CIS 559 PROJECT 3: GUNSLINGER

IAN SIBNER, DAVID LIAO, NIKHIL ROY

20 October 2015

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Gunslinger is an n-player game where each player is initialize with two disjoint sets of f friendly players (*friends*) and e enemy players (*enemies*). This implies that $e + f < n$ since the player will not have itself as a friend or enemy. The friend relationship is symmetric, so if A is a friend of B, then B is a friend of A; the enemy relationship, however, may be asymmetric.

Each turn, each player can fire one shot in an attempt to kill another player. However, a player is not killed unless two players shoot it simultaneously. Once dead, a player can no longer shoot. The game ends when the set of living players remains unchanged for 10 rounds in a row, whether or not players are still shooting. Each players' objective is have the highest number of points at the end of the round. Points are awarded as follows:

1. One point for surviving until the end of the round.
2. One point for each surviving friend (maximum f).
3. One point for each dead enemy (maximum e).

Players retain perfect information about who shot whom in previous rounds.

2 INITIAL INSIGHTS AND OBSERVATIONS

Immediately, our team observed several key features about the game.

1. While players are given a list of allies and enemies, there are more opponents than just those given. Players who we perceive neutrally could target us as an enemy. This lead us to believe that we might consider targeting those players as well to preserve our wellbeing.
2. A way to model the state of the game as perceived by our player would be to create a dynamic enmity graph. This would help us keep track of the relationships between all players

3. We needed to consider whether or not an aggressive or non-aggressive approach would be better. As a proactive player, targeting our enemies might lead others to target us (if we target their allies and they choose to defend them). On the other hand, taking a retaliatory approach might limit our influence on the game end state.
4. Winning the game meant having the highest score. By killing off other players, we would be able to raise our own rank by lowering other players' score.

These insights guided our strategy throughout and allowed us to focus on the most important aspects of the game.

3 STRATEGIES AND CONCEPTS

3.1 David's Revenge

David's Revenge was our initial attempt where we tried to create a retaliatory player. The main idea was to maintain our own player's wellbeing while minimizing involvement if at all possible. The only times that David's Revenge would take action is if an arbitrary player targeted our player.

We initially construct an enmity graph. An enmity graph captures which players seem to be enemies of others. This provided us with more than enough information to design our player's decision making. We only consider any player that targeted us.

Using our enmity graph, we identify players that have targeted us and choose a player to return fire. The choice in player evolved through many designs. Initially it was the most recent player that shot us, but later on it became a weighted selection based on the number of times the player targeted us. We used a bounding "window" to make more recent hits more relevant as opposed to more temporally distant hits. If we had no player target us, we would simply take no action.

3.2 Protective Farmer

The Protective Farmer was a reaction to the results of the preliminary test run during the first several classes. XOrganism was not faring so well later on in games with other, more aggressive, players, even when it did well in the early stages of the game. As well, due to XOrganism's courteous moves, it would lose valuable coverage of the food-populated cells on the board.

The Protective Farmer was an experiment to see if these issues could be mitigated by a completely different strategy. The core idea behind the Farmer was that if there were four organisms on each adjacent cell to a cell with food, they could all 'farm' this cell. Staying off of the cell would allow it to duplicate freely, while the adjacency pattern would prevent opposing players from reaching the food source.

The initial stage of the Protective Farmer, before farming a food source, is to find a source of food that could support this configuration. In games with more organisms, many of the cells were covered up quickly, or enemy organisms prevented the Farmer from reaching other cells. The solution for this problem in Protective Farmer was almost the exact opposite of the movement strategy in XOrganism. The Farmer would move onto a new, unexplored cell on every turn (unless it needed food). It's goal was to find a cell with greater than four units of food on it as the Farmer may have needed to eat and will have to reproduce three times all without eating up the remainder of the food.

Once it found a satisfactory source food cell it transitioned into a building phase. The Farmer would look at all adjacent tiles and see if any were filled. If more than two were empty, then the farmer reproduces a wall onto one of those cells. This continues until there is only one cell left uncovered, which the Farmer simply moves to (instead of reproducing on to) to become the final wall. This Farmer organism and the newly spawned Farmer players are set to a Protective mode. This means that the organisms will stay still on their own cell unless they need food. If they do, they will attempt to move inwards. Once they replenish their energy, they move back onto the open cell they left earlier. In addition, when they move into the center cell, they also transition back to the building phase to fill any gaps in the wall that may have formed if any other Farmer organisms died off.

