Evolving Teamwork and Coordination with Genetic Programming

Sean Luke†
seanl@cs.umd.edu
http://www.cs.umd.edu/~seanl/

Lee Spector†‡
spector@cs.umd.edu
http://hamp.hampshire.edu/~lasCCS/Home.html

†Department of Computer Science
University of Maryland
College Park, MD 20742

‡School of Cognitive Science and Cultural Studies
Hampshire College
Amherst, MA 01002

ABSTRACT

Some problems can be solved only by multi–agent teams. In using genetic programming to produce such teams, one faces several design decisions. First, there are questions of team diversity and of breeding strategy. In one commonly used scheme, teams consist of clones of single individuals; these individuals breed in the normal way and are cloned to form teams during fitness evaluation. In contrast, teams could also consist of distinct individuals. In this case one can either allow free interbreeding between members of different teams, or one can restrict interbreeding in various ways. A second design decision concerns the types of coordination–facilitating mechanisms provided to individual team members; these range from sensors of various sorts to complex communication systems. This paper examines three breeding strategies (clones, free, and restricted) and three coordination mechanisms (none, deictic sensing, and name–based sensing) for evolving teams of agents in the Serengeti world, a simple predator/prey environment. Among the conclusions are the fact that a simple form of restricted interbreeding outperforms free interbreeding in all teams with distinct individuals, and the fact that name–based sensing consistently outperforms deictic sensing.

1 Teamwork and Specialization

The most common kind of task found in multi–agent problems is one in which the task can be done *faster* or *more easily* by dividing it up among many simultaneous agents. The more agents working on the problem, the larger the speedup. But some tasks may not only be solved better by using multiple agents, but can *only* be solved, or only effectively solved, by using teams of agents working together. For example, consider the problem faced by Pac Man monsters, which are slower than Pac Man on all but long straightaways. If there was only one monster then the monster would rarely catch even poor players, and the game would be trivial.

Many such problems are solvable by multiple agents each of which uses essentially the same algorithm. These *homogeneous* tasks are common candidates for the study of emergent behavior because one (often simple) set of rules can give rise to complex patterns of behavior. An example of a homogeneous task is self–replication of cellular automata. Lohn and Reggia [1995] used genetic algorithms to breed cellular automata that spontaneously emit identical copies of themselves. Each cell in the automata grid can be thought of as an individual helping to create these duplicate patterns: the problem is homogenous because all cells follow the same cellular automata rule set. An example in the genetic programming (GP) literature is the evolution of herding behaviors [Reynolds 1993]. Reynolds used GP to evolve "critters" which reacted with a herd instinct to outside predators. He evolved a single controller which moved each critter based on its position and information about its neighbors and predators. Each critter used this same controller algorithm to make movement decisions.

In contrast, *heterogeneous* tasks can only be solved (or solved easily or effectively) with multiple individuals each of which uses a distinct specialized algorithm. For example, Haynes *et al* chose a simple grid–based predator/prey game to test a variety of breeding strategies on a team of (typically) distinct GP individuals [Haynes *et al* 1995]. And Andre's MAPMAKER system required two specialized agents (one with the sensors, the other able to pick up objects) to gather gold strewn in a 2-dimensional grid [Andre 1995]. Without the interaction of two unique algorithms, this task would be much more difficult, perhaps impossible.

The distinction between homogeneous and heterogeneous approaches can be recast as an issue of breeding policy. In the homogeneous approach, individuals breed as in ordinary genetic programming, and the individuals are then cloned for fitness evaluation. In the heterogeneous approach, individual team members breed separately, which makes possible additional strategies for interbreeding within and between teams.

In many multi–agent domains, agents must coordinate their

[1]To appear in the Genetic Programming 96 (GP96) conference proceedings, Stanford, July 1996.

