Project 1

Group 9 Report

September 29, 2004

Deepti Jindal

Sasha Davidev

Jeffrey Regier

Rob Tobkes

Organism2 is based on an electronic world populated by virtual organisms. The goal is to design an organism that can survive in all conditions for as long as possible. The focus of our group was to sustain a high amount of energy per organism and the maximum possible number of organisms on the board in a multiplayer game.

Our initial designs were motivated by the strategy of Black Plague. We felt that their strategy of remaining dormant rather than behaving aggressively was key to winning.

We quickly concluded that onion farms were largely not worthwhile. The only situation where onion farms might be remotely useful would be on a large board and where the doubling would be very high. In fact, even then, there would not be enough time to communicate when to create a farm before an enemy would come close and possibly eat the food thus destroying the farm you were trying to create. Another problem with farming was that by creating a farm you lose a lot of time communicating. If the other organisms multiply very quickly then you are constrained to a very small area with very few organisms.

Our first player was "Lavender Emo Crushers" which based its actions according to "emotions" based on an estimation of p. The second player we designed was "Geriatric" which based its movement on energy and enemy density.

PLAYER #1: LAVENDER EMO CRUSHERS

Motivation

We built our first player with the intent to compete against Black Plague. We noticed that they maintained an excellent balance of expansion to laziness. Early in the game their strategy remains incredibly conservative, but as other organisms start dying out, Black Plague turns very aggressive. We also noticed that many of the players that died early made the mistake of expanding too quickly and spreading themselves too thin.

Lavender Emo Crusher was modeled to behave mainly in response to the amount of food on the board. To control its behavior we relied on four main strategies:

1. Estimate the amount of food on the board. (This was attempted by many of the groups from last year, but it seemed that most were unsuccessful.)
2. Reproduce onto food.
3. Moving is expensive.
4. Maintain a consistent level of energy per organism

States and Strategy

In our Emo Crushers player, we implemented the previous four strategies based on emotions (states) of our player. Just as in humans, our player can exhibit a broad display of emotions; and also just in humans, behavior modifications occur based on one's emotional state.

Specifically, we implemented 9 distinct states, ranging from Happy to Sad. The following is a complete list of states:

1. Full
2. Eating
3. Happy
4. Apathetic
5. Sad
6. Searching
7. Pregnant
8. MoveToFood
9. MoveAway

We will now discuss each state in greater detail. The first 5 states are based on the automaton's current energy level and these are assigned first when determining an organism's current emotion.

Full:

An automaton is considered to be full when its current energy level plus the energy gained from eating a unit of energy is greater than the maximum allowable energy. In terms of game variables, an automaton is full when the following condition is true:

```
energyleft + u > M
```

In this scenario, the simulator will not let the automaton eat. A full automaton will not move (it clearly needs time to digest its food).

Eating:

An automaton is eating when it is on a square that contains food and it is not full. In terms of game conditions, an automaton is eating when the following two conditions are true:

1. `energyleft + u < M`
2. `foodleft > 0`

An automaton that is eating will remain consume the food on the square it is currently on. It will continue to remain in the eating state until at least one of the above conditions is false.

Happy:

An automaton is happy when its current energy level is greater than 70% of its total energy but it is not eating or full. A happy automaton will not move because this we believed this organism is better off waiting for food to come to it as opposed to expanding valuable energy searching for energy.

Apathetic:

An automaton is apathetic when its current energy level is less than 70% max level but greater than 40%. An organism that is apathetic may search for food if certain food quantity conditions are met. If these conditions are not met, an apathetic organism will not move.

Sad:

An automaton is sad if its energy level is less than 40% of the maximum energy level. A sad organism may also search for food if certain conditions are met. The only difference between a sad and an apathetic organism is that a sad organism is not allowed to reproduce.

The final four states are based on the food conditions on the board and the previous five emotions. An organism may originally be assigned one of the previous five emotions, but that emotion may be overridden by one of the following four.

Searching:

There are two necessary conditions in order for an organism to be allowed to move. Because our group evaluated moving to be an expensive operation, it is only allowed when an organism needs to find food and when the probability of the organism finding food is good.

1. The organism has either emotion Apathetic or emotion Sad.
2. Either case a or case b
   a. The probability of finding food is greater than 30% where the probability of finding food is defined as:
      `1-(1 - estimate of food density)^(energyleft / v * 3)`
   b. Our estimate of food density < .005 and our estimate of enemy density < .05

MoveToFood:

An automaton will move to food that it can see as long as it is not currently eating or currently full.

