Software Engineering Course Project

El Farol Bar Problem and the Minority Game

Project designed by Ivan Marsic

Department of Electrical and Computer Engineering

Rutgers University

Project website: http://www.ece.rutgers.edu/~marsic/books/SE/projects/

This project develops a software simulator that models a population of simple brains engaged in decision-making and learning. It is intended to be done by a team of 4-6 undergraduate students during an academic semester, in conjunction with lectures and other class activities. Other related projects and a software engineering textbook are available for download at this website: http://www.ece.rutgers.edu/~marsic/books/SE/

1. Project Description

"Nobody goes there anymore. It's too crowded" —Yogi Berra

El Farol Bar problem was proposed by W. Brian Arthur in 1994 (http://en.wikipedia.org/wiki/El_Farol_Bar_problem). It is an example of inductive reasoning in a scenario of bounded rationality. Due to a limited knowledge and analyzing capability of agents, inductive reasoning generates a feedback loop where the agent commits an action based on its expectations of other agents' actions. These expectations are built based on what other agents have done in the past. Inductive reasoning assumes that with the help of feedback, agents could ultimately reach perfect knowledge about the game and arrive on steady state.

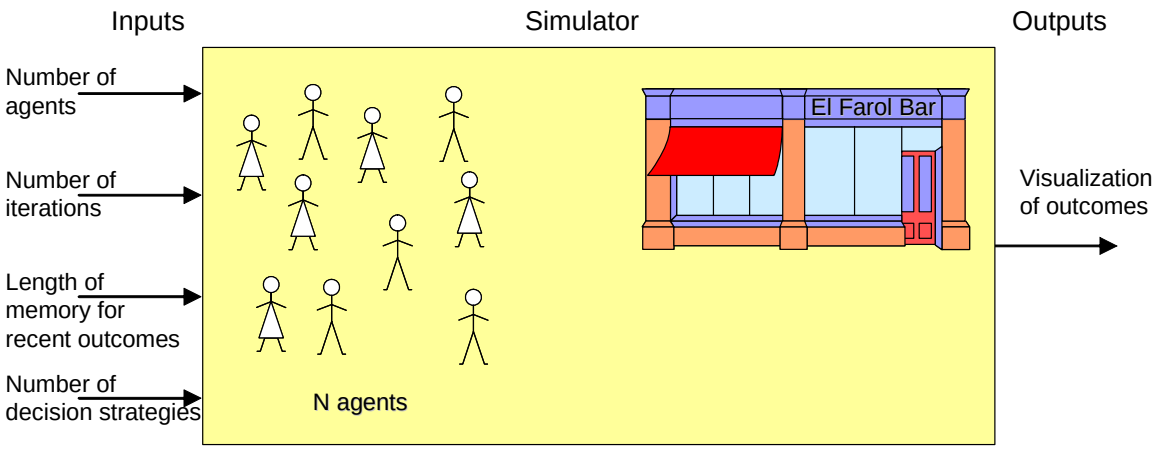


Figure 1: El Farol Bar problem simulation.

The problem is posed in the following way (Figure 1): *N* people have to decide independently each week whether to go to a bar that offers entertainment on a certain night. The space in the bar is limited and the evening is enjoyable if it is not too crowded—specifically, if fewer than 60% of the possible *N* people are present. There is no prior communication between the agents and the only information available is the number of people who came in past weeks. It is necessary for everyone to decide at the same time whether they will go to the bar or not, and they cannot wait and see how many others go.

