Meerkat Manor: An approach to simulated, genetic co-evolution

Chris Tanner and Eric Wood

December 10, 2008

1 Abstract/Goal

In our project, we aimed to create a simulated environment where co-evolution occurs amongst two animats: meerkats and hawks. Specifically, we aimed to have meerkats who would learn how to survive attacks from the hawks by (1) obtaining food, (2) watching for hawks, and (3) communicating with other meerkats via signaling. We succeeded in creating this co-evolutionary system, and we detail our representation, approach, obstacles, and results.

2 Introduction

We created a $2\frac{1}{2}$-dimensional simulated environment filled with (1) meerkats and (2) hawks. Our goals were to allow our animats to learn:

- meerkats learn motor-skills to find
- meerkats learn to eat food when it's in front of its mouth
- hawks learn motor-skills to find/eat meerkats well
- meerkats learn that hawks are dangerous and to move accordingly
- meerkats learn that digging and hiding burrows can protect them from the hawks
- meerkats learn that when they receive a signal from other meerkats, it represents hawks are near

We are strictly concerned with and interested in the meerkats' ability to survive and the hawks' ability to survive. That's the main goal, but we suspect and hope that having such will require the meerkats to learn how to dig burrows and signal each other. Clearly, the meerkats have more to learn, so we were primarily watchful of the meerkats' survival rate as we incrementally developed our animats' abilities. We created a dynamic environment that has each of the aforementioned features and learning abilities, and that is our grand project. However, for demonstration and testing purposes, we show that when we remove or adjust some of our animats' capabilities (i.e., ability to burrow or ability to send signals), our system does in fact perform worse, as expected. **More noteworthy is the finding that even when we restrict our animats' abilities, our animats still evolve well and learn survival techniques–often yielded pretty impressive survival-rate results.**

3 Architecture

We chose to develop our code in JAVA, for it offers object-oriented-ness and has a Swing package that allowed us to "easily" create a GUI for our system.

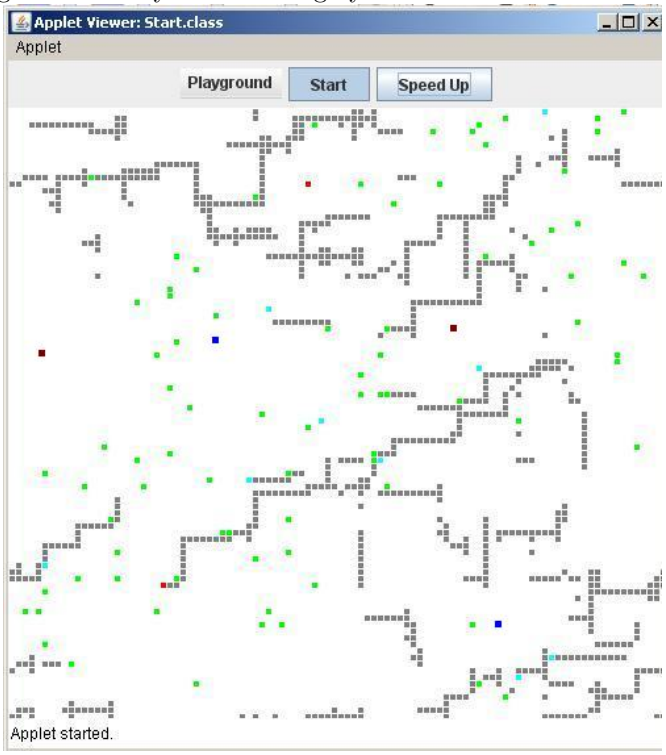
3.1 Environment

Based on [1], our world is graphically represented by a 2D n x n grid. However, it's technically $2\frac{1}{2}$-dimensional because hawks will reside at higher altitudes than meerkats, but may appear to occupy the same blocks as meerkats. Each animat and object is presented by a distinct color and performs an action at each timestep. A given simulation of animats moving around within a grid is called a *playground.* A playground will initially have food objects randomly sprinkled throughout, and the simulation for the playground will last for 1,000 time units.

At each time unit, animats will perform an action, which we'll describe more about later. We took an approach similar to [5] in that within a playground, all of the meerkats will have the same genetic makeup, and all of the hawks will also have their own genetic makeup that is identical amongst their species. Yet, we have 20 playgrounds that are simultaneously running, which enables us to have diversity and evolution.

We show a screen shot of our fully working system in Figure 1. Note, at that moment the meerkats have learned to very effectively utilize burrows (represented by grey blocks).