Pregnant:

There are two necessary conditions for an organism to be pregnant:

1. The organism has an emotion of eating or full. This ensures that our organisms only reproduce when they are high in energy or are replenishing their energy.
2. The probability of the child finding food is greater than 40% where the probability of the child finding food is defined as:

```
1-(1-estimate of food density) ^ (energyleft/v * 0.5 * 3)
```

However, an organism that is in state Pregnant will not necessarily reproduce. At first, we developed a strategy when the only time an organism can reproduce is if it is pregnant, and the child organism will be placed on a square of food. Even though this organism was relatively stable and each organism had a high level of energy, this automaton was too conservative. In multi-player games, this organism would be surrounded by others, and trapped. It was unable to move because of our strict limits on when an organism can move and unable to reproduce because it could not find a free square of food (and was not allowed to look for one), this organism often lasted late into the game but could not win or dominate a game. So our final organism will split on food if possible, but should split about once every 500 turns when pregnant only if our estimate of $p > .005$.

MoveAway:

In an effort to de-cluster parent and newly born child, a parent will move away from its child as long the parent does not see food and is not currently eating.

Food-Estimation Formula:

```
1-(1-estimate of food density) ^ (energyleft/v * 0.5 * 3)
```

- **<energy left> / v** : the number of moves that an organism can make before running out of energy.
- **3** : the number of new squares an organism sees every turn
- **<energy left> / (v * 3)** : total number of new squares that will be seen before the organism dies
- **1 - <estimate of p>** : the probability of an organism never finding food
- **(1 - <estimate of p> ^ (<energy left> / v * 3)** : probability that the organism will not find food in a new square before it dies
- **1 - (1 - <estimate of p>) ^ (<energy left> / v * 3)** : probability the organism will find food on a new square
- The exponent in the probability of a child finding food is multiplied by .5 because the energy of the child will be half that of the parent.

Communication and Estimation

Early in our discussions, we suggested the use a simple communication algorithm to relate game board information from player to player. We had initially allocated the eight bits thus: four to identification, two to a food quantity estimate, and the remaining two to an enemy density estimate. Our original idea was to acquire information from every friendly player and to increment one of the four buckets for both enemy density and food quantity, ranging from very low for bucket 0 and high for bucket 3. These buckets would then be weighed and combined with the player's internal information. The weights would be determined by the lifespan (experience) of the player receiving the information.

Additionally, we decided to make heavy use of the parent-child 32-bit key. We divided the 32-bits as such:

0-3: Significant digit of the enemy density estimate

4-5: Number of zeros after the decimal point for enemy density

6-9: Significant digit of the food density estimate

10-11: Number of zeros after the decimal point for food density

12-15: Number of offspring the parent organism has produced, in multiples of 10

16-31: Exact number of turns the game has been played.

For example, the number .007 would get translated to binary 100111, the leading 10 representing the 2 zeroes after the decimal point and the 0111 representing the significant digit of 7.

Interestingly, using a simple [squares with food seen/all squares seen] ratio, we were able to estimate our probability of finding food with within about 100 turns. Though not exactly a measurement of p, the estimate allowed us to change our strategy based on what we felt was our probability of finding food. Our estimate is, in fact, the probability of finding food, regardless of whether it is new food or old food.

As each organism encounters food or other enemies, they add to the data they have received from their parent. With each turn played, the estimates solidify. Since p remains constant throughout each round, we felt that there is no reason to use a moving average. In multi-player games, the board fills up very quickly, so the probability of finding food does not change significantly since overall player density remains fairly constant. In single player games, as our own organisms fill up the board, the probability of finding food does decrease since our own density increases, but we felt it unnecessary to make exceptions since we were able to consistently survive single player conditions.

We prototyped a moving average for calculating player density, but found that our performance was not improved.

In our final version of the player, we did not end up utilizing player-to-player communication though we did implement it since our food estimates were accurate. We

simply identified if another player is friendly and adjusted our enemy density accordingly.

Tournament Results:

The multiplayer tournament results are not interesting. Our player performed decent, not great but not terrible. We survived in 2 games and lasted over 4000 in a third game. Our results are most likely due to our conservative reproduction strategy. Emo Crushers does not expand quickly at the onset of the game and is easily overwhelmed by some of the other players who search and reproduce more aggressively. Geriatric, our other organism, was tuned to multiplayer performance and subsequently returned superior multiplayer results. Please refer to the Geriatric section on multiplayer results for more information.

However, the single player tournament results demonstrate the strength of the Emo Crushers. The player survived every scenario and did so with striking consistency.

