Evolution of Simple Intelligence Distribution in Artificial Organisms

Philip Dhingra
PO BOX 12473
Stanford, CA 94309
(858) 485-1952
philipd@stanford.edu

Abstract: This paper uses Genetic Programming to evolve groups of ants to push a box from the center of a room to a wall. Group sizes and ant capabilities are varied to observe the speed, effectiveness, and nature of the intelligence that evolves for each ant. As expected, larger groups compensate for lesser intelligent ants by having more of them to solve the task. The ant-box-pushing problem then becomes a coverage problem whereby solutions are found by adequately covering the space in which the task is to be completed.

I. Introduction and Background

A. Introduction

A group of ants are placed in a room with a box of food that they have to push to a wall. The objective of this simulation is to evolve the individual ants so that when placed in a group they are able to effectively coordinate the pushing of this box. The more ants that help push the box, the faster the food is collected. The faster a group collects the food, the more favored that group of ants will be in breeding for subsequent generations. Also, since the box is rectangular, the ants must also coordinate evenly pushing the box so that not too much time is wasted in pushing a corner of the box thereby rotating the box in circles.

B. Simulation Details

The *n* ants in a group are represented as points in a 100cm x 100cm square room. Each ant can either rotate 45 degrees or move forward 2cm per time step. There are 100*n* time steps to complete the task of pushing the box. The number of time steps is based on providing sufficient time for an ant to come in contact with the box (< ~35 time steps from any point to the box) and sufficient time to push it toward a wall (< ~ 25 time steps from the box to the wall).

The box is 7cm x 7cm and is placed in the center of the room. The ants are randomly placed around the room with random orientations in a trial. A trial is repeated 50 times in order to ascertain the average effectiveness of the program used for that group of ants.

There are three levels of capabilities provided for the ants:

Level A – Here the ants have no cognitive or communicative capabilities. They are only allowed to rotate to the left or right or move forward.

Level B – At this level, ants get the additional ability to sense if there is a box 20cm ahead.

Level C – This level adds the most basic communication among the ants, simply allowing them to detect whether there is another ant 20cm ahead. It is arguable whether this can be considered communication. We choose to consider it as such since as an ant suddenly observes another ant moving, he may want to follow to aid in the pursuit of the box. So ants' movements and directions, on a basic level, implicitly communicate to other ants the potential location of the box.

Initially, the groups of ants will not have any intelligent method of using their capabilities. As successive generations of ants are evaluated for their fitness, the better species in each generation will tend to breed more often and produce offspring for subsequent generations. Eventually, intelligence will emerge in later generations of ants.

The fitness of each ant is determined by how quickly and how often the box is pushed to the wall. If the box is not pushed all the way to the wall but is dislodged from its center within the time allotted, then the proximity of the box to the wall is used for fitness.

C. Technology Used

The simulation in this paper uses Genetic Programming (GP) based on the work of John Koza at Stanford University. Each ant is represented by a simple LISP-based program. The LISP-code represents a tree with each internal node representing a cognitive or communicative ability of the ant and each terminal representing an action, like move forward, rotate, etc.

During breeding, sub-trees of programs are swapped to simulate the cross-over effect that occurs in DNA. This provides for novel species that are, in many cases, better suited for the task at hand. In addition mutations can occur during breeding. These are used sparingly, but they allow for program trees to change spontaneously so as to escape from situations in which the simulation leads to locally maximum performance.

D. Motivation

GP offers a method for simulating Darwinian natural selection. Every application of a GP-based problem gives at least some insight into the way organisms in nature have adapted to their various ecosystems and niches. These adaptations can then be extracted for use in solving problems in other fields, such as economics and electrical engineering.

In autonomous organisms, or nodes, common adaptations that emerge include communication, cognition, and group-wise coordination. Coordination is seen as a natural result of communication and cognition; once organisms can sense their environment and peers, they can leverage the power of groups by dividing a task amongst themselves.