Figure 1: Our system with highly evolved meerkats and hawks



3.2 Evolution Algorithm

Basically our system is represented by:

- world, which has:
  - 20 playgrounds, where the essential items of each playground are:
    - 20 meerkats with identical genotype
    - 5 hawks with identical genotype
    - 100 food items initially sprinkled throughout, which randomly re-grow at a given rate

During a run of 1,000 time units, the animats move around and perform actions that include meerkats' eating food and hawks' eating meerkats. Even though a given playground's meerkats all have identical genetic makeup, the meerkats will of course behave differently from one another because their surroundings may be different. For example, at time T, if meerkat A has a nearby hawk but meerkat B is on top of a food unit, it is highly likely that their genetic makeup will tell them to perform different acts.

After one generation (1,000 time units), we will have a good indication as to how well the given genotype serves the meerkats within the given playground P. Specifically, we measure how good or bad the genotype was by our fitness functions for each animat:

- MeerkatFitnessFunction() = # of survived meerkats within playground P. Ties will be broken by the total number of food objects eaten.
- HawkFitnessFunction() = # of survived hawks within playground P. Ties will be broken by the total number of meerkats eaten.

These fitness functions were chosen to provide two concurrent goals. Firstly, survival should be the main goal of a species. We chose the amount of food eaten as a tie breaker since food is an essential component to survival. After experimenting with a number of different fitness functions, we found that these performed the best.

At the end of a generation, we will have 20 playgrounds, each with a different meerkat score and hawk score. For our animats to evolve, we pick the top 5 (out of the 20) meerkat genotypes. Let's call these $M_1 \ldots M_5$. Likewise, we pick the genotypes of the top 5 (out of 20) performing hawks and call them $H_1 \ldots H_5$. For each

possible pair of elements in M, we crossbreed them using a 1-point crossover function. And, our crossover point is randomly chosen each time. For example,say we are breeding M_2 and M_4, and our randomly chosen crossover point is at the half-way mark. Well, we also randomly choose which genotype section comes from which. So, M_2 could provide the first half of our new genotype just as likely as it could provide the latter half. We also randomly decide to mutate a gene with low probability.

We do the same thing for the hawks, yielding us with 20 new hawk genotypes. We then combine these with our meerkats in order to give us 20 new playgrounds. We use these new 20 playgrounds to completely replace our original 20 playgrounds. Now that we know how we will evolve and what our fitness function for each animat is, we focus on how we create each animat, their capabilities, and how we represent their brain.

3.3 Food

Food is represented as a square green block with a size of one grid unit.

For our experiments, we initially place 100 units of food on the 100x100 grid, and we grow 1 new food unit (to be randomly placed) at each time step with a probability of .1. Each food unit provides 200 units of energy, and will be eaten fractionally, for meerkats consume exactly 50 units per time step.

3.4 Meerkat Animats

A meerkat's body is represetned as a square red block with a size of one grid unit.

In order for our meerkats to learn each of the tasks in section 2, we want our meerkat to have the following (8) capabilities:

- move in any direction (north, east, south, west)
- rest
- eat food
- dig burrow
- send signal to nearby meerkats

So, at each time step, the meerkat must do one of the above actions. Each meerkat an energy level that would decrease by performing these actions, with each action costing a different amount of energy. When their energy reaches 0, the meerkat would die. The point of having burrows is they may offer safety for meerkats. Otherwise, the meerkats would have no defense from hawks other than running. To mimic nature, we allow the hawks to travel faster than the meerkats. Thus, it will become vital for meerkats to utilize burrows. When a meerkat chooses to rest while occupying a burrow-grid-unit, the meerkat is essentially in the burrow and consequently invisible to the hawk. If he choses to do so, he eats 50 energy units of the food. We set our environment to have 20 starting meerkats per playground.

3.5 Meerkats' Brains

The meerkats must learn that hawks are dangerous and food is good. We chose a very low level approach in creating the creatures to see how complex of behaviors they would produce given low level stimuli and actions. Throughout the project, we took great care to limit the amount of predetermined, high-level, actions the creatures would have. Ultimately, we were interested in the complexity of the levels of behaviors we could achieve given low-level inputs and actions. We were less interested in creating a high level 'run-from-hawks' function, and more interested in seeing if the meerkats could produce this behavior themselves.

We decided to use feed-forward neural nets to represent the brain. The weights would be determined by the genetic algorithms mentioned above. Specifically, we define a *gene* that basically represents a dendrite that has 3 attributes:

1. neuron our dendrite connects from
2. neuron our dendrive connects to
3. weight for our dendrite

With this approach, a neural net is simply defined by an array of genes (a.k.a. *genotype*). And now that we've defined our fitness functions, we know that when we cross our genotypes, we will be changing the weights in our neural net. Note, the connectively will remain fully-connected, regardless of how we cross two genotypes.