Table 1: Basic Lion Operators

Name	# Args	Description
`last`	0	One unit in the direction the lion went last. On the first move, return a random normal vector.
`rand-dir`	0	A random normal vector.
`gazelle`	0	The smallest vector from the lion to the gazelle.
`+`	2	Add two vectors.
`-`	1	Negate a vector.
`*2`	1	Multiply the magnitude of a vector by 2.
`/2`	1	Divide the magnitude of a vector by 2.
`->90`	1	Rotate a vector clockwise 90 degrees.
`rand`	1	Given a vector, return a vector in the same direction, whose magnitude varies randomly between 0 and the magnitude of the original vector.
`inv`	1	"Invert" a vector's magnitude, so that small vectors are "more significant" than large vectors. This is done similarly to the method used in the gazelle algorithm: Let v be the vector, and $\|max\|=\sqrt{(w/2)^2+(h/2)^2}$, where w is the width of the world (15 units), and h is the height of the world (also 15 units). Then inv returns $\frac{v}{\|v\|}(\|max\|-\|v\|)$.
`ifdot`	4	Evaluate the first and second arguments. If their dot product is greater than or equal to 0, then evaluate and return the third argument, else evaluate and return the fourth argument.
`if>=`	4	Evaluate the first and second arguments. If the magnitude of the first argument is greater than or equal to the magnitude of the second argument, then evaluate and return the third argument, else evaluate and return the fourth argument.

actions with one another. Various mechanisms may facilitate the evolution of such coordinated behavior. In some cases it is sufficient for agents to be able to sense one another in some way, though in other cases it may be necessary for agents to have more elaborate communication mechanisms. Even in the case of simple sensing, there are many possibilities. For example, one can allow each agent direct access to the positions of each distinct fellow agent, referenced by agent name; we call this *name–based sensing*. Another approach, used for example in evolving herding behaviors, allows only sensing relative to the agent— for example, "nearest neighbor agent," or "front of the pack." We call this *deictic sensing*. It may also be possible that team members can evolve to coordinate with one another in a hard–coded way with *no sensing* of neighboring agents.

This paper examines three breeding strategies (clones, free, and restricted) and three coordination mechanisms (none, deictic sensing, and name–based sensing) for evolving teams of agents in the Serengeti world, a simple predator/prey environment. We show that a simple form of restricted interbreeding outperforms free interbreeding in all teams with distinct individuals, that name–based sensing consistently outperforms deictic sensing, and that as the sensing becomes increasingly direct (more name–based), heterogeneous approaches work better than homogeneous approaches. In the paper we describe the breeding strategies, followed by descriptions of Serengeti world, of our runs, and of our results. We conclude with a brief discussion of directions for further research.

2 Breeding Distributed Agents with Genetic Programming

Heterogeneous agents may be difficult to evolve with GP because existing GP methods are designed primarily to evolve one single algorithm. There are two common ways to tackle this problem. The first is to use *co–evolution* [Koza 1992] to select individuals from a population to form a team at trial–time. Co–evolving specialized cooperation might be made more effective if the population is divided into subpopulation *demes* [Tackett and Carmi 1994]; each subpopulation could provide a specialized member to join each team. However, selecting team members from the population complicates the evolution process with the *credit assignment problem:* when a trial is complete, which individuals get more credit for its success or failure [Haynes *et al* 1995]?

Haynes *et al* note that another way to generate a team is to consider the whole team as one GP individual. This eliminates the credit assignment problem, since there is only one individual per trial. An easy way do this is to take advantage of GP facilities for *Automatically Defined Functions* (ADFs) [Koza 1994] or *Automatically Defined Macros* (ADMs) [Spector 1996]. For teamwork purposes, ADFs and ADMs can be thought of as subindividuals within a main individual, each of which may serve a distinct specialized purpose. By modifying existing ADF systems, it's possible to treat each ADF as a separate agent; GP systems like *lil-gp* [Zongker and Punch 1995] allow the evaluation of each ADF in an individual as if the ADF was an individual entity. We implemented our heterogeneous breeding strategies this way,

Table 2: Deictic Sensing Lion Operators

Name	# Args	Description
`nearest`	0	The vector from the lion nearest the gazelle, to the gazelle.
`lion`	0	A vector from the lion to its nearest neighbor lion.
`rlion`	0	A vector from the lion to the first neighbor lion encountered in a clockwise sweep. The sweep begins in the direction the lion moved last. All the minimal vectors to each lion are gathered, and the first one found in the sweep is returned.
`llion`	0	A vector from the lion to the first neighbor lion encountered in a counterclockwise sweep. The sweep begins in the direction the lion moved last. All the minimal vectors to each lion are gathered, and the first one found in the sweep is returned.

Table 3: Name–Based Sensing Lion Operators

Name	# Args	Description
`lion-1`	0	A vector from the lion to lion #1.
`lion-2`	0	A vector from the lion to lion #2.
`lion-3`	0	A vector from the lion to lion #3.
`lion-4`	0	A vector from the lion to lion #4.

since it is functionally very similar to the clear method of producing homogeneous individuals. In both cases we grow standard GP individuals, and only one "individual" is tested at a given time. For homogenous teams, individuals are tested by cloning them to form teams, and the resulting teams are tested in the environment. Heterogeneous teams, however, are formed from the individual's collection of ADFs.

