Evolutionary Dynamics

Shane Celis
Applied Science
secelis@ucdavis.edu

June 10, 2008

Abstract

This paper examines genetic algorithms and asks whether coevolutionary algorithms produce qualitatively different results than those employing fitness functions, and can one derive a fitness function from a coevolutionary algorithm?

1 Introduction

Genetic Algorithms (GAs) are a stochastic search method inspired by biological evolution. They are useful for searching large multidimensional spaces usually for local minima or maxima to achieve some objective. The basic recipe for a GA looks like this:

1. Seed a population (all random for generation 0)
2. Evaluate fitness of individuals
3. Select individuals proportionate to their fitness
    (a) Copy fittest individuals to the next generation
    (b) Mutate fittest individuals to seed the next generation
4. Restart at step 2 with the new population

Despite the simplicity of this algorithm, it can produce non-trivial results. The fitness function used in step 2 can be arbitrarily complex and comes in two varieties. Explicit fitness functions operate on the gene values directly. Implicit fitness functions do not have "global" knowledge, and instead may use simulation, or some other indirect means to measure fitness [2]. For example, Karl Sims' work [3] demonstrates virtual creatures that were evolved for visually realistic locomotion. Sims' work is what initially provoked my interest in GAs. Sims' work uses an implicit fitness function where the distance the creature moved in the physics simulation from its starting position after a certain time is its fitness; the farther it moves the fitter it is.

Another form of a GA is a coevolutionary algorithm. In the coevolutionary algorithm, there is no fitness function (either explicit or implicit). Instead there is an environment that defines a replication dynamic. There can be "death" and "competition" as well. This mirrors natural evolution much closer, since the only objective measure of fitness is differential replication of genes.

I would like to consider the case where one does not have a fitness function. When one has a fitness function, one can take any two genes and determine which is better or more fit. However, in a coevolutionary algorithm, one cannot pluck two genes out of the population and determine which is more fit. The questions I would like to entertain are, does a coevolutionary algorithm produce qualitatively different results than a GA with a fitness function? Some of my thinking about whether something is qualitatively different or new was provoked by [1]. It may be that the coevolutionary algorithm is merely an obscured fitness function. If so, one could approximate the results of a coevolutionary algorithm by finding that obscured fitness function. And following up on that, supposing that coevolutionary algorithms are qualitatively different, therefore are not merely obscured fitness functions, over certain time intervals can one still approximate them with a fitness function?

1.1 Motivation

One motivation of this paper is that being able to derive an approximate fitness function could be useful. Consider a scenario where there is a GA, but no clear cut means of evaluating fitness at least not computably (e.g., human selects what they find aesthetically pleasing). GAs are terribly tedious if a human must oversee every individual to rate its fitness. If, however, one could observe what the human selects and infer a fitness function, one could still leverage a GA despite the non-computable fitness function. Another motivation is that assuming one has a biological model implemented as a coevolutionary GA, one could analyze it to determine what kind of selective pressure was being applied to the population, and perhaps analyze how that changed over time.

1.2 Synopsis

This paper details the simulation used, the results of the simulation run in two different modes, and unfortunately is inconclusive as to whether coevolutionary algorithms are qualitatively different from those employing fitness functions. It does dicuss how one might actually come to an answer and what extensions to the simulation may be worthwhile.

2 Method

A difficult part of this project was trying to think of a replication dynamic. I wanted something that was not so simple that there was only one stable, trivial solution that the GA quickly settled on. I wanted something that was not so complex that it would be too difficult to analyze. I settled on simulating particles on a plane.

The concept is that these particles live on a flat grid with an edge. If they go over the edge, they "die." If they stick around in the simulation long enough, they will replicate themselves with a chance of mutation. The behavior of a particle is partially controlled by its genes. The genes define the acceleration the particle experiences. A second order differential equation was selected so that collisions could be implemented between the particles. The gene is a vector of real numbers (in my case $n = 17$, which also encodes some non-functional elements like color).