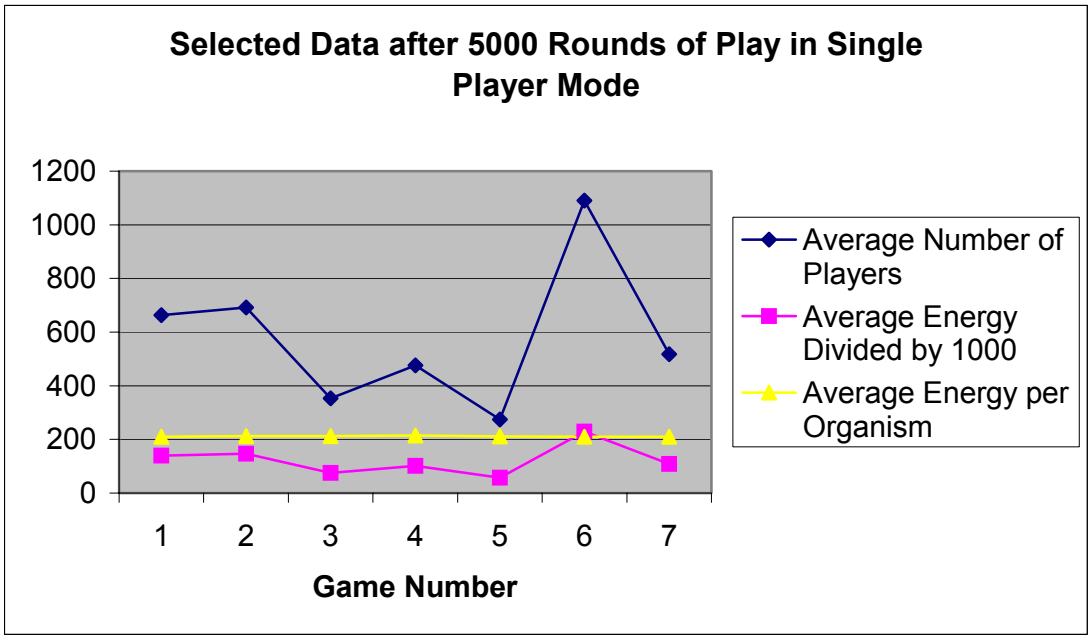
One of the aspects we had discussed as a group is the importance of the energy to organisms ratio. Our single player results exhibited even more consistency than we had hoped. Though the average number of players and energy varied greatly from configuration to configuration, the ratio of energy to players was the most consistent of any group.

Below is a chart of every player that participated in the single player tournament, their average energy per player over all the rounds, and the standard deviation of the energies per player with our players highlighted in yellow. Lavender Emo Crushers had the lowest standard deviation any of the organisms, with an incredibly low 1.518.

Rank	Player	Avg. Energy Per Player	*Standard Deviation*
1	Group9Player1	211.476	1.518
2	Group0Player3	134.77	2.14
3	Group8Player2	227.27	3.918
4	Group6PlayerAG2	179.628	5.594
5	Group6PlayerN	149.918	6.748
6	Group4Player1	121.482	7.962
7	Group4Player2	134.675	8.42
8	Group0Player1	142.504	8.474
9	Group9PlayerJD	192.207	8.591
10	Group8Player1	212.884	9.469
11	Group3Player2	234.346	11.272
12	Group1PlayerFinal	237.018	14.438
13	Group0Player5	215.931	16.753

14	Group5Player5	268.737	34.318
15	Group3Player1	158.083	35.749
16	Group5Player9	254.279	39.709
17	Group2Player3	137.342	47.58
18	Group7PlayerFinal	192.327	49.513

Focusing on Lavender Emo Crushers, we can clearly see the close relationship between energy and players over all seven rounds played. Notice that the average energy per organism line is practically flat over all seven scenarios.



Because of its consistency and ability to perform in all enviroments, the Lavander Emo Crusher is very good at two player games, where the settings are similar to single player mode. Granted, other players can defeat the Emo Crusher on the default settings, but by varying p and q to the disadvantage of some of the more robust players, the Emo Crusher will come out on top.

Player #2: Geriatric

Motivation

The motivation for the design of "Geriatric" was a realization that the game occurs in two distinct stages. During the first stage, organisms do not cover the entire board, whereas during the second stage, the board is saturated with organisms. During this second stage, Lavender Emo Crusher was highly effective; its inactivity was ideal once the board was dense with other players. However, during the first stage of the game, the passive nature of Lavender Emo Crusher caused it to cede control of the board to more aggressive organisms.