There is no deductively rational solution to this problem, since we are given only the number attending in the recent past; a large number of expectation models might be reasonable. So, without the knowledge of which model other agents might choose, an agent cannot choose his or hers model in a well defined way. If all believe most will go, nobody will go, invalidating that belief. Similarly, if all believe very few will attend; all will end up in the bar. Also, agents need not necessarily know how many total agents are participating in the game, but they do know what fraction of agents attended the bar in past weeks. For example, the total number of agents in system is $N = 100$ and attendances in recent weeks are (the rightmost is the most recent):

… 63, 42, 72, 53, 49, 36, 70, 39, 51, 40, 44, 84, 35, 19, 47, 54, 41 (current time)

Following are some of the possible predictors:

- Same as 3 weeks ago: 47

- Mirror image around 50 of last week's attendance: 59

- Minimum of last 5 weeks: 19

- Rounded average of last 3 weeks: 48

Each agent monitors his predictors by keeping an internal score of them which is updated every week by giving points (or not) to all agents, depending on whether they correctly predicted the outcome (or not). Each week an agent chooses his or her predictor with the highest score to decide his or her action. Even though this problem deals with non-market context it offers a very good framework to build a simple market model. El Farol Bar problem can simply be extended to market scenarios. At each time step agent can buy or sell an asset. After each time step, the price