2.1 Dynamical System

$$\mathbf{g} \in R^n \tag{1}$$

$$\ddot{x} = g_0 + g_1 x + g_2 x^2 + g_3 y + g_4 y^2 + g_5 v_x + g_6 v_y \tag{2}$$

$$\ddot{y} = h_0 + h_1 x + h_2 x^2 + h_3 y + h_4 y^2 + h_5 v_x + h_6 v_y \tag{3}$$

$$h_i = g_{i+\frac{n}{2}} \tag{4}$$

The mutation operator has two parameters: mutation chance per vector element c_m, and standard deviation of mutation σ_m. The mutation chance per vector element is evaluated against a uniform distribution ($A \sim U[0,1]$). If $A \leq c_m$, then the element is mutated using a Gaussian distribution similarly to what is described below for an offspring's position.

The coefficients for drag (g_5, g_6, h_5, and h_6) can be turned on or off conditionally. In this paper I have turned drag off. The condition for falling off the edge is if $|r_i| \leq 2$ where $\mathbf{r}$ is the position of the particle, so the grid goes from $[-2, 2]$ on each axis.

2.2 Fitness Mode

The simulation has two major modes: the fitness function mode, and the coevolutionary mode. The fitness function mode runs the basic evolutionary algorithm as described in the introduction. The population of genes is a constant parameter N. To evaluate a gene, M particles with the same gene but different initial conditions are run in the simulation for a number of simulation steps. Collisions are turned off so that the particles behave independently of one another. The fitness for the gene is the average of the time the particles lived. The bigger the average time, the more fit the gene is considered to be. (One concern with this method is that the evaluation of the gene's fitness might have large fluctuations since the initial conditions can change, but the fact that we're averaging over M independent trials leads me to think this is fine.) Selection is implemented simply as taking a percentage of the N genes with the highest fitness. The percentage of genes to keep is encoded in parameter $k \in [0,1]$ and $floor(kN)$ genes are kept and mutated to seed the next population.

2.3 Coevolutionary Mode

The coevolutionary mode does not have a fitness function, instead it has a replication dynamic, i.e. long lived particles replicate, and the population fluctuates. It starts with random genes. The particle population is kept above a minimum P_{min} and below a maximum P_{max}. If the population of particles sinks below the minimum, a new particle is added by randomly (uniformly) selecting an existing particle in the population to replicate with a chance of mutation. The particles interact by colliding.

The particles have a reproduction clock t that keeps track of how long they have been alive since they last reproduced. A cumulative exponential distribution is used to determine when the particle replicates (if $1-e^{-\lambda t} < B$ where $B \sim U[0,1]$), the thinking being that the longer the particle has been alive the more likely it is to replicate. After it replicates, its reproduction clock is reset to zero. (Because this conditional replication is evaluated every clock tick, finding the appropriate λ value was a non-trivial task, for me at least. Ultimately, I ended up finding a linear relation between the mean time to replicate and λ empirically.)

When a particle replicates, its genes are copied with a chance of mutation to its offspring. Each gene has generation number g_n and its offspring receives a generation number of $g_n + 1$. The initial population has a g_n of 0. The offspring's position $\mathbf{R}$ is inherited from the parent's position $\mathbf{r}$ using a random Gaussian distribution as shown below. The offspring's velocity is done similarly.

$$X \sim N(r_i, \sigma_r^2) \tag{5}$$

$$R_i = X \tag{6}$$

Collisions are turned on for the coevolutionary mode. There is a coefficient of restitution parameter $e_r \in [0,1]$ where $e_r = 0$ is a perfectly inelastic collision, and $e_r = 1$ is a perfectly elastic collision. By default e_r is kept at 0.5, and all the cases examined in this paper keep it there.

The aim is to run these two different modes and compare the results. The fitness function and replication dynamic were selected in an attempt to make their results somewhat comparable and try to avoid comparing apples to oranges.

