CIS 559 PROJECT 2: PARALLEL FOOTBALL

IAN SIBNER, MICHAEL MOLISANI, AND LAURA KINGSLEY

20 October 2015

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

The Organisms game consists of a world (an $m \times n$) grid populated by one or more organisms. Each organism has a *brain* (its controlling program) and an *energy level*. An organism expends energy each turn - 1 unit to stay in the same place, and some variable amount of energy v to move to one of its four unoccupied neighbor squares (up, right, down, and left). Moving off the board is allowed; the world wraps around and the organism simply appears on the opposite side. An organism gains energy by eating food, which has some probability of spontaneously appearing on an empty cell. Multiple units of food can occupy a given cell, up to some threshold, and each turn a unit of food in a cell which is not occupied by an organism has a chance of doubling (so a cell with three units of food may have anywhere from 3 to 6 units the next turn depending on randomness). An organism can also *reproduce* onto an adjacent empty square at the cost of v energy. Each unit will have half of the organism's remaining energy after paying the reproduction cost.

The parameters of the simulation are listed below:

Parameter	**Meaning**
m	Horizontal board size (X-direction)
n	Vertical board size (Y-direction)
p	Chance of food spontaneously appearing in an unoccupied space
q	Chance that a given unit of food in an unoccupied space will double
v	Energy consumed by moving or reproducing
u	Energy gained by an organism for consumption of one unit of food
M	Maximum energy level for an organism
K	Maximum number of food units per cell

Additionally, an organism has a *state* (an integer between 0 and 255) which it can set each turn it acts. An organism can only "see" in four directions (up, right, down, and left). It can sense the amount of food in its current cell, the presence of food in the four neighboring cells (but not the amount), and the presence and state of any organism in the neighboring cells. Organisms are aware of only a limited subset of parameter values (v, u, M, and K) but not the others.

Each turn, an organism can change its state, stay put, move, or reproduce. Moves are calculated row-by-row, from top to bottom; this means that an organism at (1,1) will move before an organism at (2,1). If a move is not valid (possibly because another organism moved into a previously unoccupied space before another could act), then it is treated as a stay-put move.

2 INITIAL INSIGHTS AND OBSERVATIONS

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

1. The information available to organisms would be very limited. They can only 'see' in four directions, and could only detect food (or lack of food) and other organisms' states. So strategies that required non-local knowledge or multi-organism coordination would be difficult to implement.
2. Staying put is much cheaper than moving since $v \geqslant 1$. Also, for the majority of the setups we considered, we had $v \geqslant 20$. Therefore, an organism should stay put more often than it moves.
3. For all setups, we had $q \geqslant p$. This means that food tended to 'collect' in certain squares, rather than being widely distributed on empty squares.
4. A player with fewer individual organisms tends to die out faster, due to the random nature of food spawning. So it pays to try and reproduce.
5. Reproducing produces two organisms with half the energy of the original, so the more energy an organism has when it reproduces, the more likely both offspring will survive.

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 XOrganism

XOrganism was our first attempt, so we tried to create an organism that would survive on its own in a variety of conditions. The goal was to have a good survival rate in "desert" conditions, where $p \leqslant 0.005$. We did this in several ways.

First, we defined the idea of an organism being 'sated'. In effect, a sated organism is close enough to its maximum energy that it is not in immediate need of food. There are many possible definitions for satiety; we initially defined an organism as sated if $Energy_{organism} \geqslant M - u$. This was a fairly intuitive definition: an organism is sated if eating one more unit of food would fill it up completely.

For movement, we tried to stay put very often because it only cost 1 unit of energy. We only moved under a few conditions:

1. An adjacent square has food, and the organism is not 'sated'. The rationale behind this movement pattern is fairly obvious - organisms tend to survive longer if they seek out food.

2. The organism is 'sated', but its square still has food. The idea here was to be *courteous*, and allow other organisms who might need that food to jump on it.
3. The organism has not found food in 20 turns. In this case, it moves in a random direction to try to look for food it might have missed.

For reproduction, our organism was similarly simple. If the player is sated, then there is a 70% chance of reproducing in a random direction. This ensures that when an organism does reproduce, both offspring have about half of their maximum energy, giving them a better chance of survival.

This organism tended to do pretty well on its own, surviving in about 30% of our test runs with $p = 0.005$. However, it performed significantly worse than other players in configurations where food was readily available. Worse, it tended to get "choked out" by more aggressive players, who would often take advantage of its courteous moves to take the food for themselves and reproduce. So, while it was a fairly good start, we needed to make some changes before XOrganism could be competitive.