By abstracting the problem of coordination into a simple ant-box-pushing problem, we can gain some general insights into the influence certain factors may have on the utility of group behavior. Such insights are useful for systems theory and other evolutionary systems. Many of these insights also confirm our common sense notions about group behavior. For example, if there is a simple task to be completed by a large group of people, there is a tendency for simplicity and laziness to propagate in individuals. These and other generalizations about group behavior and intelligence distribution are useful for designing group-based solutions.

II. Simulation Setup

A. Simulation Intent

Other GP simulations focus on developing an optimum or novel solution to problems or situations. The box-pushing-simulation is different. Since a solution is fairly simple, our goal is to see how close and fast the simulation comes to hypothetical target solutions.

<u>Level A General Target Solution:</u>

```
(progn2  (progn2 fwd right)
         (progn2 fwd left )
```

This is also known as "zig-zag." An ant with this program moves forward, turns left, moves forward again, turns right, and then moves forward again. A group of zig-zaggers will crawl around the room and have a high chance of hitting and pushing the box to the wall.

Level B General Target Solution:
(if-box-ahead fwd zig-zag)

This is a "zig-zag-finder," a slight modification of "zig-zag." Here the ants will crawl around the room until one sees a box. At that point the ant will then pursue the box, pushing it to the wall

Level C General Target Solution:
(if-box-ahead (if-ant-ahead fwd fwd)
(if-ant-ahead fwd zig-zag))

This is called a "zig-zag-finder-chaser." As the naming suggests, ants with this program crawl around, and if they see an ant, chase them. This is to have ants detect other ants that might be moving toward boxes and then follow them to aid in pushing the box to the wall.

If we limit LISP-trees of ants in a Level A simulation to have a minimum depth of 2, the probability of generating the Level A General Target Solution is 1 in 2,500 (.5 chance on nodes at depth 1 and 2, and 1/5 chance for each node in depth 3). Plus, there are many variations of the zig-zag program, so by simple enumeration, this solution should be reached in much less than 2,500 iterations. Level B and Level C have modifications on the zig-zag problem and only add one or two levels of depth. Since these solutions can be created easily, our goal then is not to test whether we achieve these target solutions through GP, but how quickly and with what proximity we do so.

B. Major Preparatory Steps

In Koza's *Genetic Programming* five major preparatory steps are essential to every GP-based project (Koza 1992):

1. Terminals

These are the terminal nodes in any program tree representing each ant:
- Left (45 degrees)
- Right (45 degrees)
- Forward (2cm)

2. Functions

These are the internal nodes in any program tree representing each ant:
- Level A – None
- Level B – If-Box-Ahead
- Level C – If-Box-Ahead, If-Ant-Ahead

3. Fitness Measure

There are 50 trials for each ant. During each trial, the ants are randomly placed in the room with random orientations. When the trial ends (by either time-out or by a box hitting the wall), a score is assigned to the trial. The score is based 50% on the time it took and 50% based on how far the box was from the wall. A group of ants, in an impossible scenario, that pushes the box to the wall and takes 0 time steps will receive a score of 0. If the box never moves from the center and all of the time steps are used, then that trial will receive a score of 1. The average of 50 scores is taken and that becomes the fitness of a program. The fitness also tracks the number of hits for 50 trials based on the number of times the box made it to the wall.

The number of trials, 50, was discerned through trial-and-error. I tested various trial sizes to see which ones were sufficient for identifying the average performance of that program.

4. Parameters

Population Size = 100
Since we require that the initial population have a minimum depth of 2, a population of 100 ensures that each population will represent all possible functions and terminals. Although each combination will not be represented in the initial random population, the primary concern in population size is to ensure that the original ancestors of generation 0 have enough diversity to ensure that over a number of generations their children can adequately represent the best individuals for that level.

Max Generations = 20
As mentioned in Section A, random enumeration could eventually get the target solutions for each level. Hence, we keep the number of generations relatively small. In GP, once individuals with a high fitness emerge, they tend to crowd the population with duplicates of themselves, creating a plateau or equilibrium in the evolution of that run. So, by trial and error, 20 generations was found to be sufficient for the runs to reach a plateau.

Max Depth = 6
This is based on the maximum depth of the General Target Solutions mentioned in Section A.

