0.32$. A T test shows this difference to be statistically significant with $p < 0.005$. Again, we see a substantial improvement in problem-solving performance provided by trivial geography.

## 5. Discussion

We have presented a simple modification to the standard genetic programming technique that appears, from the tests we have run to date, to provide substantial benefits to problem-solving performance on both artificial and real-world problems. The modification incorporates notions of geographical distribution that have a long history in evolutionary biology and many precedents in genetic programming and other forms of evolutionary computation. Our modification, however, is arguably simpler to implement than any of its predecessors; in many cases it can be implemented in one or a handful of lines of code. We were surprised to find that this "trivial" form of geography nonetheless provides real benefits, and although we cannot make general claims about

[6]QGAME documentation and code is available from `http://hampshire.edu/lspector/qgame.html`.

*Table 1-5.* Parameters for quantum computing tests of trivial geography. For this problem a developmental approach was used in which certain instructions add quantum gates to a developing "embryo"; see (Spector, 2004) for details.

| | |
|---|---|
| Problem | AND/OR oracle problem (Spector, 2004), with two calls to the oracle permitted. |
| Embryo | Three-qubit quantum circuit with a final measurement on one qubit (index 2 of (0–2)). |
| Fitness cases | All possible two-input, one-output Boolean oracles, specifically ($I_{00}I_{01}I_{10}I_{11} : answer$): 0000:0, 0001:0, 0010:0, 0011:0, 0100:0, 0101:1, 0110:1, 0111:1, 1000:0, 1001:1, 1010:1, 1011:1, 1100:0, 1101:1, 1110:1, 1111:1 |
| Fitness function | $Misses + MaxError$ where $Misses$ is the number of cases for which the probability of error is greater than 0.48 and $MaxError$ is the maximum probability of error of any case. |
| Runs | 92 with trivial geography, 92 without trivial geography. |
| Radius ($R$) | 15 |
| Population size | 2500 |
| Crossover rate | 40% |
| Mutation rate | 40%, fair mutation (Crawford-Marks and Spector, 2002) |
| Duplication rate | 20% |
| Tournament size | 7 |
| Maximum generations | 500 |
| Initial program size limit | 100 |
| Child program size limit | 250 |
| Program evaluation limit | 250 |
| Ephemeral random constants | integer $(-10, 10)$, float $(-10.0, 10.0)$ |
| Instructions | `FLOAT.%, FLOAT.*, FLOAT.+, FLOAT.-, FLOAT./, FLOAT.DUP, FLOAT.POP, FLOAT.SWAP, FLOAT.FROMINTEGER, LIMITED-ORACLE, HADAMARD, U-THETA, MEASURE, SRN, CNOT, U2, CPHASE, SWAP, END` |

its utility[7] we recommend that trivial geography be adopted more widely in genetic programming systems.

For researchers and practitioners using genetic programming systems that already involve geographical distribution (e.g. in isolated demes with migration) an obvious practical question, not addressed here, is that of how trivial geography compares to their presumably more complex techniques. One might also be interested in the effects of combining several forms of geography, for example

[7]Such claims would require analysis and discussion of the results in the context of the No Free Lunch theorem (Wolpert and Macready, 1997, Droste et al., 1999).

by using an island model in which trivial geography is used within each island. Although comparisons of these techniques are simple to make in principle, one would have to conduct large numbers of tests using each of many geographical schemes to make definitive recommendations. Our contention here is not that trivial geography necessarily outperforms other forms of geography, but only that it appears to provide benefits over non-geographical models in many cases for nearly no cost.

The mechanism by which trivial geography improves problem-solving performance is presumably a form of diversity maintenance. An obvious follow-up study would apply diversity measures to runs like those conducted here and investigate the relations between problems, performance, and diversity. Many diversity measures for genetic programming have been developed, as have methodologies for correlating various diversity measures and aspects of system performance (Burke et al., 2004).