3.3 David's Revenge's Revenge

As the name suggests, David's Revenge's Revenge (DRR for short) was built directly on top of David's Revenge. Determining who to shoot is the exact same as David's Revenge: when DRR shoots, it will shoot the enemy who shot it most in previous rounds. However, unlike David's Revenge, DRR does not shoot whenever it can. It uses a logistic function in order to determine a *shooting percentage*, and then shoots with a probability equal to this function value.

The intuition behind the logistic function is that in situations with lots of friends, it is better to be conservative with shooting. The friends of the enemy you shot are likely to retaliate in subsequent rounds, so a player that is too aggressive in these situations is likely to die. However, in situations with few friends, being aggressive makes more sense because retaliation is unlikely. This applies regardless of the number of actual enemies; it generally pays off to target neutral players as well, as long as there are few friends, because it improves a players' *relative* ranking compared to other players. It was noted by Spriha that this might bring down a players' ranking in certain situations because it would eliminate another player's enemy; however, as we will see in the results section, we actually did quite well in situations where there were many enemies.

Thus we used the ratio of non-friends to players as the independent variable in the logistic function: $x = \frac{n-f}{n}$. Then the shooting percentage is calculated as $P = \frac{1}{1+e^{-\frac{x-\mu}{\beta}}}$, where β and μ govern the steepness and the mean of the distribution respectively. Initially, our logistic curve had $\beta = 0.07, \mu = 0.07$ (see Figure 1). However, we wanted to find the best-performing combination, so we systematically tried different combinations of μ and β to find out which produced the best rank. In the end, we found that $\beta = 0.06, \mu = 0.4$ (see figure 2) was the best performing. This curve was more aggressive than the original curve;

it appears that shooting more often improves performance even at relatively low non-friend to players ratio, reaching 90% at an x-value of just 60%.