Breeding Parameters
Individuals are selected with a frequency proportionate to their fitness and have a 90% chance of undergoing a cross-over genetic operation, a 10% chance of undergoing reproduction, and a 1% chance of mutation. The mutation is used to help in breaking out of local maxima. The 10% reproduction is to ensure that at least some of the top individuals are duplicated.

Terminating a Run
Runs are terminated when an individual group of ants, in all 50 trials, pushes the box to the wall.

III. Results

A. Snapshots

In none of the runs did an exact replica of the general target solutions occur. However, some other patterns emerged. These appear on the following page.

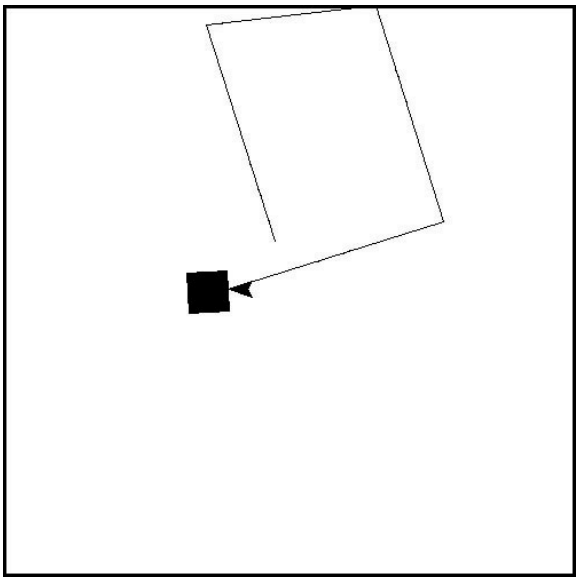

Figure 1, Spiraler: This was the best-of-run program for a Level A experiment with 1 ant. Here the ant spirals around in order to hit the box. Achieving a sea-shell spiral is not possible as the program for the individual cannot keep track of an increasing exponent after each run. However some basic circling so that the ant has more chances of hitting the box at least scores a few hits

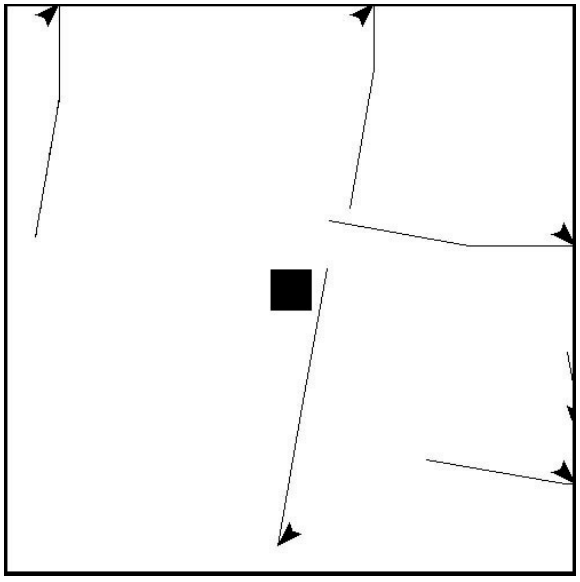

Figure 2, Zig-Zagger: This best-of-run for Level A, 6 ants, somewhat mimics the zig-zag approach. You can see a bend in the path of the top three ants. These bended lines are better than straight lines as they increase the chances that an ant will hit and push a box.