The values of $R$, the neighborhood radius, that we used in the experiments reported here ($10$ and $15$) were chosen somewhat arbitrarily. We conducted preliminary runs with several values of $R$ and many appeared to perform well; we chose the values that we did because they appeared to give good results, but we did not investigate other values of $R$ systematically. Our suspicion is that trivial geography will often provide benefits with a range of $R$ values and that the choice of $R$ is not critical; another obvious follow-up study would test this suspicion.

## 6. Summary

An extremely simple modification to the genetic programming algorithm, incorporating "trivial geography," appears to improve problem-solving performance for nearly no cost. This modification has many precedents in genetic programming and evolutionary computation, but it is surprising that so simple a form of the idea can have such substantial effects. We recommend that trivial geography be adopted more broadly in genetic programming.

## Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0308540 and Grant No. 0216344. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. We thank Nic McPhee, Ellery Crane, Christian Jacob, and the other participants in the Genetic Programming Theory and Practice Workshop for many helpful comments that led to substantial improvements in this chapter.

## References

Andre, David and Koza, John R. (1996). A parallel implementation of genetic programming that achieves super-linear performance. In Arabnia, Hamid R., editor, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, volume III, pages 1163–1174, Sunnyvale. CSREA.

Avise, J. C. (2000). *Phylogeography: The History and Formation of Species*. Harvard University Press.

Axelrod, R., Hammond, R. A., and Grafen, A. (2004). Altruism via kin-selection strategies that rely on arbitrary tags with which they coevolve. *Evolution*, 58(8):1833–1838.

Banzhaf, Wolfgang, Nordin, Peter, Keller, Robert E., and Francone, Frank D. (1998). *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann.

Barnum, Howard, Bernstein, Herbert J, and Spector, Lee (2000). Quantum circuits for OR and AND of ORs. *Journal of Physics A: Mathematical and General*, 33(45):8047–8057.

Bryden, Kenneth M., Ashlock, Daniel A., Corns, Steven, and Willson, Stephen J. (2005). Graph based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, forthcoming.

Burke, Edmund K., Gustafson, Steven, and Kendall, Graham (2004). Diversity in genetic programming: An analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, 8(1):47–62.

Collins, Robert J. and Jefferson, David R. (1991). Selection in massively parallel genetic algorithms. In Belew, Rick and Booker, Lashon, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 249–256, San Mateo, CA. Morgan Kaufman.

Crawford-Marks, Raphael and Spector, Lee (2002). Size control via size fair genetic operators in the PushGP genetic programming system. In Langdon, W. B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M. A., Schultz, A. C., Miller, J. F., Burke, E., and Jonoska, N., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 733–739, New York. Morgan Kaufmann Publishers.

D'haeseleer, Patrik and Bluming, Jason (1994). Effects of locality in individual and population evolution. In Kinnear, Jr., Kenneth E., editor, *Advances in Genetic Programming*, chapter 8, pages 177–198. MIT Press.

Droste, Stefan, Jansen, Thomas, and Wegener, Ingo (1999). Perhaps not a free lunch but at least a free appetizer. In Banzhaf, Wolfgang, Daida, Jason, Eiben, Agoston E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E., editors, *Proceedings of the Genetic and Evolutionary Computa-*

*tion Conference*, volume 1, pages 833–839, Orlando, Florida, USA. Morgan Kaufmann.

Eshel, I. (1972). On the neighbor effect and the evolution of altruistic traits. *Theoretical Population Biology*, 3:258–277.

Fernandez, Francisco, Tomassini, Marco, and Vanneschi, Leonardo (2003). An empirical study of multipopulation genetic programming. *Genetic Programming and Evolvable Machines*, 4(1):21–51.

Folino, G., Pizzuti, C., Spezzano, G., Vanneschi, L., and Tomassini, M. (2003). Diversity analysis in cellular and multipopulation genetic programming. In Sarker, Ruhul, Reynolds, Robert, Abbass, Hussein, Tan, Kay Chen, McKay, Bob, Essam, Daryl, and Gedeon, Tom, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 305–311, Canberra. IEEE Press.