Determining the specifics and developing a sound representation was a major part of the project, so we detail them in section 4.

3.6 Hawk Animats

A hawk's body is represented as a square blue block with a size of one grid unit.

The functionality for hawks would be similar to that of meerkats, just instead of stationary food units, their food is meerkats who move. Like the meerkats, they have an energy level, and gain energy by eating meerkats. Also, hawks do not have to learn to eat meerkats; rather, anytime they occupy a grid-unit that contains a meerkat, they automatically eat the meerkat. Our motivation is that this is an innate act. Hawks have no signaling or burrowing capabilities. Thus, they have less to learn and should be able to learn much faster, leading us to suspect that hawks will dominate the meerkats quickly, the with meerkats eventually developing more complex strategies to evade the hawks.

3.7 Hawks' Brains

The structure is similar to meerkat brains; we create a feed-forward neural net whose weights are dictated by a genetic algorithm.

4 Approach (To Represent the Brains)

4.1 Stage 1: Adding Food

After creating our basic environment without food, we added the concept of food. We were unclear how to get the meerkats to find and go towards food.

4.1.1 What Didn't Work

Having recalled my experience from [2][4][3], we tried several neural net approaches. First, we tried using various sensors. We considered the idea of having 1 sensor per eye/antenna of the meerkat. With this, we would know exactly which direction is ideal for moving. Yet, this seemed to complicate the issue too much. We then tried having the meerkats only able to see in front of them, and we would require them to turn around in order to see elements behind them. Likewise, knowing when to turn seemed like a difficult task to learn, so we moved on. We thought it would be okay to have an automatic, perfect honing approach to finding food. This worked by the meerkats' having a food sensor. Anytime that neuron fired and our output neuron told us to go to food, we called a function that automatically moved perfectly towards the food. This was cheap and unrealistic, so we removed it.

4.1.2 What Worked

Eventually, we came up with a good idea: create 4 input neurons for our food sensor:

1. food is north
2. food is south
3. food is west
4. food is east

And have 4 output neurons:

1. move north
2. move south
3. move west
4. move east

We set our food sensor to have a radius of 10 units. We then find the closest bit of food to the meerkat, and fire the neurons that associate with the direction the food is in relation to the meerkat. After we get our sensor input at each time step, we look at our output layer and perform the action of the highest firing neuron. Instead of meerkats intuitively knowing to eat food, we decided to add an extra element of complexity by having an action called 'eat-food', where it is up to the animat to decide to eat or not. We also created another input neuron that would fire when the meerkat shares the same block as food. Our NN now has 5 input neurons and 5 output neurons.

4.2 Stage 2: Adding Rest

Adding food worked well, as meerkats were able to learn how to move appropriately when food is nearby. Mind you, no hawks are present yet. It's merely meerkats and food in a safe environment. Yet, the meerkats were still often too good; **they initially ate all the food quickly and died due to starvation. At best, only a few out of the 20 meerkats would survive,** and it would take a few evolutions before the meerkats learned not to eat so greedily. So, we added the ability to rest, which cost less energy than moving.

We gave the meerkats 4 energy input neurons: 1 for each of the 4 discretized bins that represent the amount of remaining energy. For example, meerkats have a capacity of 500 energy units. So, exactly 1 of the 4 energy input neurons will fire: (1) $0 \leq 125$; (2) $126 \leq 250$; (3) $251 \leq 375$; (4) $376 \leq 500$. The hope was that the meerkats would then act according to their hunger level.

We added a rest output neuron, and our neural net brain is still full-connected such that each of the 9 input neurons has a link to each of the 6 output neurons. This solved the problem of meerkats committing suicide by running out of food to eat. They learned to be patient and not eat so greedily. **Often, all 20 of the meerkats would survive.**

4.3 Stage 3: Adding Hawks

This added element was our biggest obstacle, and was the core of our project. We knew that our moving in a direction had to factor in both nearby food and nearby hawks. We knew that we must learn a fine balance of weights so as to soundly give precedence to each.

4.3.1 What Didn't Work

At first, we had independent, disjoint neural nets, one for the hawk sensors and one for the food sensors. Yet, this approach of disjointness did not allow our meerkats to easily learn that in order to determine which direction to move, we should take into consideration both sets of sensors. We experimented with the idea of having a hidden layer for memory. However, we had two main conflicts with this approach: (1) it appeared it would take an unreasonably long time to learn the weights, as we will not have direct, continuous feedback for adjusting our hidden weights; (2) we could not determine a good way to sum up the weights from the input layer units, as most sigmoid functions would squash our values to a range that appeared difficult to work with.