Estimation of Food and Energy Density

Though these two stages of the game appear quite distinctive to observers, programming organisms to recognize these stages is far from trivial; each organism has only limited perspective on the state of the board. Lavender Emo Crusher attempted to estimate energy density by averaging its experience over the entire game. The problem with this approach is that taking an average over the entire game results in reliance upon outdated estimates. Computing a more up-to-date set of estimates about the state of the board was the first step towards creating organisms that promptly identifying the transition between a game's stages.

"Geriatric" attempted to estimate the current density of energy and enemies on the board. Accurate and up-to-date estimates were made using techniques inspired by the design of Kalman filters. The average amount of food and enemies seen during the entire lifetime of the organism would be averaged. However, unlike the averaging done by Lavender Emo Crusher, this average would weight more recent estimates more heavily.

Timely estimates of food and enemy density made it possible to distinguish when the game had transitioned from stage 1 to stage 2. Thus, there was no need to maintain a constant strategy across stages. Similarly, there was no need to relay on a fixed number of rounds occurring before predicting that the transition between stages of game play occurred; doing so would be inferior to relying on estimates of food and enemy density, since different game settings cause this transition to occur at varying numbers of rounds.

Movement based on Estimates

The technique of trial and error was used to methodically determine appropriate behaviors for each stage of the game. During the first stage of the game, it was determined that maximally aggressive behavior resulted in the best performance, in terms of both the total amount of energy and the number of organisms living. During the second stage of the game, even occasional movement was not overly helpful, except, of course, if moving directly on to food. One notable exception to this occurs during the second stage

of games with high amounts of food, when movement away from clustered organisms is seemingly helpful.

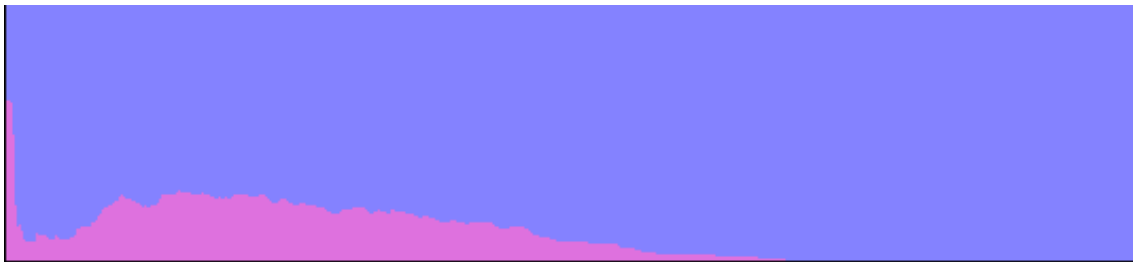


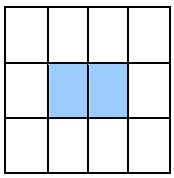
Figure 3: Lavender Emo Crushers versus Geriatric (default settings)

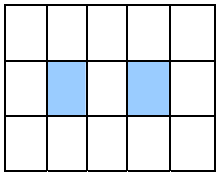
Figure #3 (above) shows the value of these changes. The decline of Lavender Emo Crushers is not rapid, because its conservative late game strategy is very similar to that of Geriatric. However, this decline is inevitable, given the huge lead developed by Geriatric during the first stage of game play.

Strategy

The strategy we used was instead of attempting to calculate the values of p and q, we would calculate an estimate of energy density. This would tell us approximately how much energy is around us at any given time. We noticed that the accuracy would be fairly correct. When there would be very little food then our densities would range way below 0. When the environment contained lots of food then our estimation would be way over 10. This was a useful factor when implementing declustering. We found that in a surrounding where there was a lot of food then it was beneficial to cover as much of the board as fast as possible. When standing next to your own organism then you can't read the maximum number of cells possible thus not allowing you to cover as much quickly.

In the first example our two organisms can only see 6 squares, as opposed to the second example our organism can see 7.





Another major factor that affected the moving was the calculation of enemy density. If enemy density is really high then our organism goes into a lazy state. When the density is low then the organism moves in a straight line and every now then will randomly change direction and keep going in that direction.

For reproduction the strategy we used was reproducing only when we see food. We either reproduce onto the square of food, or if food is underneath us we stay on the food and reproduce next to it. We noticed that by reproducing right away at the beginning then we have automatically cut our energy level down to half, which leaves very few moves to find food. Thus in the condition of very little food we should wait to find food.

