Group 5 Final Report

Harjot Gill, Yang Li, Tim Miller

September 26, 2013

1 Introduction

This report describes our approach and observations while working on the *Organisms* simulation problem. The simulation is a class of popular *Discrete Cellular Automaton* model, similar in spirit to *Conway's Game of Life*.

The simulation world is a finite grid in which virtual organisms occupy individual cells in the grid. These virtual organisms can choose to move, eat or reproduce in each discrete cycle of the simulation. They are also able to "see" the neighboring cells and are thus able to sense the presence of food and/or other organisms. The simulation is subject to various rules and constraints. Each organism has finite energy and there is a cap on maximum energy an organism can accumulate. Energy is replenished by consuming food and it is consumed in every cycle depending on organism's action - i.e. moving, reproducing or even staying still consumes some finite amount of energy. An organism ceases to exist when it's energy level drops to zero. Food may appear on any empty cell in the simulation world with some finite probability and it may increase by a unit with some finite probability. These probabilities are not known to the organism. The maximum amount of food that is available in any cell is also bound to a finite number. An organism may also "communicate" with nearby organisms by broadcasting an external state, that can used in a versatile manner. The simulation can be run in single-species mode or in multiple-species mode. The winning conditions are not concretely defined but some of the factors that can be considered while judging a winning strategy can be average energy per organism, average extinction rate, total number of organisms for a species vis-a-vis other species etc. after some finite number of rounds/cycles.

Our group came up with various approaches to the simulation and depending on results, we pivoted a few times. Initially, our approach focused primarily to maximize the average energy. We developed an organism brain which used different *threshold* levels to decide whether to move, reproduce or to stay still. The implementation assumed fairly high probability of food appearing on the board. However, the strategy fell flat in conditions where food was scarce. We then implemented a strategy in which the organisms tried to gauge the probability of food appearing and doubling, either individually or collectively using some means of communication. We also came up with mathematical equations to determine the "utility" of making a move, consuming food, moving or staying still in each cycle of the simulation. The utility functions allowed us to choose the "best" move for an organism.

However, the release of *"Sheep"* organism from the previous undertaking of this class completely changed our thinking of this game. We realized that our strategies were not agile enough to adapt to multi-species simulations and our organism species was being constantly starved into extinction by a fairly aggressive reference organism species. We discarded the old approaches in favor of a *"Flood"* based approach, which aimed to reproduce effectively in the simulation arena so as to avoid possible extinction by other organisms. To achieve balance, an organism initially tended to stay at one place, then it transformed to aggressive behavior and the older it got, it again preferred to stay at one place. We found this implementation to be quite effective against multi-species simulations and our implementation was even able cause extinction of the reference *"Sheep"* organism in all scenarios, even in the scenarios when it was outnumbered by 1:5 initially at the start of simulation. On the side, we also took cues from strategies used by other groups in the class and we particularly noticed that structured/patterned farming colony approaches were quite

effective in increasing average energy of the organisms. We also implemented a *"Pattern-Maker"* organism brain which tried to form structured farming colonies, so as to achieve the sustainable balance between number of organisms in the simulation and food growth rate. We found this implementation quite effective in single-species simulation.

The rest of the report is structured as follows: Section 2 describes our initial observations about the simulation world and initial strategy. Section 3 provides a 10,000 feet view of various strategies that went into our implementation. Section 4 discusses in great detail, our implementation and how we deduced various mathematical equations that were used to model the organism behavior. Section 6 enumerates unique contributions of our group in the form of ideas discussed and presented in class to come up with effective strategies. Section 7 discusses limitations of our solution and possible improvements that can be made in the future work. Section 8 acknowledges ideas which were adopted from other groups or external sources. Finally, Section 9 summarizes and concludes this report.