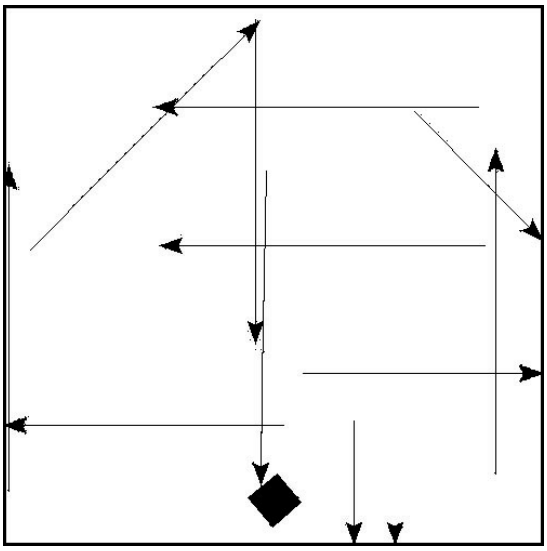

Figure 3, Lungers / Marchers: This snapshot shows the best-of-run for Level B, 12 ants. Here the ants simply lunge forward as far as they can, hopefully hitting the box and therefore pushing it to the wall. Lungers will try to move forward as much as they can before letting other ants move. Marchers will only move forward once or twice at time.

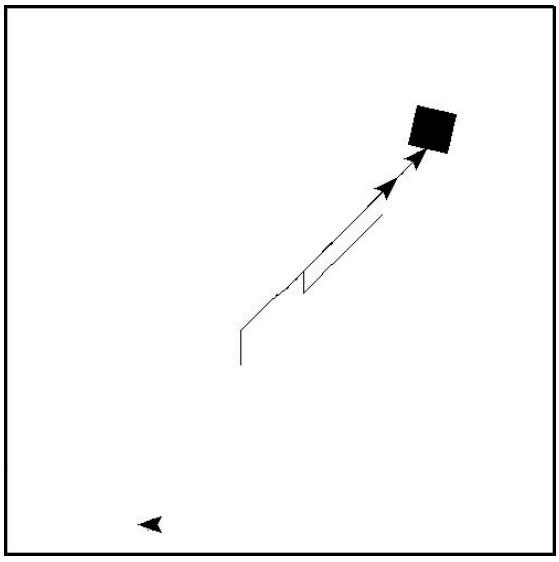

Figure 4, Spinners: This best-of-run from Level B, 4 ants, shows the ants acting on cognition. Here the ants spin in place until encountering a box, at which point they will move toward the box.

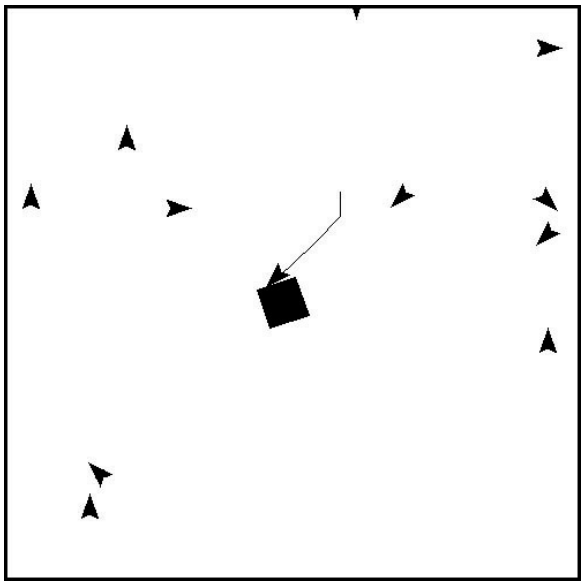

Figure 5a, Spinner-Lunger: This snapshot from Level B, 12 ants, shows in more detail how an ant will push a box, possibly rotating it out of reach.

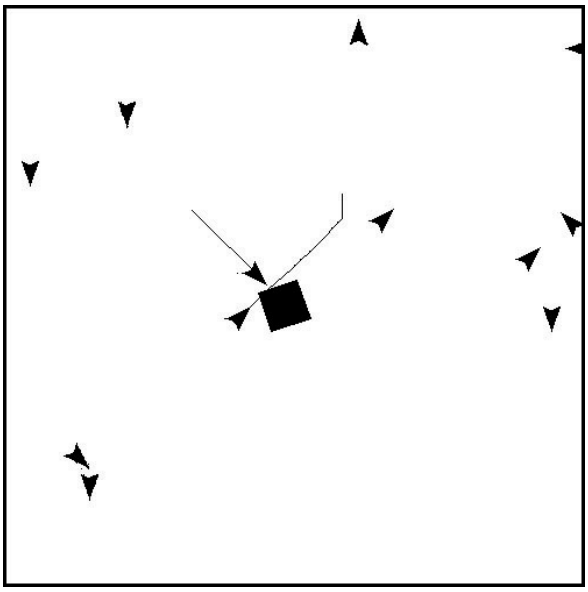

Figure 5b: Pushing the box, though, may put it in view of another ant which could then lunge forward and complete the task.