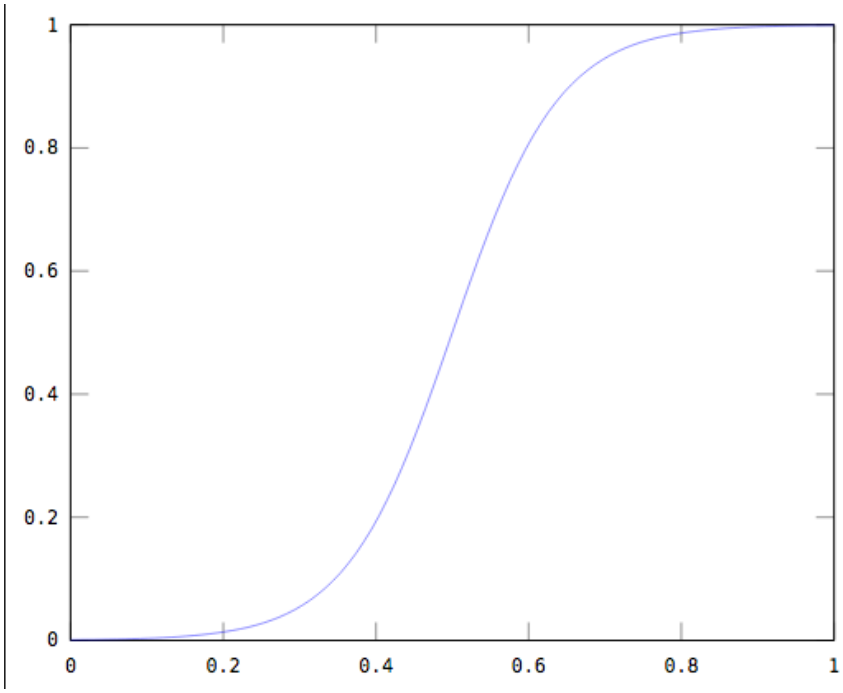

Figure 1: Initial Logistic Curve with $\beta = 0.07, \mu = 0.5$

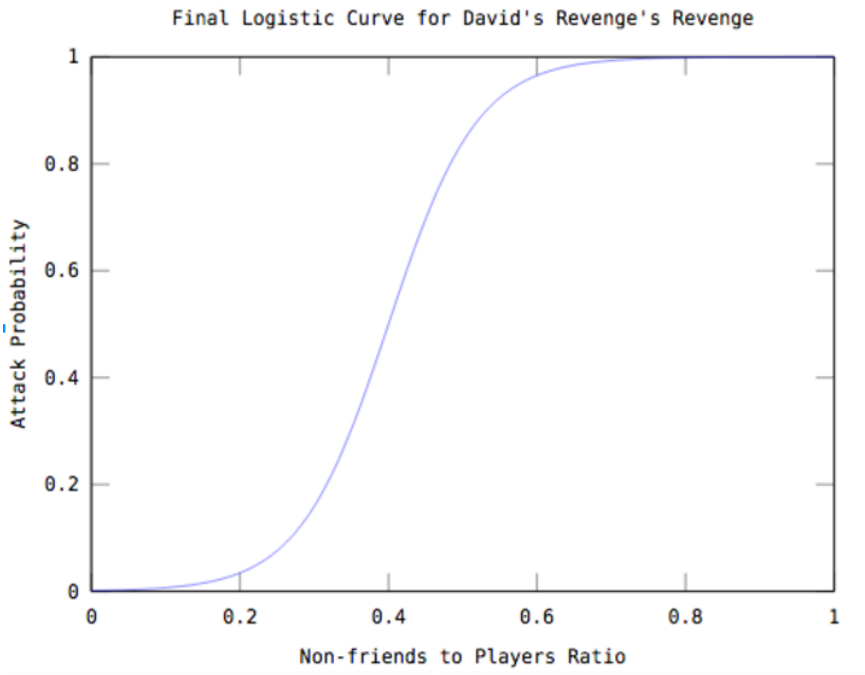

Figure 2: Final logistic curve with $\beta = 0.06, \mu = 0.4$

4 IMPLEMENTATION

4.1 XOrganism

The implementation of XOrganism was fairly straightforward, as it only considered its current energy level and known parameter values. The only bit of state that was carried over (in addition to game state, such as current energy level) was the number of turns it had been alive (which was used in order to move every 20th turn to seek out more food). We first defined an `isSated` function (which just determined whether $\text{Energy}_{\text{organism}} \geqslant M - u$). Then, every turn, we did the following:

1. If the organism was sated, we would reproduce with a probability of 70%.
2. If the organism did not reproduce, but still had food on its square and was sated, then it would move into an empty adjacent square - a courteous move designed to help other XOrganisms who might need food.
3. If there was no food in its square, and the organism was not sated, then it would attempt to move into an adjacent square with food. Usually, if there was no food available, the organism would stay put. But every 20th turn (determined by the age counter), it would make a random move in order to seek out pockets of food that might be just beyond its field of vision.

This implementation was very simple - only 100 lines of Java code. This is notable because this organism did tend to survive on its own, showing that even very simple heuristics can lead to acceptable results.

4.2 Protective Farmer

The implementation of Protective Farmer was divided up into several different phases. The Farmer uses external state to keep track of which phase of its lifecycle it's in, but this information is only used between players during the building phase.

The first phase was classified as the SEARCHING phase. Only the first organism would start out with this state. If the cell that the Farmer was on had no units of food, it would attempt to move in an unvisited direction with food. If there was no unvisited direction with food, then it would be an unvisited direction. If there was no remaining unvisited direction, then the Farmer would attempt to move in any open direction. These choices were calculated by filtering down the set of possible choices (from open directions to unvisited directions to unvisited directions with food) and then randomly picking a remaining option. The Farmer's visited steps were calculated by keeping track of its position relative to its point of birth. Once the Farmer takes a step, its previous position is added to a set of Points (later used in determining if the Farmer is attempting to visit a cell that has already been visited). One potential drawback to this strategy is that the Farmer won't be able to figure out the global coordinates of its position and therefore could be revisiting certain cells if it wraps around an edge during its life. As well, due to the random nature of direction selection, it is possible that the Farmer could trap itself inside its own path. In this case, the Farmer will look at only open directions as it will eventually have no other options. The Farmer will only stop if is hungry and needs food, or it find an adequate food source to start farming.

Once the Farmer finds a satisfactory (food units are greater than four) food source, it transitions into the CENTER phase. In this phase, the farmer looks at the adjacent tiles and looks to see if they are filled with other Farmers in the WALL phase. For this implementation, the WALL external state is an integer from three to six, so it is possible a malicious imposter could break this strategy. If more than two of these directions are empty, the Farmer reproduces in that direction, passing along this direction information to the new Farmer organism. There are four WALL states: WEST_WALL(3), EAST_WALL(4), NORTH_WALL(5), and SOUTH_WALL(6). If there is exactly one direction that is not protected by a wall, then the Farmer moves in that direction, becoming that wall.

All of the walls have the same logic for their phase. If they reach a minimum energy threshold they will attempt to move back onto the food source (transitioning into the CENTER phase as well). Otherwise, they will remain motionless. The minimum threshold is the movement cost v plus nine times the stay cost (1). The absolute minimum survivable energy level is one more than the movement cost, because even if the organism can move to get food, it will not survive the move. The only other action a Farmer in the wall phase can take is reproduce (with some probability, just like XOrganism) away from the food source if the cell it is originally spawned on has food.