3 Results

Here are the results from the two modes of the simulation.

3.1 Fitness Mode Results

I used the following parameters for the fitness mode runs: $N = 10$, $M = 10$, $k = 0.2$, $\sigma_m = 1.0$, radius $= 0.05$, $\Delta t = 0.05$, and $c_m = 0.1$. Figure 1 is a graph of the mean fitness for the simulation run in fitness mode.

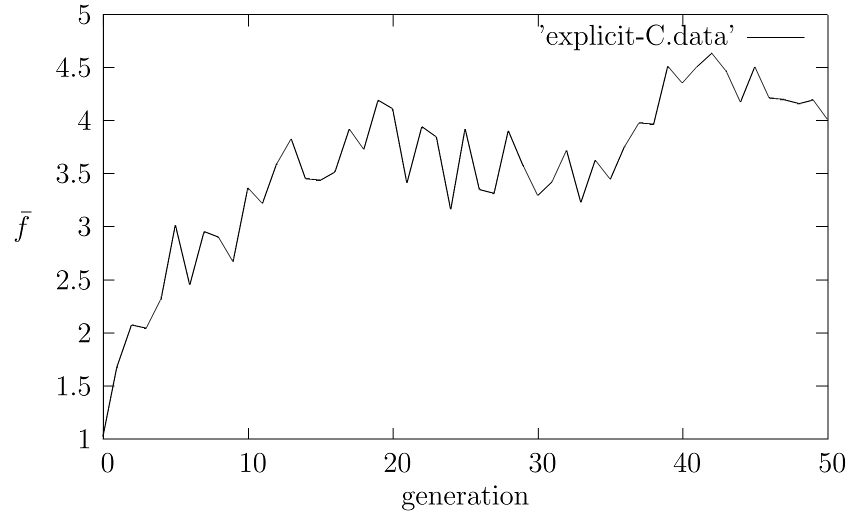

Figure 1: Mean fitness for a run of the simulation in fitness mode

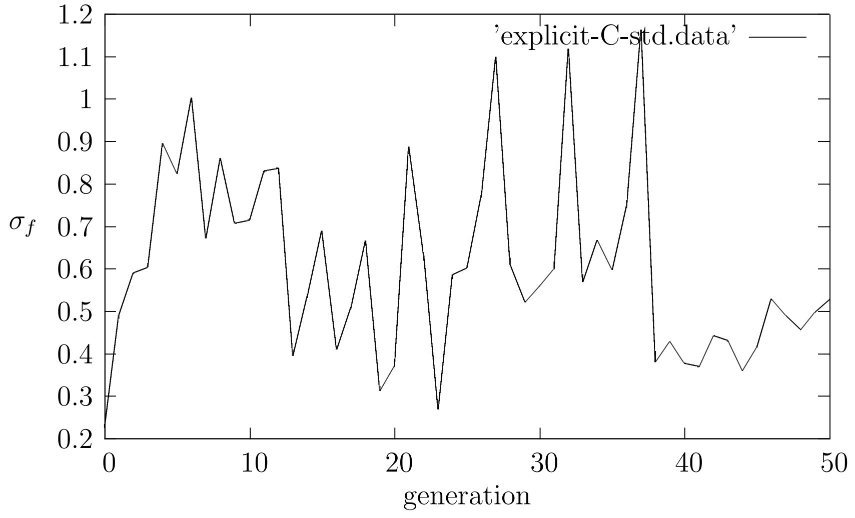

Figure 2: Standard deviation of the fitness

To try and cut down on the noise, I ran the simulation with the same parameters for 50 generations and then averaged the results from three different runs.