4.3.2 What Worked

We decided to represent the hawk sensor as its own group of input neurons, just like the previously mentioned food sensors. Thus, each input neuron is fully-connected in a feed-forward manner to our output neurons, and no hidden layer is used. In fact, by the time we decided on this approach, we weren't fully convinced that this fully-connected, feed-forward net approach would be ideal for even the food and resting elements. It all came together at once, and the results were pleasing. We wanted the hawk to be able to move faster than the meerkats. If this weren't the case, then the only situation where they could eat meerkats would be if meerkats were dumb enough not to move out of the way. Plus, they would be able to merely move back-and-forth between 2 adjacent squares like a stalemate in chess. Thus, we gave the hawks the ability to sprint 2 blocks at a time, instead of 1. So, anytime the closest meerkat is more than 1 block away within on a given axis, we move 2 blocks.

At this point, we now have **13 input neurons** (4 for the food sensor, 4 for the energy level, 4 for the hawk sensor, and 1 for on-food) and **6 output neurons** (4 for moving in a given direction, 1 for resting, and 1 for eating food).

Adding hawks definitely leveled the playing field. Meerkats struggled to survive, and they clearly needed a defense.

4.4 Stage 4: Adding Burrows

With the success that we had with our current neural net representation, we decided to follow suit and add burrows in this manner. So, we created 4 more burrow input neurons that would represent the direction of the closest burrow to a meerkat. Additionally, we created 1 dig burrow output neuron. In the early generations, the dig burrow output neuron may fire at completely inappropriate times. Sometimes, it will cause us to dig when there is no nearby hawk, or not to dig when doing so could have saved us. However, in later generations, it's clear that it offers a huge benefit, and the battle of co-evolution begins. More details are in the Analysis section.

4.5 Stage 5: Adding Signaling/Fear

4.5.1 What Didn't Work

Initially, we thought of having 3 unlabeled signals, whereby the meerkats would have to all agree what each signal represents. We thought about using them as inhibitor neurons, but none of our ideas seemed to work. The signal neurons ultimately served as nothing more than as noise, for it took way too long for the meerkats to have such a perfect genotype where they all represent the signals to mean the same thing.

4.5.2 What Worked

Alternatively, we introduced the concept of fear. The main problem before was that the meerkats were not strongly learning that when hawks are nearby, they should enter this different state of having a goal just to survive. So, we figured we could either (1) introduce fear signals, or (2) introduce goals and planning into our system. The latter seemed much too ambitious. Each meerkat has a fear level that jumps up when a hawk is around. They also gain fear whenever they hear a signal. This way they associate signals with hawks, and should perform a defensive manuever.

Our concept of fear signaling works very similar to the other elements we've added. When the fear level in a meerkat reaches a certain level, we fire the fear input neuron. When our signal output neuron fires, we send a signal to each of the meerkats within our signal range. The meerkats learn how to appropriately use their signals so that they aren't too strict or liberal with their alerting.

4.6 Final Brain Representation

Our fully-featured brains are shown in Figure 2 and Figure 3. Note, the input layer is at the bottom, and the output layer is up top.

Figure 2: Meerkat's Brain

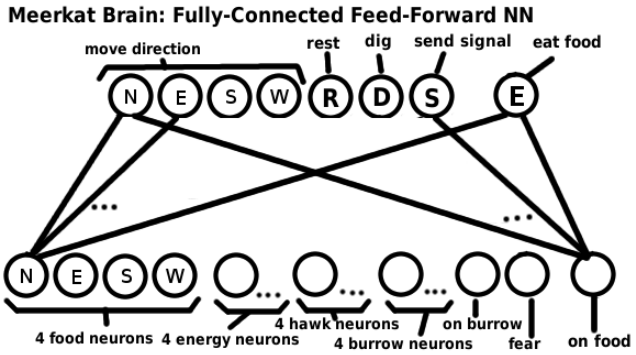
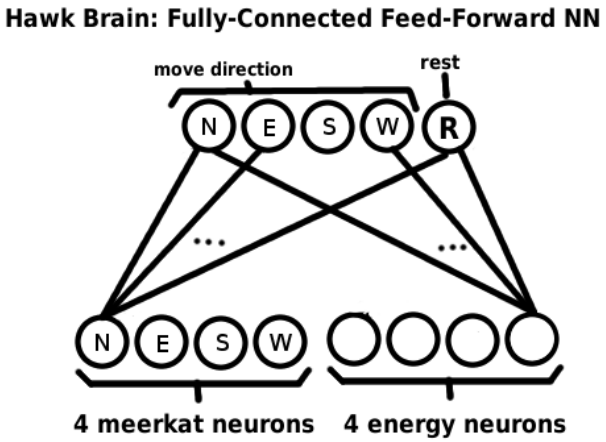