3.2 Protective Farmer

The benefit function was inspired by what we thought a human player would consider while deciding where best to move. We realized that given a point on the board, factors considered while deciding whether or not to move towards that point included not only the Euclidean distance to that point, but also the number of opponents, concentration of balls in the general vicinity of that point, and distance of that point from our home goal. Accounting for the proximity of opponents was particularly important becuase it gave us a measure of how susceptible to change the benefit of a particular cell was: the nearer the opponents, and the higher the number of opponents, the more likely it was for balls in the cell in question to be kicked out. This allowed us to take into consideration not just how the board was laid out now, but also gave us a proxy for how each cell might change in the near future.

The following is the benefit function for each cell that was calculated from the perspective of each player on the board so as to allow for the player to move to the cell that offered him the greatest benefit:

$$B_{i,j} = G_{i,j} - D_{i,j}$$

where $G_{i,j}$ is the gain offered by that cell and its neighboring 4×4 cluster of cells to the player and $D_{i,j}$ is the distance penalty for that cell given the position of the player and the cell.

$$G_{i,j} = \sum_{a:|a-i|\leqslant 2} \sum_{b:|b-j|\leqslant 2} \alpha \times numBalls(cell_{i,j}) - \beta \times numOpponents(cell_{i,j})$$

and

$$D_{i,j} = \gamma \times cellDistanceFromPlayer(cell_{i,j}) + \delta \times numKicksFromCellToOurHomeGoal(cell_{i,j})$$

where $\alpha = 1.2$, $\beta = 0.4$, $\gamma = 1$ and $\delta = 5$ are weights. These weights were assigned based on the optimal values we found through emperical observations of our performance against other teams over a range of weights.

3.3 SimplePlayer

On the second iteration of our benefit function, we accounted for ball and opponent clustering by keeping track of the number of balls and opponents surrounding a particular cell. This allowed our players to seek out areas of high benefit, rather than single cells.

After initial trials with a large radius, we settled on a radius of 4. We observed that larger radii caused the players to move back and forth somewhat inefficiently, and hypothesize that these larger zones changed too quickly for players to capitalize on them.

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 $\mathsf{Energy}_{organism} \geqslant M - u$). 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

In the early phases, we enumerated the different placement strategies and switched according to values of P and K. Our final implementation was fairly simple, and consisted of a random placement of players near the homegoal, with a bounded parameter on k determining their range.

By starting in a tight cluster bounded by the square from our goal out $\text{MIN}(K, 10)$ spaces, our players would fan out across the board, while the kicking algorithm led to a trail of k-spaced players who formed a bucket brigade. The relatively small square space for player placement was chosen as it balanced the advantages of fanning outward from the goal with scoring an initial k-distanced ball for each player.

4.3 SimplePlayer

As we found through our trials, using a benefit function to determine where a player should go runs into problems when players want to go to the same cell. The players will eventually converge and go to the same cells the rest of the game. To prevent this major inefficency, we implemented a locking mechanism where only one player can go after a single cell at a time. However, we found that although this works great through most of the game, when the number of balls is very small, it becomes better for multiple players to go after the same ball to have a higher likelyhood of kicking the ball to the goal. As such, when the number of balls reaches the number of players on the field, we turn off the locking mechanism.

After only locking the cell that the player is going towards, we also lock a radius around that cell. This allows us to create zones such that only one player clearing a zone.Through experimentation, we found that choosing a radius of max(0,3-P/10) works well. With higher number of players, the smaller the zones should become because there is not enough space to create so many zones. Addditionally, it is more likely that opponents will steal balls from a particular zone before a single player can clear that zone out.

5 RESULTS

Generally speaking, we had mixed results. Our rank ranged from 1^{st} to 7^{th} place. Our most common standing was 3^{rd} place. We observed that scores followed relative rank, so we chose rank as the proxy for overall performance in the tournaments.

The overall trend in our performance was that we did better than average given low-to-moderate k-values (less than 15), and any p-values. This makes sense in terms of our initial placement strategy. With very high values of k (at least 30), our players simply started too close to our goal, and did not take advantage of the potential for long-range kicking. Teams starting in the middle were able to clear balls in a single kick, so that even when our players reached the middle sections quickly, there were few balls left. Contrarily, our players remained competitive with smaller k. We think that starting players near the goal led to more effective bucket brigading as described in the initial placement section above. Larger values of p did not have a major impact on performance because more players allowed for a more effective covering of the map while maintaining a bucket brigade path back to the goal. It is possible that parameterizing our intitial placement by p as well as k could have improved our scores, but it may have negatively impacted the bucket brigading, and led to similar results.

Two notable configurations are $P, K = 45, 1$ and $P, K = 30, 1$. We placed third and first in these categories, respectively. This was a surprise to us, as we had not optimized for single-player configurations at all. We attribute our success here to the globally-optimal strategy we adopted, so that in slower games with long ranges, our player was able to seek out the most beneficial balls and ball clusters across the entire map. This is in contrast with games with higher p, where the board layout changed so rapidly that a global-optimized strategy did not provide much benefit.