2 Initial Insights and Observations

The initial challenge of this class was to beat RandomPlayer, an organism which moves and reproduces randomly. A number of incremental improvements occurred to us in the first round of brainstorming:

- A simple greedy strategy would consistently beat RandomPlayer. This strategy would always move onto food in adjacent squares:

```
for (i=0 to 4) {
    if foodPresent[i] {
        move in direction i;
    } else move in random direction;
}
```

- A consistent movement strategy would always consistently beat RandomPlayer. This strategy would consistently move in one direction, stopping to eat, and consistently reproduce in another direction.

```
if (foodLeft>0) stayput;
else if ((randInt from 0 to 10)==1) reproduce North;
else move west;
```

- Staying still is very cheap relative to movement. A greedy strategy like the one above the else condition resulting in stayPut rather than random movement is superior.
- Food tends to multiply faster than it appears. This means that there is an advantage on a single-player board to waiting for a number of rounds before beginning to move with one of the strategies above.

Essentially, we used these four insights to guide all of our future strategies:

1. Greedy strategies work
2. Movement should be controlled and deliberate
3. Stay still when advantage to moving is unclear
4. Allow food to accumulate to increase total board energy

3 Strategies and Concepts

3.1 Single Player

After identifying four tenets for guiding organism development, we attempted to build strategies that would account for the variable elements inherent to the Organisms board, including:

- p–probability of spontaneous food appearance
- q–probability of food doubling
- v–energy consumed by moving or reproducing
- u–energy per unit of food
- M–maximum energy per organism
- K–maximum food units per cell
- m,n–board size

The first strategy we developed aimed to qualify the third insight: stay still when advantage of moving is unclear. To do this, we planned to develop thresholds for each behavior, effectively quantifying the relative value of moving, reproducing, and staying put. A mathematical approach quickly revealed that these thresholds would be functions of p and q.

Having found this, we elected to design an organism behavior that would communally measure these variables. To do this, we developed the Talker, an organism which observed board conditions and relayed those conditions, along with an indicator of the data quality, back to the other organisms on the board.

3.1.1 Talker

The Talker was effectively a proof-of-concept–while it did successfully relay board conditions, albeit only approximately, we soon realized that a pattern-based strategy was better suited for a single-player board. The implementation started with some initial estimation of probability of food appearing and the probability of food doubling. An organism initially started in food scavenging mode. It used utility functions to estimate expected pay-off in staying at one location vs. moving and looking for food. As the time passed, the organism tended to move and look for food. When an organism discovers food, it tried to estimate the number of moves it needed to wait for the food to increase by certain amount, essentially transforming it's behavior to farming. Reproduction decisions were taken based on the amount of available food, i.e. if the organism felt that the amount of food on the board is sufficient to sustain itself and it's offspring, it would reproduce. Moreover, all the organisms preferred to move in the WEST direction, whenever they want to and preferred reproducing in the south direction. It is interesting to note that the resulting layout closely resembled "checkerboard" pattern once it converged, with organisms on all sides of cells containing food. This strategy was quite wasteful when the simulation arena had plenty of food. This realization became apparent when trying to calculate the maximum sustainable energy of the board. To calculate this, we assume that each organism is adjacent to exactly one farm. In this case, there are 355 organisms under normal conditions:

```
Total number of cells = 400
Number of Farms = 400 / 9  =  45
Number of Organisms =  400 - 45 =  355
```

If this is the case, and each organism can maintain maximum energy, the maximum sustainable board energy is 177,500

```
Total Number of Organisms * Max Energy Per Organism = 177,500
```

3.1.2 PatternMaker

Upon realizing this theoretical maximum, we began to approach the problem differently: rather than problem mimicking biological behavior, this became a packing problem, or an environment that had to be most efficiently utilized. Rather than attempting to hardcode behaviors into the organisms, we gave each one of six specific roles. These role were differentiated on two counts:

1. Each role only reproduced children of one or two other specific roles in a specific direction, and
2. Each role only harvested the farm at a specific, unique point during a 32-turn cycle.

Each organism had a cyclical concept of a turn number which was incremented every turn. This, rather than energy thresholds, determined when behaviors would happen. PatternMaker organisms on their own are very dumb, but collectively, they do an excellent job of maximising the resources on the board.