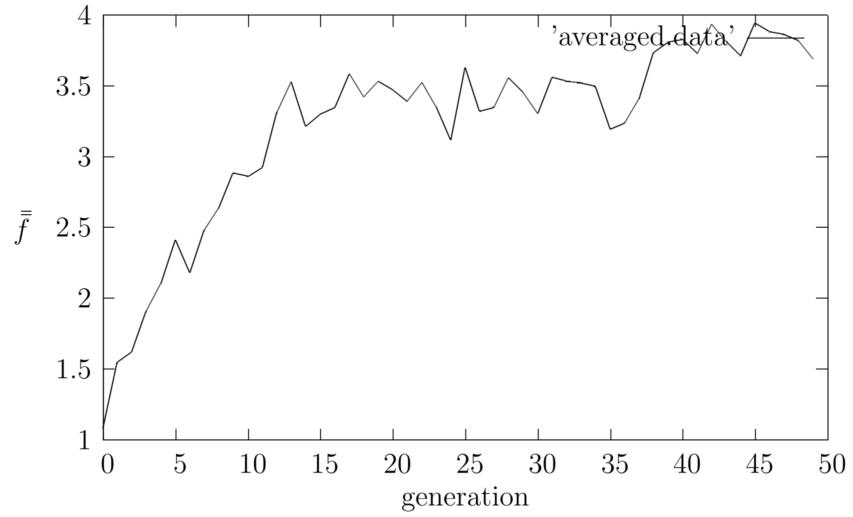

Figure 3: The mean of the mean fitness from three different runs

As one can see, the average fitness appears to approach an asymptote, which is the expected result. The way fitness is being calculated there is a maximum fitness, where all the particles stay alive for the entire duration of the run. The maximum possible fitness is 5 for the parameters I used.

3.2 Coevolutionary Mode Results

One of the difficulties of this project is determining what measurement is important for the coevolutionary results. I ended up making a lot of measurements in the hope that examining them would present a graph comparable to Figure 1 which shows the mean fitness increasing to an asymptote, which seems to be the telltale sign of a fitness function. Indeed, in the limit where one has an infinite population, the mean of the fitness is a monotonically increasing function with respect to time as shown in [4], so I think that provides support to look for increasing functions as candidates for the obscured fitness function.

The results shown used the following parameters: $P_{min} = 5$, $P_{max} = 20$, $c_m = 0.1$, $\sigma_m = 1.0$, radius $= 0.05$, $\Delta t = 0.05$, and collisions are turned on. Figure 4 shows the mean of the particles' reproduction count r_c for each time step. What I thought I might see is a graph that approached an asymptote, much like Figure 1, as the "fittest" individuals got better at surviving and therefore replicating. However, that does not appear to happen.

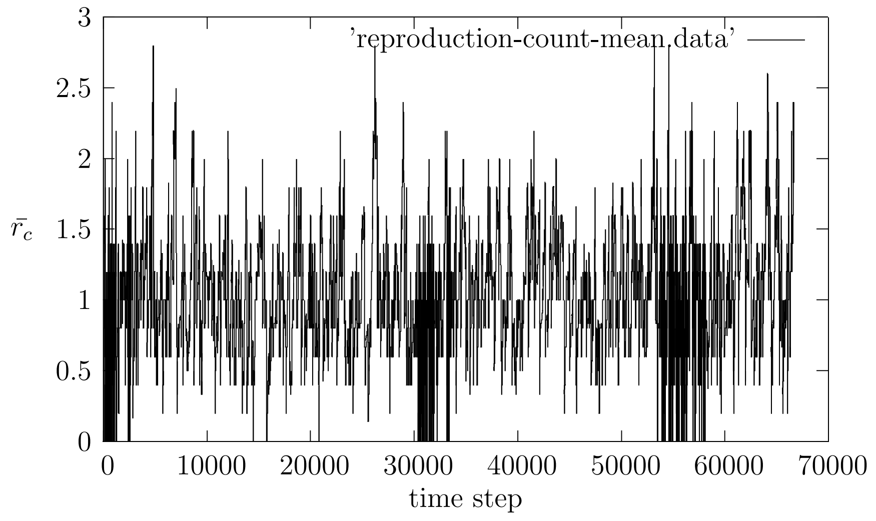

Figure 4: The mean of each particle's reproduction count (run A)

Figure 4 did inspire me to run another simulation and capture the maximum of the reproduction count. Perhaps the newly born particles were adding too much noise to the data.