4.3 SimplePlayer

The implementation of Simple Player was true to its name, simple. It relied on the basic input parameters to the game and simple strategic calculations. Most of the behavior was identical to the XORganism, with only a few changes and improvements (which are listed below).

- **Reproduction:** The more aggressive reproduction strategy was implemented from two angles.
  - We increased the reproductive probability to 90% after many trials of different values with other organisms in various configurations. Any higher, and the organism tended to overextend and die off; lower, and the organism would not spread fast enough to be competitive even on large boards.
  - Simultaneously, we decreased the energy required to reproduce to $\text{Energy}_{organism} \geqslant M - 1.5u$ - i.e., we changed the definition of "sated" to require slightly less energy. This tended to make the organism much less conservative about reproduction, while still ensuring offspring had sufficient energy. The value of 1.5 was also determined to be an effective value through many trialss.

  Therefore, on a given turn in the energy is greater than the new energy threshold, there is a 90% chance of reproduction.
- **Movement:** The organisms performs the following checks for movement on a given turn.
  1. If there is food on the square of the organism, then the organism stays put and eats.
  2. If there is food on at least one of the squares, then it will move to a random surrounding square containing food.
  3. If there is not food on any of the surrounding squares, then the organism, provided it has enough energy, will move to a random square. If the north square is available, then the organism will take it with probability 75%.

5 RESULTS

This project has many more configuration variables compared to the previous project (eight rather than two). However, the players were not generally aware of most of these changing parameters. As such, players were designed to function reasonably well in all cases, especially those where hidden parameters were set to extremes. The results from the several tournament runs are essentially a sampling of different viable configurations for this game. As well, there are two completely different tournament scenarios to consider. For these results, only the scenarios where the players were by themselves or in a group of the best six will be considered.

These tables include t-scores for SimplePlayer compared to all of the players involved. Significance, as used below, refers to significance with a confidence level of 95% (against a t-score of 2.571).

Liechtenstein	The Medium Desert	The Easier Desert	Goldmines	The Dark Age	Default	The Rainforest	Grasslands
1.77187327	5	4.859805612	-1.280325073	2.191041864	1.803715007	1.462463905	1.628158065
3.738631686	4.980683808	4.938235181	-1.212168282	2.057820425	3.864156772	2.638700964	3.104471845
3.958924646	4.001789422	2.46348375	-1.553470225	0.8753761549	4.066190844	2.552269374	2.94376406

Hot Rainforest in Chile	Hot Rainforest in Tennessee	Himalayas	Atlantis	Utopian Future	Hot Rainforest	Post-Nuclear War	Americas
1.644630091	1.658363301	1.71323914	1.845748438	1.58113883	1.799297461	1.996965643	3.830720234
2.876045405	2.825060209	2.530999457	3.16849614	2.439155207	2.862324889	2.975209734	3.738504112
1.304317793	0.8782210148	0.1652104842	3.149982142	2.439560662	1.523118536	1.165165693	2.474944379

Mean Count: Maximum, Significant

Mean Energy: Maximum, Significant

Mean Energy Density: Maximum, Significant

Figure 3: t-score significance of count, energy, and energy density

We firstly consider the results from the single player tournaments (Figure 1), where each tournament was run with ony one player. From the results, it appears that SimplePlayer did exceptionally well when considering the total energy held on the board. SimplePlayer had a significantly greater amount of energy than any of the other five players in thirteen out of sixteen configurations. For many of these configurations, SimplePlayer also has a significantly greater or greatest energy density (Energy / Count). These successes are most likely due to a movement strategy that favors staying on, moving to, and reproducing to cells with food.

The one configuration from the single player tournaments that produced a remarkably poor performance from the SimplePlayer is Goldmines. This is due to the abnormally low value of p, which makes it very unlikely that a player standing still will ever see food before starving to death. Since it was more common that the SimplePlayer never had food spawn next to it, it most likely went extinct. Normally, the SimplePlayer is naturally optimized for low-p conditions (seeking out food as a movement priority), but Goldmines is just too sparse of a configuration. If p was slightly higher, SimplePlayer may have performed as well as it did in both The Medium Desert and The Easier Desert. Once SimplePlayer successfully managed the starting phase of a low-p configuration, it ended up doing the best out of all six players included.