We allowed for two different kinds of breeding strategies for heterogeneous teams. *Free Breeding* allows any member of a team to freely breed with any other member of another team. *Restricted Breeding* allows team member 1 to breed only with other team member 1's, and team member 2 to breed only with other team member 2's, etc. This restriction further promotes specialization since it breaks team members up into separate breeding pools.

3 The Serengeti World

To test the effectiveness of homogeneous and heterogeneous team algorithms under a variety of sensing capabilities, we chose a predator/prey domain nicknamed the Serengeti. The Serengeti world is a toroidal, continuous 2-dimensional landscape, 15 units on a side. In this world roam a gazelle and a group of lions, initially placed in random positions. The gazelle moves 3 units in one bound, but each lion may only move one unit at a time. The gazelle and lions take turns moving until the maximum number of turns is used up, or a lion has moved to within 1 unit of the gazelle (the gazelle is "killed" and the simulation stops). The programs for the animals return 2-dimensional vectors which determine the direction they move. Animals may move in *any* direction; they are not limited to, say, the cardinal directions. The goal is to generate a lion "pack" that regularly gets as close as possible to the gazelle.

Predator/prey pursuit domains like the Serengeti have been used often to test both evolved and non–evolved agent coordination with genetic programming or distributed AI [Korf 1992, Haynes *et al* 1995, Benda *et al* 1986]. Commonly, predator/prey domains exist in a simple grid world with limited movement abilities (up, down, right, and left), and predators often move faster than the prey (though sometimes the object is not just to "catch" the prey but to surround it with all four predators). In contrast, the Serengeti allows for continuous positions and a much more difficult–to–catch prey. We feel the Serengeti is a useful domain to test heterogeneous versus homogeneous team strategies for two important reasons. First, it presents a problem that is *impossible* for a single agent (one lion) to solve. Since the gazelle moves faster than a lion, the only way to effectively bring down the gazelle is for a group of lions to hunt together. Second, the Serengeti does not clearly lend itself to heterogeneous teams of specialized members: the Serengeti is symmetric and has no obstacles or hill slopes, etc. that give clear niche opportunities for team members. On the other hand, Serengeti world does not clearly promote homogeneous teams either: group members start in random positions, and obvious homogeneous algorithms like "head directly at the gazelle" or "circle the gazelle" turn out to be suboptimal solutions [Korf 1992]. This is especially true for the Serengeti because, unlike earlier domains, in this domain the prey moves much faster than the predators.

Table 4: Trial Results

	Restricted Breeding		**Free Breeding**		**Clones**	
Sensing	Average	Best	Average	Best	Average	Best
Deictic	1.65	0.13	2.03	0.23	1.52	0.20
Name–Based	1.33	0.03	1.79	0.07	1.93	0.22
None	2.20	0.49	2.23	0.50	2.18	0.45

Table 5: Control Results

	One Lion		**One Random**		**Four Random**	
	Average	Best	Average	Best	Average	Best
Controls	7.39	5.43	7.87	6.74	4.41	2.57

3.1 The Gazelle

The difficulty of the domain is directly affected by the intelligence of the gazelle. We chose an avoidance algorithm that runs quickly, but still makes it surprisingly difficult for the lions to corner the gazelle.

The gazelle's heuristic is as follows: Let $\|max\|=\sqrt{(w/2)^2+(h/2)^2}$, where w is the width of the world (15 units), and h is the height of the world (also 15 units). This is the largest possible distance from one object to another in the Serengeti. Then the gazelle moves 3 units in the direction of the vector given by

$$-\sum_{v\in V}\frac{v}{\|v\|}(\|max\|-\|v\|)$$

where V is the set of smallest vectors from the gazelle to the lions. In other words, the gazelle modifies the magnitude of vectors to lions so that closer lions are more significant than far-away lions, and then moves away from the sum of the vectors.

3.2 Lions

For genetic programming breeding purposes, each lion group was treated as a single individual, since the group stays together and is graded together based on its combined success. We tested groups with either one lion or four lions (labeled lion numbers 1, 2, 3, or 4); within a group, a lion was a single GP function tree using 2-element vectors as its data type. All lions used at *least* the basic function set shown in Table 1.