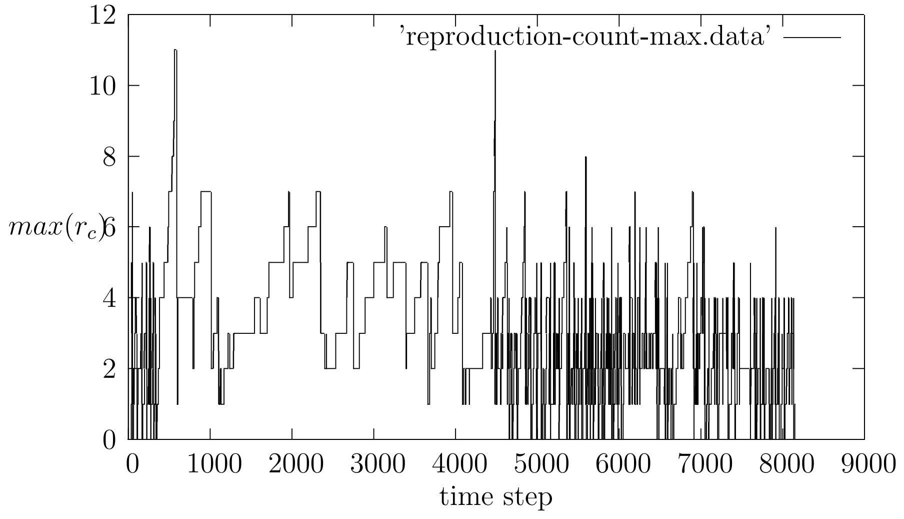

Figure 5: The maximum of the particles' reproduction count (run B)

Figure 5 shows the maximum, and it is less noisy, but the trends do not reflect anything like Figure 1.

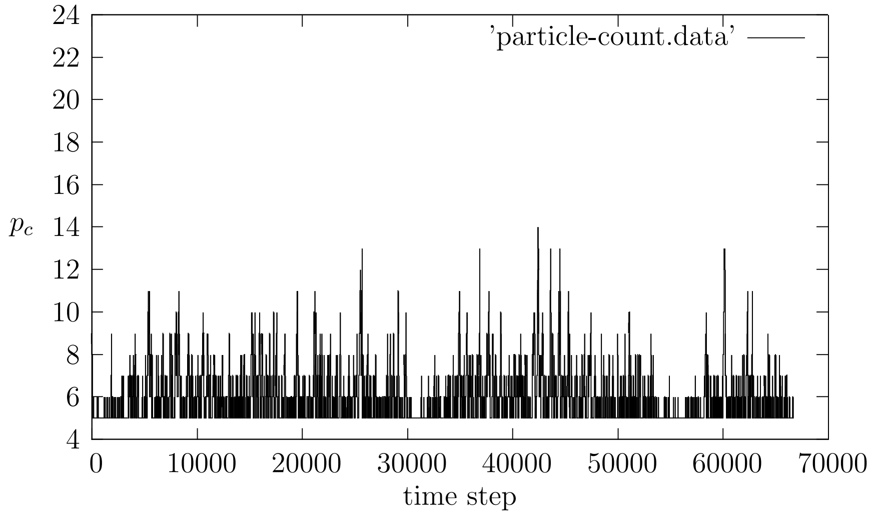

Figure 6: The count of particles (run A)

Figure 6 shows the population count over time, which fluctuates pretty severely. In Figure 7, one can see that there are number of extinctions where all particles die out, and the population is seeded with a new set of randomly created genes whose generation number starts at 0.