Liechtenstein	The Medium Desert	The Easier Desert	Goldmines	The Dark Age	Default	The Rainforest	Grasslands
-0.833097228	5	4.898979486	-1.221524007	-0.215490662	-1.029254109	-1	-1
-0.546435132	4.982852193	4.952994079	-1.191130259	-0.109784464	-1.015408144	-1	-1
3.045482801	4.391550328	3.501987019	-1.573609786	2.307380857	0.3237892037	-1	-1

Hot Rainforest in Chile	Hot Rainforest in Tennessee	Himalayas	Atlantis	Utopian Future	Hot Rainforest	Post-Nuclear War	Americas
1.541695575	-0.273482958	4.624170746	-1	3.567626967	1.380482076	-1.923870778	1.926105341
1.970444788	-0.008302234	4.7539258	-1	3.746487198	1.732445194	-1.702569698	3.583279511
1.828964919	2.90242565	3.642166645	-1	3.365081171	2.343148403	1.832980601	4.056195168

Mean Count: Maximum, Significant

Mean Energy: Maximum, Significant

Mean Energy Density: Maximum, Significant

Figure 4: t-score significance of count, energy, and energy density

The results from the best six player tournaments (Figure 2) are very different from the single player tournaments. SimplePlayer only did well in several configurations: The Medium Desert, The Easier Desert, Himalayas, Utopian Future, and Americas. The reasoning for the successful performance of SimplePlayer has already been explained above. However, the remaining two configurations are very strange. SimplePlayer performed moderately well on Hot Rainforest, but for Utopian Future it was significantly greater than any other player. The main differences are an increase in K and u and a decrease in v. The addition of such mobility, combined with the incredible reward for finding food allowed SimplePlayer to function the best. Himalayas is a larger than default board, with a higher moving cost v. One reason that SimplePlayer performed so well is that the movement logic is set to only move when food is available, it is not wasteful with movement. This strategy allowed SimplePlayer to perform significantly greater than every other player on both The Medium Desert and The Easier Desert, as well as the Himalayas.

For the best six player configurations, there were noticeable declines in performance from the single player runs. Many of the configurations ended up being monopolized by one player (normally Group 4's FarmingPlayer), leaving very poor results for the SimplePlayer. However, despite the fact that many organism totals and organism energy levels were lower than the average produced by other organisms, the energy density was still fairly high (significant in some cases like Liechtenstein).

6 CONTRIBUTIONS

We had two main contributions to the class strategy and discussion:

1. **Importance of initial expansion -** We were one of the initial teams to implement a strategy, XOrganism, that demonstrated the lasting benefits of rapid initial yet sustainable reproduction. While were not the only team to arrive at this conclusion, it proved very important to many teams' strategies throughout the class.
2. **Northward tendency -** Our team was the first team to recognize the downward trend in organism movement and implement a strategy to alter our movement with a directional tendency to increase longevity of organism survival.

7 FUTURE DIRECTIONS AND LIMITATIONS

7.1 Faster Initial Spread

Our players tended to do best on larger boards because of its high energy density. It tended not to overextend, unlike some other organisms which would tend to start dying frequently due to very aggressive population growth. But on small maps, this aggressive overextension was actually beneficial to these organisms, because they could "choke out" less aggressive competitors and dominate all food resources. Once their opponents went extinct, their organisms would settle into a steady state of lower population and energy compared to their initial population explosion. However, by that time, it was too late for slower-growing competitors like SimplePlayer.

Although an organism might not know the size of the board it starts on, we believe that our SimplePlayer could benefit from employing a more aggressive strategy *only during the first part of the game*. The ideal length of this "aggressive period" might vary based on known map conditions, but based on our observations of population explosions in configurations where the total grid size was less than 1000, we believe that around 100 turns of aggressive expansion is usually sufficient to gain dominance over a large portion of the map. Once this period is up, we would switch to our current, more conservative strategy to take advantage of its natural tendency towards higher energy density. This would make our SimplePlayer much harder to choke out at the beginning, but would still allow it to take advantage of its effective conservative strategy after the intial territory grab. Following the initial population explosion, organisms that maintained aggression would tend to move more and use more energy, which would allow our less aggressive organisms to slowly chip away at their territory while maintaining a high average energy level.

8 ACKNOWLEDGMENTS

The evolution of our players was heavily based on class discussion and iteration of our original ideas based on in-class runs. Group 2 was particularly helpful, due to its "Rabbits" strategy (later renamed to "Harvey"). Their fast expansion - and the way they efficiently choked out our XOrganism and our first draft of SinglePlayer - convinced us of the need to be less cautious about our reproduction chances if we wanted to compete in high-food environments.

9 CONCLUSION

Overall we were happy with the progression of this project, particularly our SimplePlayer's performance on large boards or boards with high movement costs. We ended up with the highest population and overall energy in several of the tournament configurations, showing that good results could be achieved even with comparatively simple heuristics.