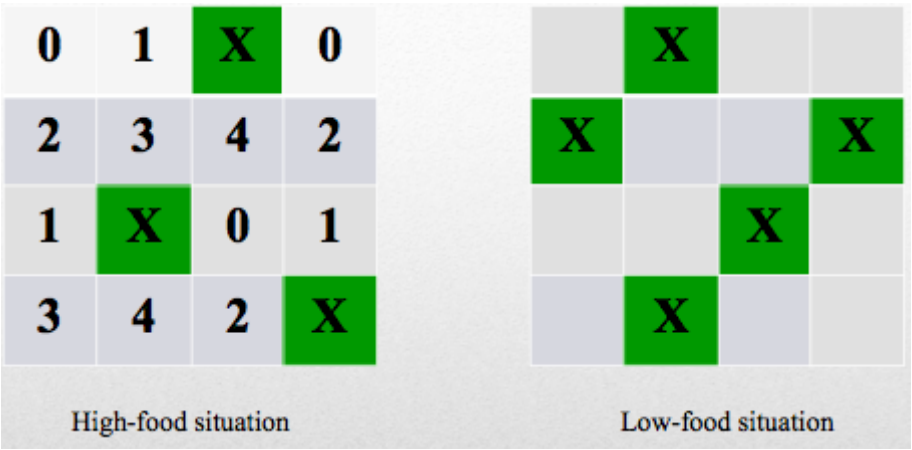

Figure 1: PatternMaker harvesting patterns in high- and low- food situations

Another PatternMaker organism was developed to deal with low-food situations. This PatternMaker relied on the same concept described above, but instead allowed each organism to be orthogonal to two farms instead of one. This resulted in separated, diagonal patterns rather than L-shaped ones (see Figure 1).

An attempt was made to reconcile the Talker and PatternMaker strategies. The goal was to identify whether the board has a high or low q value and then generate the corresponding pattern. However, this proved to be ineffective: even after identifying a data-robustness level sufficient to make a patterning decision, patterning proved ineffective because of the residual Talker organisms left on the board.

3.1.3 Surviving Mode

One thing we noticed is that for harsh environment($p = 0.1\%$), the organism will easily extinct if moving around too much, especially at the beginning of game. However, if we are able to let the food to grow for a while, the board will become abundant, which then allows the organism to make some move and maximize total power. Basically, we attempt to help our organism *Survive* the beginning couple hundred rounds(Stage One) and switch to energy miximizing mode(Stage Two). Note that the surviving mode is just one layer put at the beginning of the game. At some point(details in Implementation), when organism enters stage two, *any* algorithm for maximizing energy could be invoked. The two stages are totally decoupled.

Move or stay. During this surviving mode, apparently, reproduce is not a good action. The energy spliting makes the origanism weak and cost of stayput will doubled. The question remained to be whether the organism should just stay, waiting for food to appear in the neighbor cells or move around to explore food. We answer this question by introducing a probabilistic analysis over the benefit of moving or stay.

More specifically, we will calculate the probability of finding food when you move and compare it with the probablity of finding food if those energy are used to stayput and wait, so as to determine the best action in each round.

Eat or wait. The idea above is focusing on finding one food. But once a food is found in neighbor cell, shall the organism just go ahead and eat it or wait for as long as it can. Our answer is the later, based the following three reasons.

1. This is single player mode, no one will appear and rob the food away.
2. The goal of this stage is to survive the beginning harsh environment, waiting is more suitable to this.
3. The expectation of food on that cell grows exponential to the time you wait(more details in Implementation).

3.2 Multiple Player

3.2.1 Flood

Our multiple-player organism is named *Flood*, which comes from the flooding behavior of the strategy.

Generations. We defined three *generations* for this strategy, determined by the amount of energy left, playing different roles for the whole organism family.

Gen 0: Aggressively move, rob food and reproduce. (E.g., the very initial organism(the mother).)