B. Table of Results

Level A

No. Ants	Final Type	Plateau Generation	Hits
1	spiraler	7	1
2	spiraler	17	7
4	spiraler	17	8
6	marcher	15	15
8	marcher	15	18
12	lunger	17	23
22	marcher	9	29
32	marcher	20	31

Level B

No. Ants	Final Type	Plateau Generation	Hits
1	spinner	1	17
2	spinner	1	21
4	spinner	1	30
6	spinner	1	38
8	spinner	1	43
12	spinner-lunger	9	50
22	spinner-lunger	3	50
32	spinner-lunger	3	50

Level C

No. Ants	Final Type	Plateau Generation	Hits
2	spinner	11	20
4	spinner	8	27
6	spinner	11	40
8	spinner	14	44
12	spinner	5	49
22	spinner	15	50
32	spinner	8	50

C. Discussion of Individual Runs

A1 – For generations 0-6, this run held zig-zaggers. At generation 7, A1 reached a plateau and stabilized on a spiraler. It is tempting to conclude that spiraling is better than going forward in zig-zags. However, since the hits are so low with just one ant, it is difficult to say that any possible solution for a randomly moving single ant is useful at completing this task.

A2 – This went the same way of A1 except that its plateau point is much later and the number of hits is 7. With two ants, the spirals can cover more quadrants of the room. Plus, variations in the sizes of spirals also determine how much of a quadrant is covered. The best-of-run lunges forward 24cm, rotates left, then moves forward 8cm, and then rotates left again. This creates an approximate circle that occupies much of a quadrant of the room.

A6 – Until generation 6, the ants zig-zagged. Eventually, though, marching forward was found as the best strategy for pushing the box. The increase in the quantity of the ants increased the effectiveness of simple marching; with enough ants moving simultaneously in various directions across the screen, the box is likely to come in contact with an ant.

A12 – Lunging won out over simple marching for 12 ants. This makes sense given the quantity of ants. Consider a marching scenario with two ants near the box, one facing the top, the other the left. In marching, the top ant could nudge the box down. The left ant would then, instead of facing the center of the left side would be facing a corner of the box. That ant then marches forward and rotates the box. Eventually, the box could rotate out of the path of both ants. With many ants, there are many chances for this kind of interference.

A22,32 – The trend of lunging was short-lived, and with many ants, marching was good enough to get a considerable number of hits. Even if interference scenarios occurr, as described above, there are enough ants to continue pushing the box to the wall.

B1-8 – Spinning is so simple and effective that it could be developed in generation 1 and persist till generation 20. Spinning is close to the spinner-zig-zagger. However, the Level A experiments showed that zig-zagging doesn't win out over marching or spiraling.

B12 – Up until generation 9, there was competition between marching and spinning.

B22, 32 – Like B12, there was also competition between marching and spinning. Marching was somewhat effective in A22 and A32, so this is no surprise. However, with so many ants, the runs in Level B stabilized quickly on spinning. This is because 50 hits were reached quickly with so many ants spinning around.

C22 – The If-Ant-Ahead function was used in this interesting case scenario. In this program, if an ant detected both a box and an ant ahead, it would continue spinning around, rather than pursuing the ant and box. This could indicate a few conclusions. One is that this evolution was used to prevent interference with other ants that are already pushing the box. Another is that, unnecessary movement of an ant toward a box saps the number of time steps available for the group. And a third conclusion is that this unique evolution indicates an evolved laziness in the ants as they didn't need to move toward the box in all occasions to achieve the 50 hits necessary to terminate the run.