Our results are summarized in the following figures.

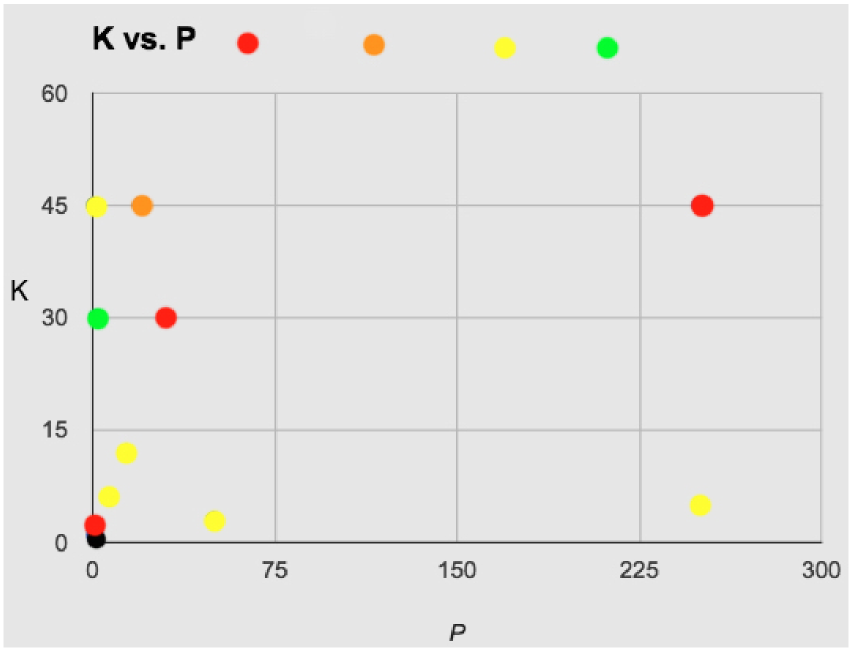
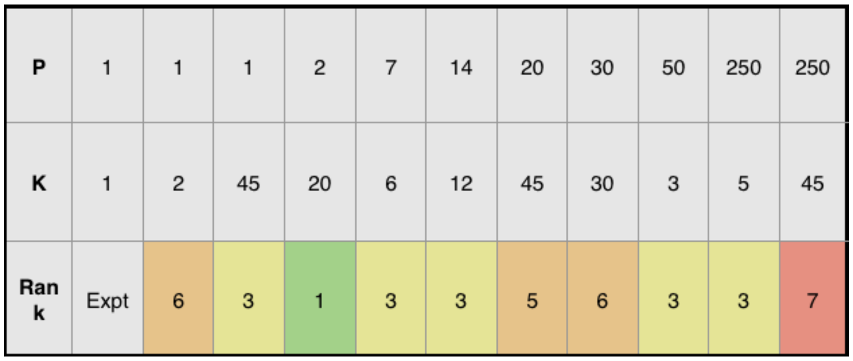


Figure 1: P on the horizontal vs K on the vertical axis. Observe the yellow 3rd place band for low k



P	1	1	1	2	7	14	20	30	50	250	250
K	1	2	45	20	6	12	45	30	3	5	45
Ran k	Expt	6	3	1	3	3	5	6	3	3	7

Figure 2: Table of rankings per configuration

6 CONTRIBUTIONS

Just as we adopted many good ideas by other teams (see Acknowledgements section), we contributed two major ideas to the group discussion over the course of the project:

1. **Benefit function -** Modelling how a human player would have made his/her movement decision gave us some key insights, especially with the realization that the runtime efficiency of our decision algorithm was not a constraint since this was only a 32×32 board. We realized that each player on the team could survey the entire board at every time-step and make a player-specific decision of where best to move, taking into account information about ball positions, opponents' positions as well as teammates' current and future positions. In the week after we brought up these insights in class, we saw that many other groups also adopted similar hotspot strategies (i.e. strategies that surveyed the board and picked out the best places to move towards). We were pleased to see that this was an insight that persisted through all further discussions in class and was validated, in different forms, through the strategies of various other teams.
2. **Composable Players and Game Phases -** Although we realized early on that the game would likely be characterized by distinct phases in which certain strategies would dominate, it was not until week 2 when other groups really started to pay attention to this idea. This was largely because our performance that week was extremely dominant after we composed `GridPlayer` in the early game and `DerickPlayer` for the rest of the round! Other groups began to think about how they could tailor their strategy to perform well in all phases of the game, or at leasthold their own. One group even took this idea to the extreme, creating a `ReinforcementLearningPlayer` that essentially composed *all* players using a 1NN algorithm!

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.