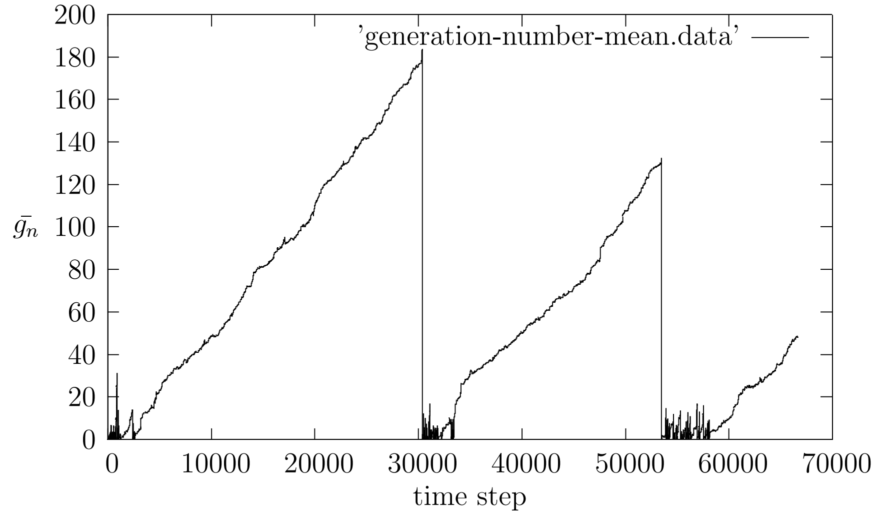

Figure 7: The mean of the particles' generation number (run A)

Figure 7 looks similar to the mean fitness shown in Figure 1, but I believe that is merely superficial. The mean fitness is increasing on average for finite populations, and here the generation number is increased monotonically so its average is increasing unsurprisingly. Although this superficial similarity is easy enough to dismiss, it does bring up questions that I will address in Section 4.

One interesting result is if one takes the real vector that represents the gene **g** and takes the the mean of the magnitude of all the genes as shown in Figure 8.

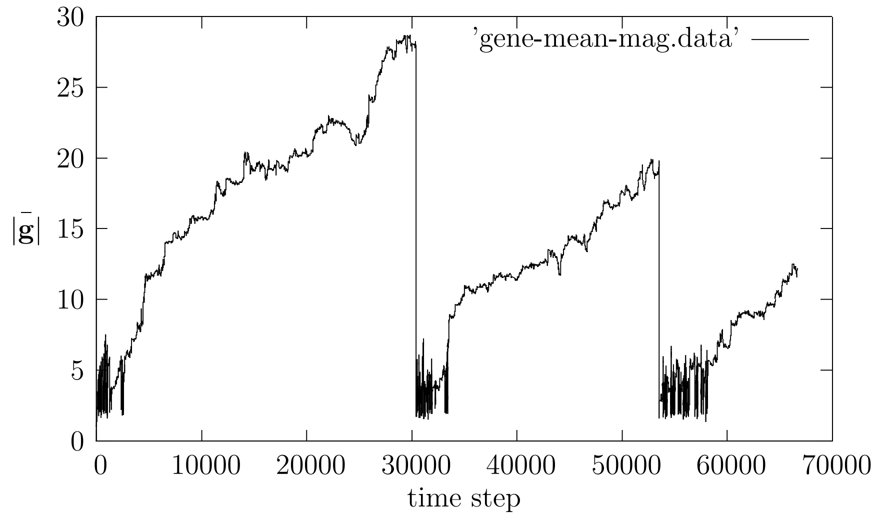

Figure 8: The mean of the particles' genes magnitude (run A)

It appears as though the initial random seeding of genes have mean magnitude of 5 but that as the simulation runs, the magnitude population of genes increase. Figure 8 does bear some similarities to Figure 1, the mean fitness graph, excepting the extinctions.