C-others – Unfortunately, no setting of the variables could consistently arrive at anything resembling a chaser. The If-Ant-Ahead function didn't provide the proper advantages necessary for ants to collect the food. This could be the result of the fact that if ant X spots ant Y, then X doesn't need to follow Y; Y already has a range of space covered by its vision, and so if there was a box near Y, Y could take care of it on its own. Two ants working together didn't seem to enhance the pair's ability to collect the food. Rather, it's the distribution of the ants all around the room that mattered.

E. Conclusion

This simulation corroborates some common sense notions about the way groups can intelligently use tools to solve a task. In our simulations, single ants were highly ineffective in collecting the food due to each individual ant's lack of intelligence. Increasing the group size, however, compensated for the lesser intelligent ants. Likewise, in general group-based systems, consider members of a group that are autonomous. If these members were left on their own, they may not be very effective at completing their task. In response, one could increase effectiveness by simply increasing the number of members attempting the task.

A generalization of these results works only for certain types of tasks. The box-pushing task involves a space—the room—and a target region—the box. The task could only be completed if a node in the space probed that region. This turns the box-pushing problem into a coverage problem. The best way to solve a coverage problem is simply to cover more of the space by increasing the number of probes. However, the presence of a probe in the space changes the nature of that space, and therefore in some cases, as in C22, too many probes may hinder progress in actually hitting the target.

This simulation also demonstrates, on a basic level, the capabilities of GP to simulate the evolution of group-based systems. GP provides the opportunity to develop novel directives for autonomous nodes in multi-node systems. This is useful because engineering the behavior of individuals so as to capture a target emergent behavior of a group is no easy task.

IV. Notes

A. Problems Encountered and Failures

- There was a problem encountered in not being able to hit target solutions. There was also a general problem of uninteresting programs such as spinners (see table). This could be remedied with more runs or by tweaking the variables, such as the visibility of the other ants and the breeding parameters.

- Duplicating nature within a program was difficult. I ultimately resorted to a modified version of nature. For example, each of the ants is represented as a point which means they can practically occupy the same space. Plus, the box doesn't move exactly as it would if there was an ant pushing in real life; in the simulation an ant has to move 2cm in every step, but in real life, an ant attempting to push a box would slow down and cover less distance per time step. This could have encouraged the ants to work together in groups.

- Providing interesting conclusions and abstractions to systems theory is difficult and vague with the ant-box-pushing task. This problem may have too many specific parameters that don't generalize to abstract systems.. For example the box rotates, the ants can only spin a certain number of degrees, the space is two-dimensional, etc.

B. Future Work

- The simulation could be improved to more accurately reflect the way groups of ants work in nature. For example, in the real world, the ants move around simultaneously, not one after the other. Other nuances could also be simulated, such as each ant occupying a rectangular space for their body.

- Just as the simulation could be improved by more closely mimicking nature, it can be improved by also being more abstract. The nodes could be more generalized. Instead of ants, have dots that can rotate in any direction possible and maybe expand the plane of movement into 3D space. The box could also be a circular, and so could the room.. This could help to make more concrete suggestions on the nature of group-based evolutionary systems.

- Many more experiments and runs could be made to get more conclusive data. The number of generations could be increased and the number of levels could also be increased. These enhancements come at a cost to processor time which must be considered.

- More function sets could also enhance the simulation. The ants could communicate with each other, have memories, and even maybe drop pheromones.

- Other strategic modifications to the problem could be made. For example, one could add more boxes to the room, change the shape of the room, or locate the box closer to certain walls.

C. Duplicating Results

This simulation was developed using the lil-gp 1.1 Genetic Programming System (http://garage.cps.msu.edu/software/lil-gp/lilgp-index.html)

D. References

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

E. Acknowledgements

Some of the ideas for this project were inspired by an experiment on a single ant pushing a box found in:

Werner, Gregory M. and Michael G. Dyer. 1992. Evolution of Communication in Artificial Organisms. In Langton, Christopher G . . . [et al.]. *Proceedings of the Second Artificial Life Workshop*. Redwood City, CA: Addison-Wesley. Pages 659-687.

This project was created for a class on Genetic Programming at Stanford University given in Spring of 2002. The class was taught by John R. Koza, who also helped give me general direction through this project.