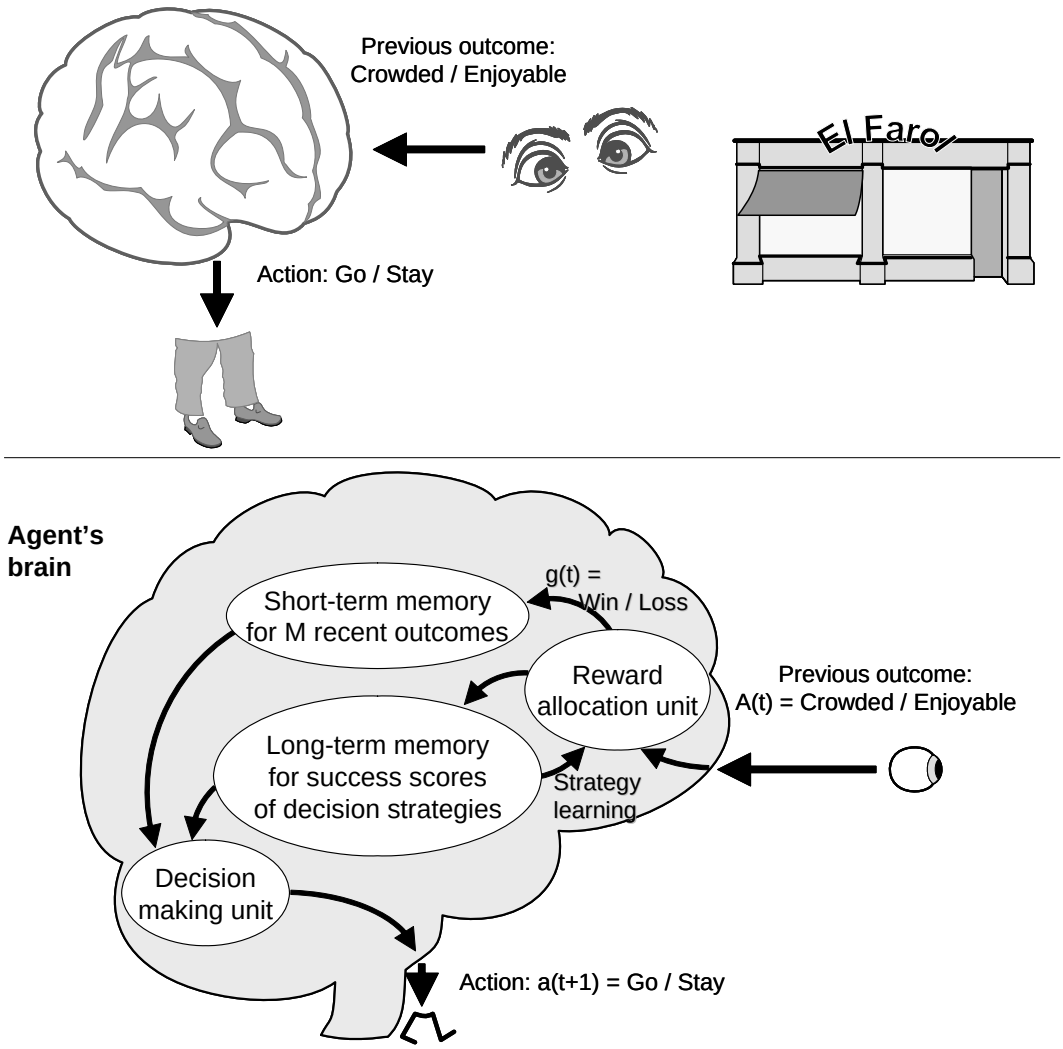


Figure 2: Minority game model of an agent playing the game.

of an asset is determined by a simple supply-demand rule. If there are more buyers than sellers, the market price will be high, and conversely, if there are more sellers than buyers, the market price will be low. If the price is high, sellers will do well, while if the price is low, buyers will win the round. Thus, the minority group always wins.

1.1 The Minority Game

One variant of the El Farol Bar problem is the minority game proposed by Yi-Cheng Zhang and Damien Challet from the University of Fribourg. In the *minority game*, an odd number of players N each must choose one of two choices independently at each turn. This is illustrated in Figure 2, top row. At each time step of the game (iteration), each of the N agents takes an action deciding either to go to the bar (denoted as $a_i(t) = +1$) or to stay at home (denoted as $a_i(t) = -1$). The payoff of the game is to declare that the agents who take minority action win, whereas majority loses. We calculate the differential headcount at El Farol's as

$$A(t) = \sum_{j=1}^{N} a_j(t) \tag{1}$$

The sum $A(t)$ will never be zero because of the odd total number of agents N. If $A(t) < 0$, majority of the agents stayed home and the bar at time t was "enjoyable." Conversely, if $A(t) > 0$, majority of the agents stayed home and the bar at time t was "crowded." We can simplify things by talking only about which side is winner (those who stayed home or those who went to the bar), without mentioning the actual attendance number.

The payoff for an agent i is given by:

$$g_i(t) = - a_i(t) \cdot A(t) \tag{2}$$

The function $g_i(t)$ represents outcome of the current round of the game for agent i and ensures that agents with minority action are rewarded. This means for $g_i(t) > 0$ agent i won the round and for $g_i(t) < 0$ agent i lost the round. The absolute value of g_i represents the margin by which agent won or lost the round. We will assume that the actual value is not of interest and agents only need to know the binary outcome: "win" (denoted by 1) or "loss" (denoted by 0). In other words,

$$g_i'(t) = \begin{cases} 0 & \text{if} \quad g_i(t) < 0 \\ 1 & \text{if} \quad g_i(t) > 0 \end{cases} \tag{2'}$$

It is assumed that agents are limited in their computational abilities and they can only retain the last M bar outcomes in their *short-term memory* (Figure 2, bottom row). This memory is "short-term" because of its fleeting nature: a new fact drives the oldest fact out of the memory and no trace is left of the lost fact in this memory. This memory acts as a shift register: a new bit will push the oldest bit out. Each agent makes their next decision based only on these M bits of historical data.

Given a sequence of last M outcomes, there are 2^M possible inputs for agent's decision making. A *strategy* specifies what the next action is (whether to "go" or to "stay") for every sequence of last M outcomes. For example, given M =3 and a historic sequence of "win," "loss," "win," i.e., 101, then a recommended action may be to "stay." Therefore, part of an example strategy is

("win," "loss," "win") → "stay"	or	101 → –1

A complete strategy should specify what to do for every possible input sequence. Therefore, if agents consider last M outcomes then each strategy should have 2^M entries. Each entry associates a specific sequence of last M outcomes to a recommended action. An example strategy is given in Figure 3.