Gen 1: Mildly expand, search food, reproduce if energy is high enough.

Gen 2: Stay quietly, only move if food around. If find more than one friend around, try moving away and be sparse looking.

Generation 0 is attempting to spread out as fast as possible, find more food, cover more ground. Generation 1 is in between, not so aggressive, but also not really quiet. Generation 2 will stay for most of the time, attempting to defend the zone covered and absorb resource on it.

Stay Sparse. We want to highlight our strategy for generation 2 of being spare here. We believe that we are the very first group to actually point this out. There are lots of advantages of such action.

1. Use less organism to cover more places.
2. The empty space in between can be treated as some sort of farm, making the coverage more sustainable.
3. Each organism has a chance to see more empty block around, provides it more chance to survive and hold the area.

Genetic Mutation. The key trick of this player is that, we enable mutation for the organisms based on the energy it has. For instance, if a generation 2 organism finds a cell with lots of food, then it has enough energy to be aggressive and will thus mutate to generation 0. Such strategy allows the family to form a self-balanced ecosystem. When organisms have low energy, they will stay still and try to stay as long as possible and wait for chance. When some organism has high energy, they will attempt to expand, use the extra energy to cover more ground and to suffocate the enemies.

Block Enemy. For any generation, if observe enemy in neighbor cell, it will stay still, attempt to block the action of that enemy and only move if observing food.

4 Implementation

4.1 Surviving Mode

Recall that the goal of this surviving mode is to live through the beginning stage of the game, when there are very little food on the ground and switch to the mode that maximize energy(pattern maker).

Move or stay. To determine whether it is better to stay where we are or to move, we try to calculate the payoff of each actions. More specifically, we will calculate the probability of *not* finding any food if (a) spend v energy to move or (b) say the energy to stay for $\frac{v}{s}$. Assume the organism has already wait for t time and there is still no food observed in the neighbor cells.

- If moving, in the next round, the organism will be able to see three new cells each having probability of $(1-p)^{t+1}$ of no food grown. Remember that the organism leave the original cell at time(round) t, which allows food to grow on this cell at $t+1$ round. Therefore, overall, the probability of finding no food after move is $(1-p)^{3(t+1)+1}$, i.e., probability of no food observed on the three new cells and no food grows on the original cell.
- If stay for $\frac{v}{s}$ rounds, each round, the probability of no food grows in each of the four neighbor cells is $(1-p)$. Thus the overall probability of no food observed after those rounds is $(1-p)^{\frac{4v}{s}}$.

According to the above analysis, apparently, the organism should stay at the beginning until the time $3(t+1)+1 > \frac{4v}{s}((1-p)$ is smaller than 1, thus larger index yields smaller probability). Note that an interesting property of this formula is that the analysis depends on p while the final formula itself does not, which is important in this case since p is not given. Once the organism starts moving, it won't stop and will keep following one direction, simply because make any turn will lose the chance to see an entirely new cell.

Eat or wait. Once the organism find a food nearby, it will wait for as long as it can, and then eat the food. The three reasons of this has already been discussed in previous section. Now we briefly explain why the expected amount of food is exponential to the time you wait(more specifically, $(1+q)^t$). Note that this is not concrete proof, just reasoning by intuition. First of all, the simplest case, when there is one food, in the next round, the expectation is $q \times 2 + (1-q) \times 1 = 1+q$. Using induction, at time t, assume the expectation of food is $(1+q)^t$. The expectation could be written as $\sum_i p_i \times i$ where p_i is the probability of there being i food on cell. For i food, each of them has expectation to $1+q$, with summing up being $i \times (1+q)$. The expectation on $\sum_i p_i \times i$ now becomes $\sum_i p_i \times i \times (1+q)$. Therefore, after time $t+1$, the expectation of food is $(1+q)^{t+1}$.

Switch mode. In the current implementation, when $energyleft + foodleftxu > 2M$. In other word, if there is enough energy for both this organism and the child it (could) reproduces.