Figure 3: Hawk's Brain

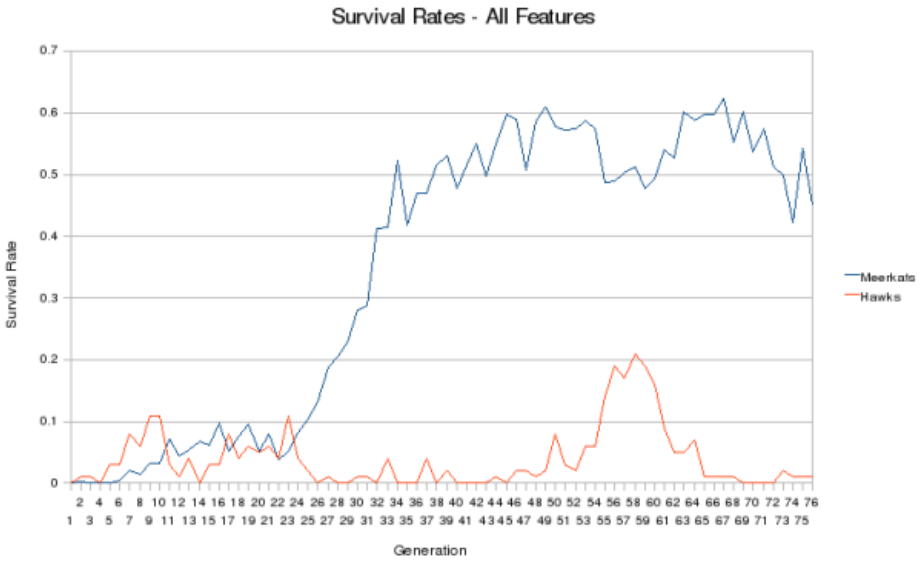


5 Analysis and Results

5.1 All Features Implemented

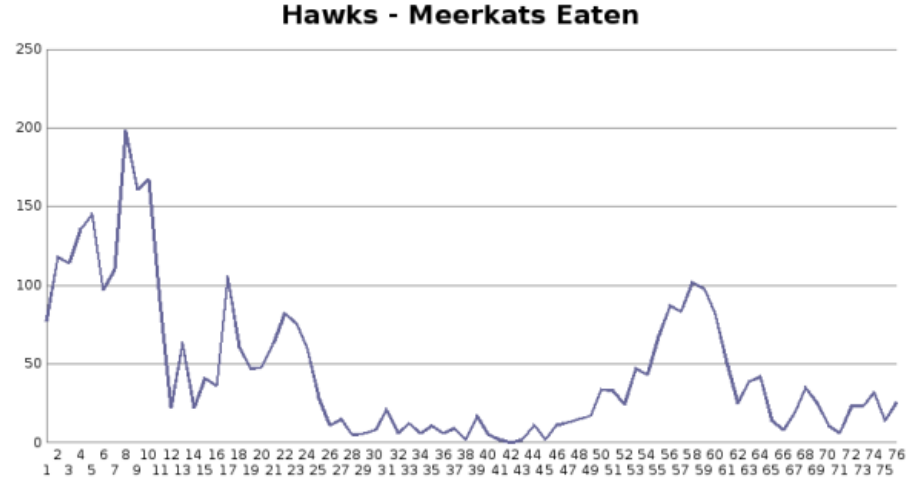
With everything fully implemented, we ran our code and fortunately saw co-evolution ensue. As seen in Figure 4, the meerkats (blue line) got better and better. **Specifically, starting at generation 25, we saw that the meerkats learned how to effectively burrow; when hawks would get close, they would dig and rest in a burrow!** Naturally, as the meerkats learned this, the hawks' survival rate worsened. As the generations progressed however, we saw that there were local battles and the balance would slightly shift back and forth as the species found new strategies to survive (for instance the hawks developed a way to eat hawks at around generation 50, but the meerkats eventually countered their efforts by generation 66).

Figure 4: Survival Rates for System with All Features



The hawks got worse and ate less meerkats as the meerkats learned to burrow (Figure 5).