In this organism the only time there was communication was for declustering in an environment with lots of food. Thus, Geriatric did really well in conditions where food was really high.

Performance in Multi-Player Games

We were able to maintain a population in multi player mode in all of the games except where food was very low (X=75, Y=75, u=100, v=10, p=0.0020, q=0.0) and the default where the board was small (X=25, Y=25, u=100, v=10, p=0.01, q=0.02).

game	ranking	18 being highest		population	energy	survivability
1	6		12	58.54	11280.2	-1
2	1		17	1134.08	211281.6	-1
3	6		12	38.35	7363.4	-1
4	2		16	99.91	19610.2	-1
7	4		14	114.39	23203.1	-1
8	5		13	42.44	8279.8	-1
5	6		12	0	0	3204
6	11		7	0	0	1677

1	[X=50, Y=50, u=100, v=10, p=0.0050, q=0.01]
2	[X=100, Y=100, u=100, v=7, p=0.02, q=0.2]
3	[X=40, Y=40, u=100, v=2, p=0.0080, q=0.0020]
4	[X=40, Y=40, u=100, v=10, p=0.0050, q=0.0020]
5	[X=75, Y=75, u=100, v=10, p=0.0020, q=0.0]
6	[X=25, Y=25, u=100, v=10, p=0.01, q=0.02]
7	[X=50, Y=50, u=100, v=10, p=0.01, q=0.02]
8	[X=50, Y=50, u=100, v=10, p=0.0050, q=0.0]

Performance in Single-Player Games

In single player mode we had close to the highest population in all the games except for when food was very low (X=75, Y=75, u=100, v=10, p=0.0020, q=0.0) - game number 4.

game number	Ranking	highest rank = 18	Population	energy
1	6	12	754.35	135562.9
2	2	16	698.25	141065.6
3	1	17	478.43	93203
4	14	4	120.22	21709.9
5	2	16	296.86	58462.5
6	3	15	1183.65	232737.2
7	1	17	725.41	141278.9

game number				
1	[X=50, Y=50, u=100,	v=10,	p=0.0050,	q=0.01]
2	[X=40, Y=40, u=100,	v=2,	p=0.0080,	q=0.0020]
3	[X=40, Y=40, u=100,	v=10,	p=0.0050,	q=0.0020]
4	[X=75, Y=75, u=100,	v=10,	p=0.0020,	q=0.0]
5	[X=25, Y=25, u=100,	v=10,	p=0.01,	q=0.02]
6	[X=50, Y=50, u=100,	v=10,	p=0.01,	q=0.02]
7	[X=50, Y=50, u=100,	v=10,	p=0.0050,	q=0.0]

From the results we can see that our ranking was in top 3 for 5 games and the other two games we ranked 6 and 14.

Transitional Strategies

Various versions involving different strategies were tried to improve the performance of the organism. As compared to the inactivity of Lavender Emo Crusher, we used a greedy approach in moving to find food and stopped only when energy levels were extremely low. This methodology was effective for the first 1000 rounds and the population dominated the board, but by 1500 rounds the population tapered to extinction. This led to the idea of a two-stage game. The late game strategy was to reduce the amount of movement and increase the energy level at which the organism would reproduce, as the organism aged. This was still not effective in sustaining the population over 5000 rounds because reproducing in the early stage consumed excessive amounts of energy. However, the idea of two-stage game was retained and used in Geriatric.

In another version of the organism, the decision to move was based upon p, the probability of spontaneous appearance of food, and the ratio of energy consumed in moving/reproducing, v, vs energy consumed in staying put, s. However, this approach

was not successful because of the limitations in variability of v and because q, the probability of food doubling, was not included in the algorithm.

Primary Contributions

Rob: I created the "only reproduce on food" idea and implemented the communication strategy. I also worked with Sasha on the development and fine-tuning of the emotion strategy.

Sasha: I developed and implemented the emotion strategy, including the ways to estimate food and the equations for the probability of finding food.

Jeff: I worked with Deepti and built and observed many different organisms, in order to discover which techniques most impacted performance. We concluded that the game was best viewed as occurring in two distinct stages. I found that density of food and energy is best estimated by forming a weighted average of observations. I created "Geriatric" based on these findings.

Deepti: I tested many versions of the organism with Jeff. We worked on the two-stage strategy of the game. I attempted to create an organism whose level of activity depended on its estimates of "p" and ratio of cost of moving vs. staying still and implemented the idea of sliding thresholds. However when merged with 'Geriatric' it had a negative impact. I also added de-clustering to "Geriatric" based on energy density.