4.2 Flood

Recall that flood is our multi-player strategy that having three generations, which can be mutated from one to another, allowing the whole family to switch between being aggressively or being quietly depending on different scenarios they encounter.

Reproduction. To cover the board as fast as possible, when an organism reproduce, we use the strategy of one north(east) and one south(west). Remember that the newly reproduced organism will following the same rule, which is essentially how we achieve the goal of exploring all directions simultaneously. The condition for an organism to reproduce is simply a check on its energyleft. If having more than 40% of M(which turns out to be not robust when M is set low, more details in Analysis section), the organism will reproduce to the direction following the rule we just described.

Genetic Mutation. The mutation is achieved by checking the energyleft of the organism. In the current implementation, it is Gen 0: $> 80\%M$, Gen 1: $60\%M - 80\%M$, Gen 2: $< 20\%M$. Such implementation turns out to be both clean and efficient.

Detect Enemy. We use same external state for all of our organisms which allows us to identify the organism appears in the neighbor cell is a friend or not. The value of such state is picked arbitrary and currently hard coded.

Keep Spare. Based on the enemy detection method above, the organism is now able to check number of friends around. If there are more than one friends, then it will try to move and stay away from each other. In addition, the organism always remember the last direction it moves so as to avoid stepping back and forth and waste energy.

4.3 Other Implementation Highlights

Directions. Instead of directly store and use the direction constant given by the simulator, we have a private lib which is essentially built on an array representation of the four directions. The most important advantage is that we can use for loop when we need to check all directions instead of having a switch and handle each direction individually. This makes our code much more compact and make it easy to have util-functions like check if there is food(enemy) around, left, right, reverse of a given direction, etc.

5 Analysis and Results

5.1 Single Player

While Talker did effectively communicate board conditions, it was never tested extensively. Instead, we focused singleplayer testing on PatternMaker. Some of the results are shown in the table below:

# of Trials	# of Rounds	Conditions	Average End Energy	Average End Organism Count
5	500,000	Standard (20x20, p=.01, q=.02)	170,031 (95.79% of theoretical max)	381
5	100,000	Large (50x50, p=.01, q=.02)	1,027,110 (92.4% of theoretical max)	2330

Testing showed that the algorithm is still partially flawed: while the theoretical maximum is unachievable in practice because of conflicting patterns due to the boards overlapping boundaries, it is clear that the current algorithm has a flawed reproduction function: only 355 organisms should be spawned on a 20x20 board; the observed average organism count was 381.

It should also be noted that the average energy per organism on the small board was 446.26. Given an ideal patterning algorithm with a thirty-two-round movement cycle, each organism should have an theoretical average energy of 475; this threshold was also not sufficiently met. Further optimisation may make approaching this threshold possible, although the randomness associated with food generation would make meeting it completely also impossible.

The larger board performed slightly more poorly, reaching only 92.4% board saturation after 100,000 rounds. However, the total energy of the board was still observed to be growing after 100,000 rounds: the asymptote of the growth function likely approaches the percentage of total potential energy observed on the smaller board.

5.2 Multi-player

Note that our submitted multi-player strategy is focused specifically on the default setup. A few reasons are listed below.

1. We believe there is no single strategy that can outperform others in both single player and multi-player case. The tournament result strongly supported this. The player perform well in single mode are easily died in multi-player case.

2. We do not want to design an algorithm that performs ok in all set up but not be able to win any of them.

3. We are really interested in multi-player mode. As mentioned in the previous section, in fact, our single player strategy can outperform the results in tournament, both abundant resource and spareness case, in terms of average sustainable energy. We decide to submit multi-player strategy simply because it is more fun to watch our organism flooding others.