Figure 5: # of Meerkats Eaten by Hawks for System with All Features



The meerkats not only effectively learned to survive better by utilizing burrows, but their ability to eat food increased too. (Not pictured due to space constraints.)

5.2 Perfect Hawks

We wanted to see how our system would hold up if we made it much more difficult for our meerkats. So, we took away the learning aspect of hawks. Instead, we created "perfect hawks," whereby when hawks sense any nearby meerkat, it will perfectly hone in on the meerkat and eat it. This first few dozen generations were pathetic to

watch. In fact, it was sad to see the massacre of meerkats, as they stood no chance against the perfect hawks. Nevertheless, eventually the meerkats learned that they must burrow very frequently. After a while, many meerkats were surviving (Figure 6), as they were outwitting the hawks. Consequently, less hawks were surviving! (See Figure 7).

Figure 6: Survival Rates for System with Perfect Hawks

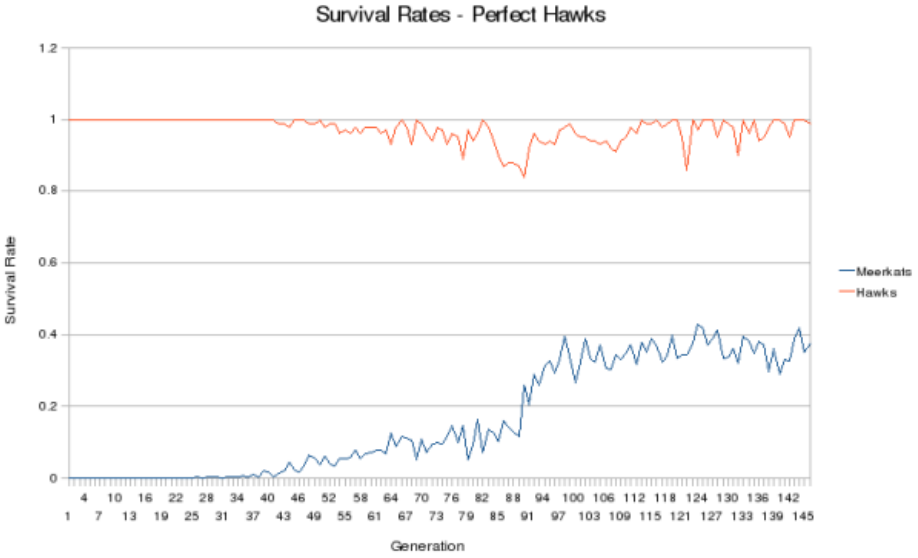
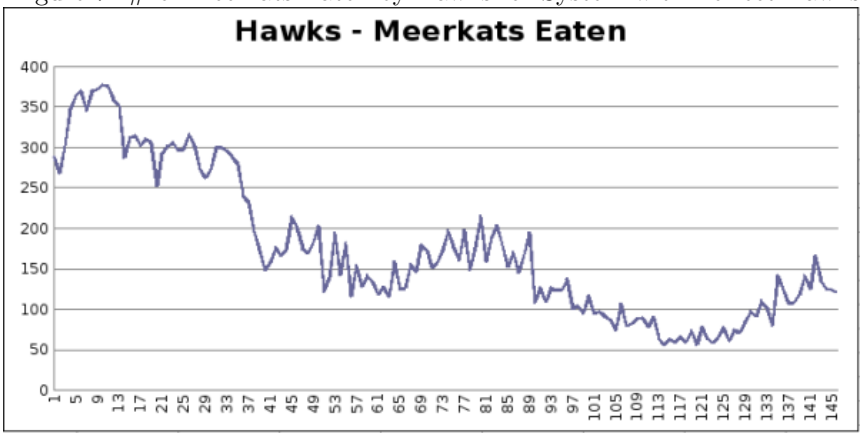


Figure 7: # of Meerkats Eaten by Hawks for System with Perfect Hawks



5.3 No Fear Signaling

When we added the feature of fear signaling, it seemed to greatly help the meerkats' ability to survive. So, we temporarily removed this functionality (while making the hawks learn like originally) and tested how the system would behave again. What we found was that the meerkats did eventually learn to evade the hawks, but it took much longer than when they had signalling. In the previous test, the meerkats were able to learn to avoid the hawks through burrowing and running within 25 generations. Without signalling however, the meerkats took much longer to learn to survive. Also interesting to note is that with signalling, the meerkats became so good and avoiding the hawks that the hawks were for the most part unable to eat any meerkats, while without signalling the hawks were still able to catch a decent number of meerkats. There was also a lot more variance in the survival rates between the two species, as they each developed strategies to outwit the other.