The lions were tested with either deictic sensing ability, name–based sensing, or no sensing ability. Lion groups with deictic sensing were given the four additional operators `nearest, lion, rlion, llion` (Table 2). Lion groups with name–based sensing were given the four additional operators `lion-1, lion-2, lion-3, lion-4` (Table 3). Lion groups with no sensing were given no additional operators. We implemented the three breeding strategies as follows: *Clones* really consisted of a single function tree, alternately given four different sets of environment variables to simulate four identical lions. *Restricted* and *Free Breeding* groups had four separate function trees, one per lion. In addition to combinations of breeding strategy and sensing operators, we performed control runs for one lion using the basic operators, one lion moving completely randomly, and four lions moving completely randomly.

4 Test Runs

We performed 1200 random runs under the *lil-gp* GP system with the Serengeti problem: 100 runs for each of the nine combinations of sensing and breeding strategies, plus 100 runs for each of the three control configurations. We assessed the fitness of groups based on the average of 5 simulation trials. At the beginning of each simulation, the lions and gazelle started in random positions. During each turn, the gazelle moved first, then the lions. This was repeated 15 times, or until the gazelle was "killed" as explained before. On each turn, lions moved one unit in the direction of the final vector returned. If a lion returned a 0-length vector, it moved randomly 1 unit. At the end of each simulation, the lion group's fitness was based on the distance from the gazelle to the lion nearest the gazelle. If this lion finished within 1 unit of the gazelle, the lion group grade was 0. Otherwise, the group's grade was the distance of the nearest lion to the gazelle, minus 1. Lower scores were better.

Each run lasted for 51 generations, with a population of 500, a maximum tree size of 70 and a maximum tree depth of 17. We used crossover (90% of the time) and reproduction (10% of the time) to generate new individuals.

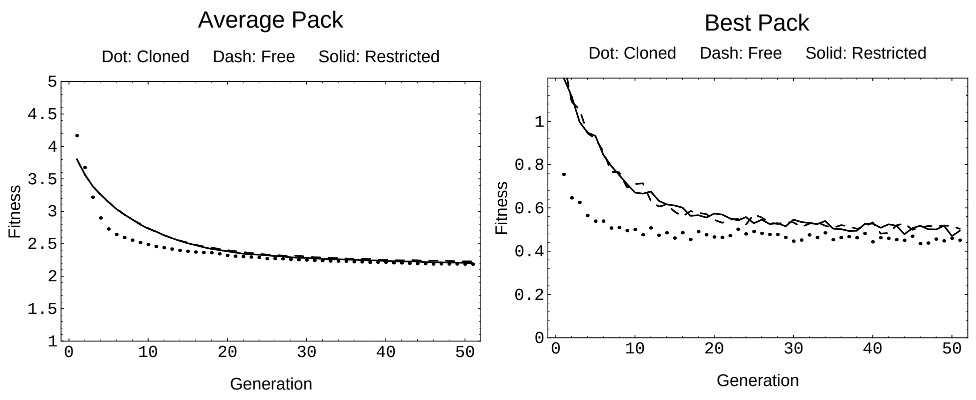

Figure 1: No–Sensing Results

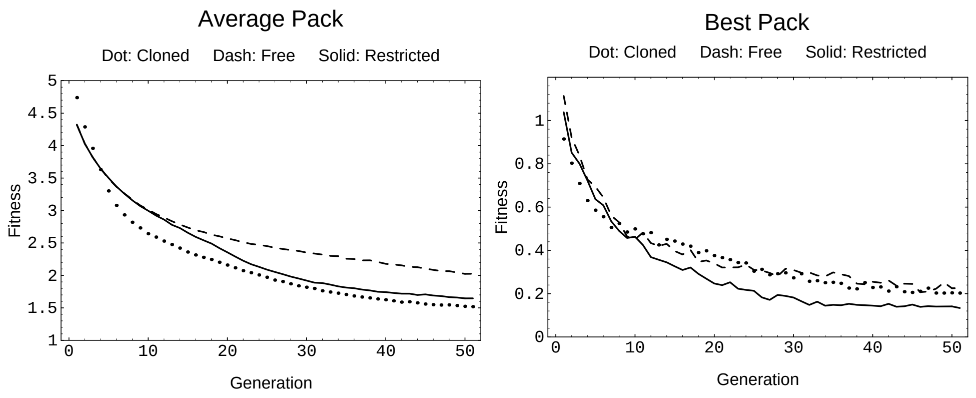

Figure 2: Deictic Sensing Results

5 Results

The results are summarized in Tables 4 and 5. *Average* gives the fitness of the average group in the population at the 51st generation, and *Best* gives the fitness of the best group.