4 Discussion

The results shown in this paper between the coevolutionary algorithm and fitness mode do not provide a conclusive answer to the question of whether they are qualitatively different. There are a few issues which may have prohibited coming to a direct answer. The cases may differ too significantly to compare them directly, e.g. in fitness mode particles behave independently while coevolutionary mode has collisions. (Overcoming that barrier is not difficult; it just requires more time. One could run the fitness mode with collisions.) The instability of the coevolutionary mode where extinctions result may be due to poor parameter settings. It may also be that the simulation is not rich enough to support the variety of behavior that I imagine would be relevant for a coevolutionary GA. I will discuss some extensions in Section 5.

Comparing Figure 8 and Figure 1 may make one wonder if there is an obscured fitness function that is increasing some element of the gene g_i, but how might one determine whether the gene's change in magnitude is causing the particles to "live" longer and replicate more rather than it just being coincidental? For instance, some non-functional elements may be becoming larger. I do not yet know how to make that discrimination between the causal and coincidental, but I have a hunch that if one perturbs the parameters of interest, one could determine whether they are causally relevant or just coincidental.

Another issue in trying to find what the fitness function may be in a coevolutionary model is that it could easily be a composition of different measurements. Suppose that one had measurements $f(t)$, $g(t)$, and $h(t)$ for a coevolutionary GA. One would want to look at those functions separately of course but one may also want to examine different compositions of them, e.g. $\frac{1}{f(t)}$, $\frac{f(t)}{g(t)h(t)}$, and many more (unfortunately).

5 Future Work

I mentioned that the simulation may not be rich enough to support certain behaviors. The behavior that I had in mind was this. One of the trivial survival strategies for a particle is to just come to an immediate stop, i.e. have a large drag. That is one of the safest strategies. However, it isn't evolutionarily stable because there need only be one mutant who doesn't adopt that strategy and, say, instead moves in a circle around the board. That mutant might end up hitting particles off the board freeing up room for it to replicate. Essentially, it wouldn't allow for a "conspiracy of doves" as evolutionary biologists would call it. One can see traces of predator/prey models between the two different strategies.

I did implement drag in this simulation, and the particles as expected evolved large drag parameters and all came to a stop. Unfortunately, there was no "natural" death dynamic, so once you had all the particles stopped, none would die so their "conspiracy of doves" would last for all time, which is why I kept drag turned off in the results I presented. Implementing a "natural" death dynamic would hopefully alleviate this issue and allow for that predator/prey behavior to develop.

In the interest of seeing the predator/prey dynamic, it might be worthwhile to make mass and radius genetically determined for the particles. Perhaps have a constant mass density, and have the radius determine the mass. That would open up two strategies: be small because you take less collisions but you will be more affected by them, or be big because you won't be affected by collisions but you will take more of them.

$$\ddot{x} = g_0 + g_1 x + g_2 x^2 + g_3 y + g_4 y^2 + g_5 v_x + g_6 v_y + g_7 s_x + g_8 s_y \tag{7}$$

Another feature of interest may be to give the particles some form of sight. Consider Equation 7 where $\mathbf{s}$ would be the relative position vector from the particle to its closest neighbor. It may have a limited sight radius, and $\mathbf{s} = \mathbf{0}$ when there were no particles in its sight radius. This would allow for more interesting interactions than just collisions. It would allow for them to be reactive to their environment.

6 Conclusion

This paper was not able to answer the question it posed: are coevolutionary GAs qualitatively different from GAs employing fitness functions? But it does provide some directions that may lead to more fruitful results.

References

[1] J P. Crutchfield. Is anything ever new? - considering emergence. In *Integrative Themes*, XIX, pages 479–497, Reading, MA, 1994. Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley.

[2] D Dumitrescu. *Evolutionary computation*. CRC Press, Boca Raton, FL, 2000.

[3] Karl Sims. Evolving virtual creatures. In *SIGGRAPH 94 Conference Proceedings*, Annual Conference Proceedings, pages 15–22, 1994.

[4] E. van Nimwegen, J. P. Crutchfield, and M. Mitchell. Finite populations induce metastability in evolutionary search. *Physics Letters*, 229:144–150, 1997.