Quote from the tournament result:
Multi-player, [X=20, Y=20, p=0.01, q=0.02, maxrounds=50000]:

	Average End Energy	Average Extinction
Group1PlayerNew	0	(2,755)
Group2Player	0	(4,498)
ShyPlayer	0	(3,047)
RainbowCastle	29,651	(2,486)
Group5Flood	19,362	(6,258)
Group6Player	0	(1,612)

The only enemy that survives with Flood is *RainbowCastle* from group 4. We have lower average energy than them, but our average Extinction is much longer than they have, which means if they die, they will die really early in the game. Here is a detailed look of the ten trails stats of the two group:

RainbowCastle	27,989	(2,362)	(2,798)	29,812	28,184
	(1,069)	(2,182)	32,620	(2,218)	(4,284)
Group5Flood	(13,959)	16,931	18,955	(3,054)	(3,808)
	17,675	20,226	(1,156)	23,187	19,196

As one can see, RainbowCastle extinct for six times which we only extinct for four, which make us question the meaningfulness of the given stats. Essentially, if extinct, the end energy is zero. If not extinct, the Extinction time will equals the maxrounds (assume world is doomed at the last round and all organisms are slaughtered). We recalculate the final stats and get the following:

	Average End Energy	Average Extinction
RainbowCastle	11860	(21491)
Group5Flood	11617	(32503)

Now, one can see that the average end energy are almost the same but our average extinction is much later. The reason for this is that we are not performing any energy maximization in the multi-player case. As discussed before, the ecosystem will always force the family to expand if having extra energy. If making an analogy to boxing, our player is always aiming at knocking out instead of earning more technical score. Nevertheless, extinct two times less out of ten trail might not be sufficient to demonstrate the power of our player. We are looking forward to do a bit more test if having their class file.

Multi-player, [X=20, Y=20, p=0.01, q=0.02, maxrounds=50000]:

	Average End Energy	Average Extinction
Group1PlayerNew	0	(3,493)
Group2Player	0	(9,740)
ShyPlayer	0	(4,190)
RainbowCastle	0	(13,715)
Group5Flood	119,438	N/A
Group6Player	0	(5,283)

We overwhelmingly win the game under this setup. One of the reason might be the expanding nature of our ecosystem. The larger the board, the more space the whole family have to flood.

Additional setup: 1 Flood vs 4 sheep, [X=20, Y=20, p=0.01, q=0.02, maxrounds=20000]

```
[Player Configuration]:
[Player0]: Flood
[Player1]: Summer 2012 Player
[Player2]: Summer 2012 Player
[Player3]: Summer 2012 Player
[Player4]: Summer 2012 Player
[SUMMARY of 10 trials]
Player: 0 Mean Count: 110 Mean Energy: 13163 Extinction: 3
Player: 1 Mean Count: 18 Mean Energy: 3230 Extinction: 9
Player: 2 Mean Count: 12 Mean Energy: 2470 Extinction: 9
Player: 3 Mean Count: 0 Mean Energy: 0 Extinction: 10
Player: 4 Mean Count: 24 Mean Energy: 4265 Extinction: 8
```