When there was no sensing at all, the best algorithm was typically "go to the gazelle" (in other words, `gazelle` or minor variants). This simple algorithm seemed to predominate in the best lion groups. Given that the better algorithms were often very simple, the groups of cloned lions (which only had to evolve one tree) evolved faster and ultimately slightly better than the heterogeneous groups. Overall, the no–sensing results were not as good as those obtained with sensing. No–sensing results are summarized in Figure 1.

This pattern held more or less for deictic sensing; the clone groups evolved faster and after 51 generations the population as a whole came out ahead, though restricted breeding usually generated the best lion groups. This might be because restricted breeding really was producing the best individuals even if its average might be worse than the average groups under other breeding strategies. Or perhaps this was because restricted–breeding's diversity helped create unusual groups that happened to be well–suited for the initial lion and gazelle positions dealt them that generation. Deictic sensing results are summarized in Figure 2.

In the case of name–based sensing, however, the heterogeneous groups clearly had the upper hand. For both the best of the population and for the population as a whole, the heterogeneous groups produced significantly better results than the clone groups, which fared only slightly better than when they had no sensing at all. Name–based sensing results are summarized in Figure 3.

In all cases, restricted breeding produced as good or better results than free breeding. Furthermore, name–based sensing produced better results than deictic sensing, which produced much better results than no sensing at all. This suggests that even in this highly symmetric domain, creating and enabling specialized team members still proved more profitable than using a uniform multi–agent method.

As expected, the controls did miserably. One lone evolved lion fared slightly better than one randomly-moving lion. Four randomly-moving lions predictably did much better still. But all three did much worse than any evolved four–lion scheme. It's clear that the evolved four–lion schemes are succeeding on their own merit and not just because there are four lions filling space rather than one. Control results are summarized in Figure 4.