Figure 8: Survival Rates for System with No Fear Signaling

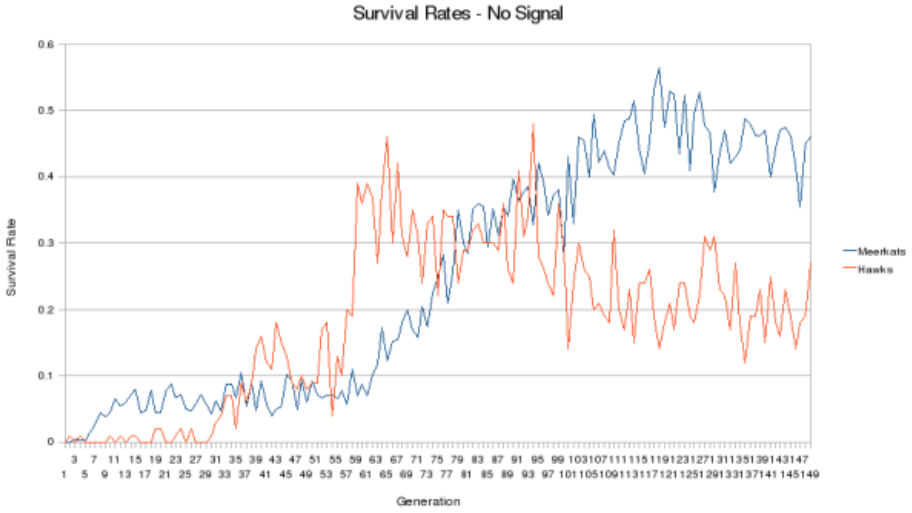
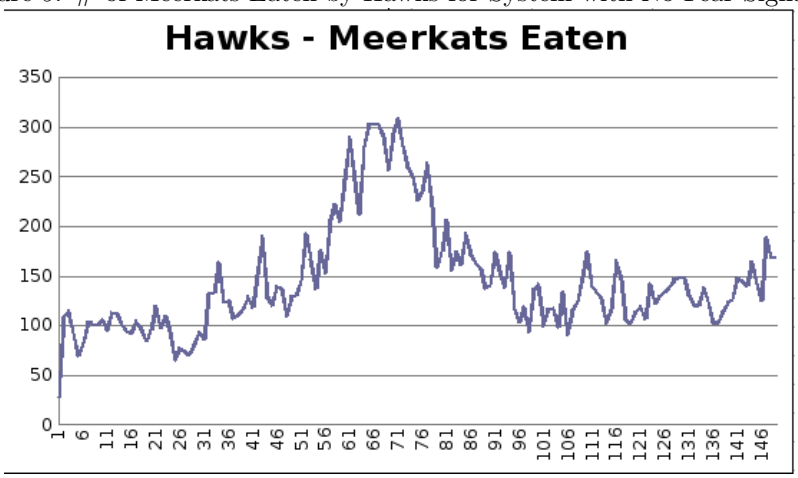


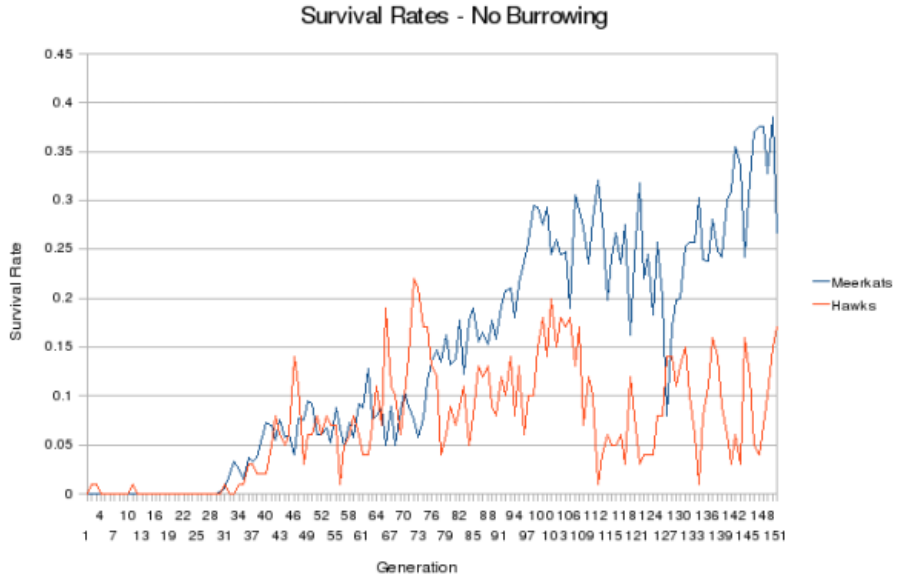
Figure 9: # of Meerkats Eaten by Hawks for System with No Fear Signaling