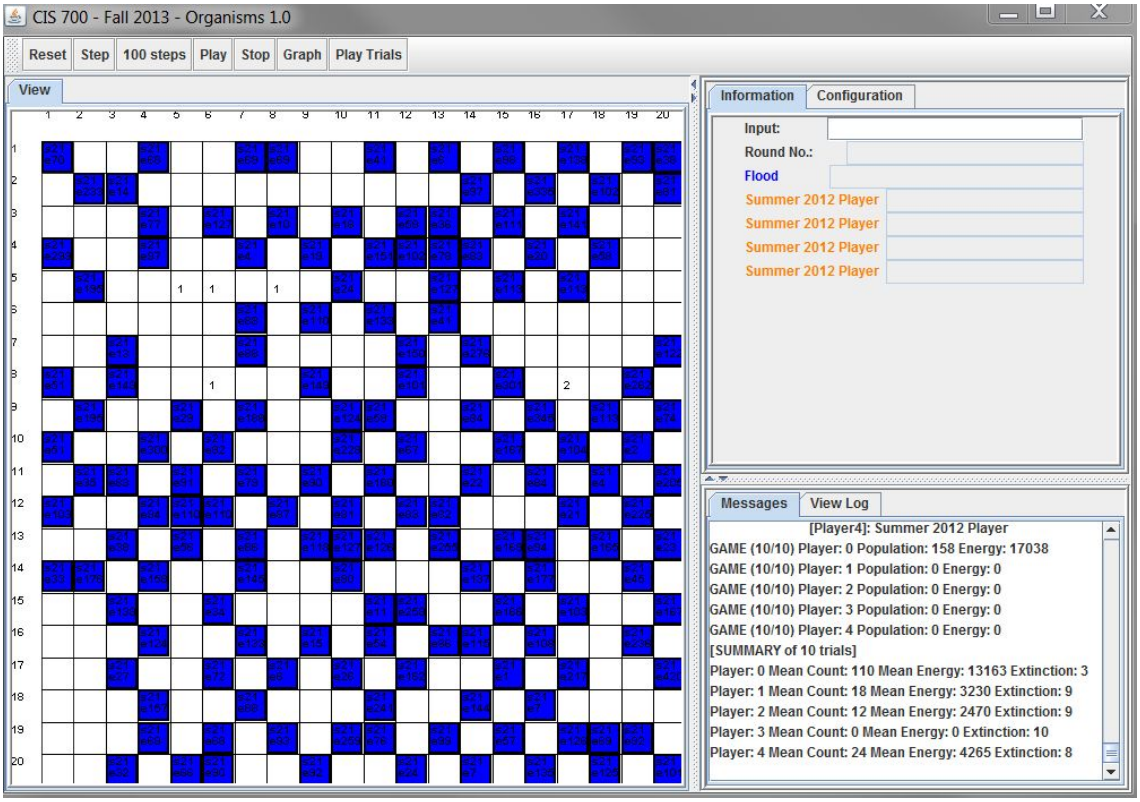

Figure 2: Screen shot of 1 food vs 4 sheep

One can see that we still win(extinct all sheep) 70% of time. Note that sheep is not communication based strategy, thus putting multiple sheep on the board will only increase the initial energy the expanding rate of sheep instead of having two sheep realm and fighting with each other.

6 Contributions

We list a few of our major contributions in the following.

- Analysis and calculate the maximal possible sustainable energy that could be achieved. We use this analysis as a guidance when designing the PatternMaker single player strategy, i.e., the final goal of this strategy is to meet the theoretical energy maximum.

- Design the communication protocol and mechanism between organism, experiment and conclude the limitation and inefficiency of communication based strategy.
- Introduce the surviving stage, use probability analysis to evaluation the immediate best move for each round in order to survive the harsh condition in the beginning.
- Discover the idea of allowing organism to switch role to either direction(aggressive or quiet) so as to achieve an ecosystem-like strategy. The resulted circulating virtual behavior successfully maintain the overall healthiness of the whole organism family. Allow the family the expand when having enough energy and being conservative under harsh condition.
- Bring up the idea of make organism spare to each other instead of staying crowdedly together, which enable a much larger coverage of board with much less number of organism, which can sustain a much longer time.

7 Future Direction and Limitations

The most obvious take away from the results displayed above is the weakness of Flood in singleplayer and in low-energy environments. This is mainly because of Flood's orientation: Flood was designed as an aggressive, field-capturing organism and optimised for multi-player fields with moderate to high resource levels.

Strategies to adjust Flood to each of these scenarios are described in the following sections.

7.1 Single-Player

As a highly-aggressive quick-board-covering strategy is necessary for multi-player scenarios, this would need to be Flood's default behavior. After several rounds, it could identify whether or not the board truly is multi-player and adjust the strategy accordingly.

One way to do this would be to use the state to keep track of the number of turns passed and the whether or not enemies had been detected:

```
//if state = 0, enemy has been detected
for (i=0 to 4) {
if (neighbors(i) != -1 || neighbors(i) = 0) state = 0;
} if (state != 0) state++;
```

Using this algorithm, any organism that encounters either an enemy or a friendly organism that has encountered an enemy has a record of that interaction.