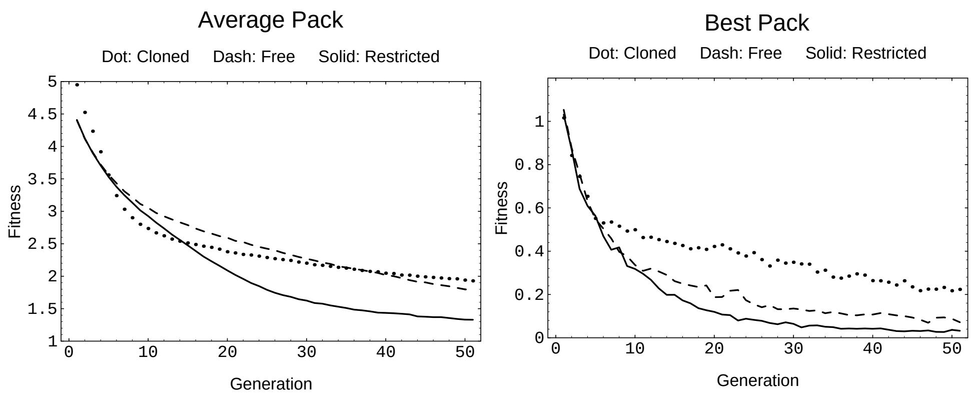

Figure 3: Name–Based Sensing Results

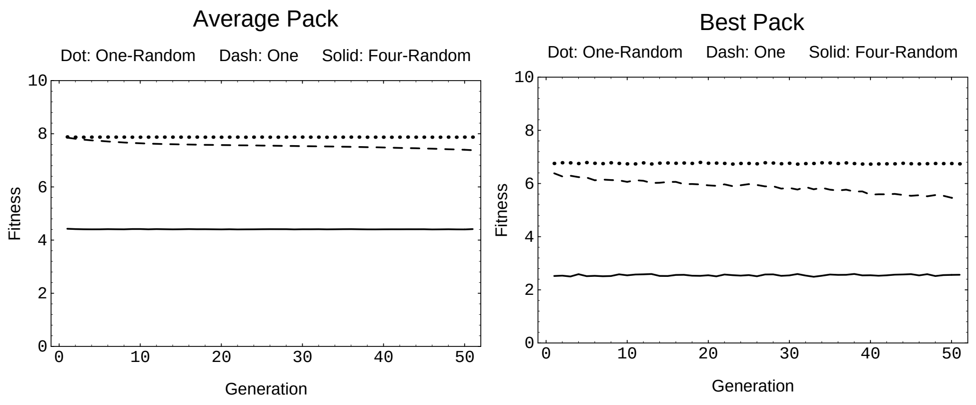

Figure 4: Control Results

6 Conclusions and Future Work

It appears that when given direct information about each other's whereabouts, distinct agents fare best in the Serengeti domain. Uniform agents only do well when coordinating with deictic sensing. And no agents did particularly well when denied the ability to sense teammates. It seems that, although it might require slightly more time for evolution, using a heterogeneous multi–agent strategy may be worthwhile even in a domain where it's unclear that it can help.

Further, restricted breeding consistently evolved faster than free breeding. While there is no reason that free–bred agent teams couldn't eventually develop as well as restricted teams, restricted breeding evolved so much more rapidly that it demonstrated a clear advantage. It seems better to try restricted breeding when developing heterogeneous agents.

In this experiment, a more direct form of sensing proved to be the best method for solving a surprisingly difficult problem. This suggests that even more sophisticated interactive strategies might do better still. One natural extension of this work would be to allow agents to *communicate* with one another through more sophisticated methods than simple directional cues. For example, agents might evolve to communicate instructions to specialized follower–agents through simple memory storage to which both have access, such as the kind used in MAPMAKER [Andre 1995]. Another open issue is whether advantages conferred from developing multiple independent agents are really worth the computational expense, even if they do perform better than uniform agents.

More domains must be studied to determine if the results for the Serengeti generalize to other interesting multi–agent domains. In addition, the conditions that were studied in this paper represented only a few of the many possible breeding strategies and coordination mechanisms. Nature provides other illustrative examples; for example, bee colonies consist of many hundreds of identical individuals from perhaps a half–dozen specialized classes (queens, drones, workers, etc.). This sort of hybrid breeding strategy could also be tested in Serengeti world, and may have more widespread applicability.

Acknowledgements

This research was supported in part by grants from ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), and ARPA contract DAST-95-C0037.

We are particularly grateful to Glen Henshaw and the General Motors Truck Division for his assistance in the development and preparation of this paper.

Thanks also to Mark Feinstein for helping to correct some of our more glaring errors concerning animal behavior.

Bibliography

Andre, D. 1995. The Automatic Programming of Agents that Learn Mental Models and Create Simple Plans of Action. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. Chris S. Mellish, editor. 741–747. San Mateo, CA: Morgan Kaufmann.

Benda, M., B. Jagannathan, and R. Dodhiawala. 1986. On Optimal Cooperation of Knowledge Sources. Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA.

Haynes, T., S. Sen, D. Schoenefeld and R. Wainwright. 1995. Evolving a Team. In *Working Notes of the AAAI-95 Fall Symposium on Genetic Programming*. Eric V. Siegel and John R. Koza, editors. 23–30. AAAI Press.

Korf, R. 1992. A Simple Solution to Pursuit Games. In *Working Papers of the 11th International Workshop on Distributed Artificial Intelligence*. 183–194.

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

Koza, J.R. 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA: The MIT Press.

Lohn, J. and J. Reggia. 1995. Discovery of Self-Replicating Structures using a Genetic Algorithm. In *1995 IEEE International Conference on Evolutionary Computing*. 678–683. Perth: IEEE.

Reynolds, C.W. 1993. An Evolved, Vision–Based Behavioral Model of Coordinated Group Motion. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Jean-Arcady Meyer, *et al*, editors. 384–392. Cambridge, MA: The MIT Press.

Spector, L. 1996. Simultaneous Evolution of Programs and their Control Structures. In *Advances in Genetic Programming 2*, edited by P. Angeline and K. Kinnear. Cambridge, MA: MIT Press, in press.

Tacket, W.A., and A. Carmi. 1994. The Donut Problem: Scalability, Generalization and Breeding Policies in Genetic Programming. In *Advances in Genetic Programming*, Kenneth E. Kinnear, Jr., editor. 143–176. Cambridge, MA: The MIT Press.

Zongker, D., and B. Punch. 1995. *lil-gp 1.0 User's Manual.* Available through the World-Wide Web at http://isl.cps.msu.edu/GA/software/lil-gp, or via anonymous FTP at isl.cps.msu.edu in the /pub/GA/lilgp directory.