Folino, Gianluigi, Pizzuti, Clara, and Spezzano, Giandomenico (1999). A cellular genetic programming approach to classification. In Banzhaf, Wolfgang, Daida, Jason, Eiben, Agoston E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1015–1020, Orlando, Florida, USA. Morgan Kaufmann.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press.

Holland, J. H. (1995). *Hidden Order: How Adaptation Builds Complexity*. Perseus Books.

Klein, Jon (2002). BREVE: a 3d environment for the simulation of decentralized systems and artificial life. In Standish, R. K., Bedau, M. A., and Abbass, H. A., editors, *Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems*, pages 329–334. The MIT Press.

`http://www.spiderland.org/breve/breve-klein-alife2002.pdf`.

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

Koza, John R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge Massachusetts.

Lieberman, E., Hauert, C., and Nowak, M. A. (2005). Evolutionary dynamics on graphs. *Nature*, 433:312–316.

Maruyama, Tsutomu, Hirose, Tetsuya, and Konagaya, Akihiko (1993). A fine-grained parallel genetic algorithm for distributed parallel systems. In Forrest, Stephanie, editor, *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, pages 184–190, San Mateo, CA. Morgan Kaufmann.

Mayr, Ernst (1942). *Systematics and the origin of species from the viewpoint of a zoologist*. Columbia University Press.

Nowak, M. A. and May, R. M. (1992). Evolutionary games and spatial chaos. *Nature*, 359:826–829.

Nowostawski, M. and Poll, R. (1999). Parallel genetic algorithm taxonomy.

Ofria, Charles and Wilke, Claus O. (2004). Avida: A software platform for research in computational evolutionary biology. *Artificial Life*, 10(2):191–229.

Pettey, Chrisila C. (1997). Diffusion (cellular) models. In Bäck, Thomas, Fogel, David B., and Michalewicz, Zbigniew, editors, *Handbook of Evolutionary Computation*, pages C6.4:1–6. Institute of Physics Publishing and Oxford University Press, Bristol, New York.

Ray, Thomas S. (1991). Is it alive or is it GA. In Belew, Richard K. and Booker, Lashon B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 527–534, University of California - San Diego, La Jolla, CA, USA. Morgan Kaufmann.

Spector, Lee (2001). Autoconstructive evolution: Push, pushGP, and pushpop. In Spector, Lee, Goodman, Erik D., Wu, Annie, Langdon, W. B., Voigt, Hans-Michael, Gen, Mitsuo, Sen, Sandip, Dorigo, Marco, Pezeshk, Shahram, Garzon, Max H., and Burke, Edmund, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 137–146, San Francisco, California, USA. Morgan Kaufmann.

Spector, Lee (2004). *Automatic Quantum Computer Programming: A Genetic Programming Approach*, volume 7 of *Genetic Programming*. Kluwer Academic Publishers, Boston/Dordrecht/New York/London. in press.

Spector, Lee, Barnum, Howard, Bernstein, Herbert J., and Swamy, Nikhil (1999). Finding a better-than-classical quantum AND/OR algorithm using genetic programming. In Angeline, Peter J., Michalewicz, Zbyszek, Schoenauer, Marc, Yao, Xin, and Zalzala, Ali, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 2239–2246, Mayflower Hotel, Washington D.C., USA. IEEE Press.

Spector, Lee and Klein, Jon (2005a). Genetic stability and territorial structure facilitate the evolution of tag-mediated altruism. *Artificial Life*. Forthcoming.

Spector, Lee and Klein, Jon (2005b). Machine invention of quantum computing circuits by means of genetic programming. In preparation.

Spector, Lee, Klein, Jon, and Keijzer, Maarten (2005). The push3 execution stack and the evolution of control. In *Proc. of the Genetic and Evolutionary Computation Conference*. Springer-Verlag.

Spector, Lee and Robinson, Alan (2002). Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines*, 3(1):7–40.

Wolpert, David H. and Macready, William G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Wright, Sewall (1945). Tempo and mode in evolution: a critical review. *Ecology*, 26:415–419.