One method to adopt the PatternMaker strategy would be to allow a random genetic mutation if enough turns have passed and no enemy is encountered; that is, if state reaches a given threshold. At this point, a genetic mutation could be allowed, in that it affects a very small, random sample of organisms, causing them to adopt PatternMaker behavior. Should any non-PatternMaker organisms encounter a Patternmaker, they should avoid the PatternMaker organism and pursue self-destructive behavior (ie, avoid food and move often) to try to clear the board for the PatternMakers entrance.

Of course, this strategy does rely on a random element, but given a sufficiently long game, it should serve as an effective transition to a single-player strategy.

7.2 Low Resources

As Flood is a highly aggressive organism, it moves far and often, especially early in the game. In low-energy situations, this is very undesirable and leads to extinction.

While prefixing Flood with Talker-type scouts would identify low-energy conditions, this would also adversely affect Flood's multi-player performance, as early board domination is a key part of its strategy.

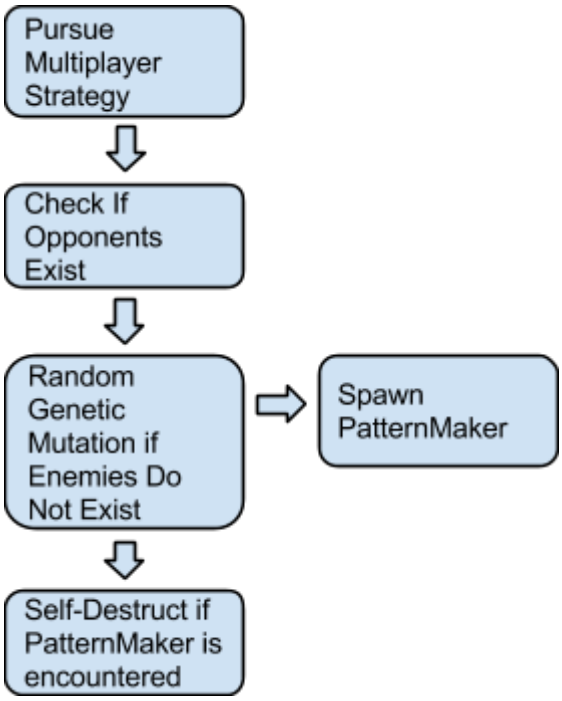

Figure 3: Flowchart of potential future design

One way to handle low resources could be an envolution of PatternMaker which give it the ability to make different patterns based on the environmental conditions. Since we have the surviving mode completely isolated from the energy maximization algorithm, the future work only need to focus on establishing a better pattern for achieving a high sustainable total energy.

8 Acknowledgments

9 Conclusion

Our ultimate findings can be summarized in two statements:

1. Multiplayer strategies tend to perform better when they are initially-aggressive ultimately-conservative behavior-based organisms, and
2. Single-player boards are best played using a structural approach which measures and adjusts according to board variables.

We found it difficult to combine the two strategies into an organism which detects board conditions and responds accordingly; however, we do believe that such a strategy is ultimately possible. Nonetheless, it seems that the optimal single-player and multiplayer strategies are orthogonal to each other: without sophisticated detection and communication algorithms, an organism must choose to specialize to either a singleplayer or multiplayer role.

While we were not successful in combining the strategies, we were generally pleased with the success of our individual strategies. The single-player oriented PatternMaker achieved ([INSERT PERCENTAGE HERE]) of the maximal potential board energy; meanwhile, the multi-player oriented Flood was able to outperform other organisms in its designed setup(win overwhelmingly on 50×50 board). Moreover, it was able to frequently compete and succeed against multiple copies of the SurviveSheep organism simultaneously.

Throughout the project, we found that our strategies did change considerably, greatly aided by new perspectives gained through class discussions. We found that frequently reconsidering our assumptions and approaching the problem from totally new perspectives was enormously helpful in moving forward. Contrary to class sentiment, we felt that overarching strategy, not parameters, still accounted for the primary differences between organisms.