5.4 No Burrowing

Burrowing is clearly the biggest help for meerkats to evade death. Our last study was to see how the meerkats faired without the ability to burrow. What we found was that initially neither species was able to survive. The hawks would eat all of the meerkats, and then have nothing left to eat and end up dying off. Eventually, however, the meerkats managed to find some strategies to evade the hawks. Once this occured, they were able to survive longer, which then allowed the hawks to have something to eat the entire run. We found this interesting in that the survival of one species actually helped the other. During this test, we found that the meerkats were much more active, and rarely stopped to rest. They essentially learned that by constantly moving, they were much more likely to evade the hawks than sitting still. We were surprised to see how well the meerkats eventually were able to survive, however naturally they did so at a lesser rate than with burrowing. (See Figure 10.)

Figure 10: Survival Rates for System with No Burrowing



6 Status of Work/Conclusion

6.1 Impressions of Our Work

Overall, we are very pleased with our system. It appears that regardless of the situation and the capabilities of our animats, our animats are able to evolve. Sometimes it takes many generations to do so, yet the learning does occur. Our results proved to us not only that we were able to create a co-evolutionary dynamic system, but that our representation of the brain works well and is highly suiting. We assert this because when we take away some of the functionality, it is apparent that it's definitely harder for the animats to learn. This suggests that our fully-featured world definitely benefits our animats in the way we intended. If our experiments shows that the animats could do equally well even when they had no capability to burrow or send fear signals, then our findings would have been disappointing. Fortunately, this was not the case.

As for remaining work, we pretty much did everything that we outlined in our proposal. We didn't add trees to our environment, but this wasn't a huge element in our proposal anyway. We have many ideas that we wished to explore, and Chris hopes to implement them by extending this project for his Master's project.

6.2 Division of Labor

Chris looked for a paper that seemed good and related (our approach was similar to [5], but we decided not to model ants), and looked for GUI systems that could serve as a basis for our GUI. Eric is better at software design, and he handled the majority of the serious design decisions and implemented much code. Chris had more experience with neural nets and genetic algorithms, so he contributed towards the research ideas. Eric and Chris both worked together to decide how we would represent much of our system, and we both looked and found bugs along the way. Eric wrote much code, and the design really is quite impressive. Both team members contributed to write this paper.

References

[1] Ant colony - java applet. http://djoh.net/inde/ANTColony/applet.html.

[2] *Fundamentals of Neural Networks*. Prentice Hall, 1st edition, 1994.

[3] *Machine Learning*. McGRAW-HILL, 1995.

[4] *Artificial Intelligence: A Modern Approach*. University of Chicago Press, second edition, 1998.

[5] D. R. J. Robert J. Collins. Antfarm: Towards simulated evolution. *Artficial Life II*, 1991.

A Appendix

A.1 Code Files

We wrote 3,235 lines of codes.

Our code is organized into (3) main directories, as follows:

- /src/Brain directory (742 total lines of code):
  - Brain.java
  - Dendrite.java
  - DigBurrow.java
  - EatNeuron.java
  - Gene.java
  - HawkBrain.java
  - HawkMoveEastNeuron.java
  - HawkMoveNorthNeuron.java
  - HawkMoveSouthNeuron.java
  - HawkMoveWestNeuron.java
  - HawkrestNeuron.java
  - HawkSprintEastNeuron.java
  - HawkSprintNorthNeuron.java
  - HawkSprintSouthNeuron.java
  - HawkSprintWestNeuron.java
  - MoveEastNeuron.java
  - MoveNorthNeuron.java
  - MoveSouthNeuron.java
  - MoveWestNeuron.java
  - Neuron.java
  - restNeuron.java
  - Signal1Neuron.java
- /src/object directory (2,241 total lines of code):
  - BrainOld.java
  - Burrow.java
  - Food.java
  - Hawk.java
  - Meerkat.java
  - MyFilter.java
  - Node.java
  - NodeObject.java
  - Nodes.java
  - Playground.java
  - PlaygroundMouseListener.java
  - PlaygroundMouseMotionListener.java
  - SerializerNodes.java
  - Signal.java
  - Traces.java
  - Tree.java
  - Universe.java

- /src/view directory (252 total lines of code):
  - buttonListener.java
  - buttonSpeed.java
  - buttonStart.java
  - JMenuPanel.java

A.2 Trace Run

See trace_run.csv