Because for a given M there are 2^M possible inputs, the total number of possible strategies is 2^{2^M}. For M = 3, the total number of strategies is 256 and for M = 5 the total number of strategies is 4,294,967,296. It has been observed that agents tend to perform poorly if the number of strategies they should consider is too big. In general, the average performance of agents tends to degrade significantly if the number of assigned strategies S is more than 8. However, the overall operation of the market model is not greatly affected by the choice of S. The reason for this behavior is that agents are more likely to get confused if they are provided with bigger strategy set and they would switch the strategy often if another strategy appears to be slightly advantageous compared to the one currently in use. Setting a threshold for allowing the switch could improve this result.

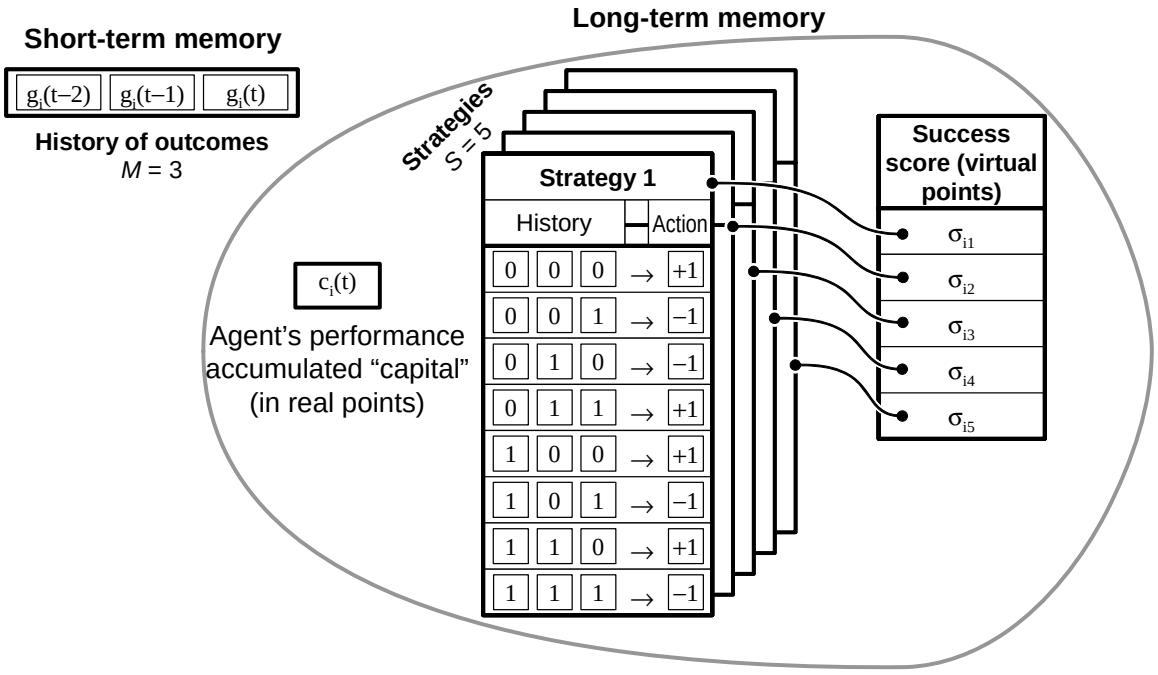


Figure 3: Data memorized by each agent in their short-term and long-term memories. Five strategies are initially selected randomly from the strategy space of 256 possible strategies.

We will assume that initially, at the start of the game, each agent chooses randomly S strategies out of the pool of 2^{2^M} strategies, where S is much smaller than 2^{2^M}. For example, in Figure 3 we assume that each agent memorizes $S = 5$ strategies. Because each agent chooses its own strategies independently of other agents, other agents may or may not share some of the strategies that this agent chose. These S strategies are memorized in the agent's *long-term memory* and will remain there for the duration of the game.

As the agent plays the game, each strategy will be scored based on how successful it might have been in securing the winning outcome for this agent. In an attempt to learn from the past mistakes, after each round, each agent assigns one virtual point to all its strategies that might have correctly predicted the actual outcome, i.e., strategies that would have placed the agent into the minority group. Thus agent reviews not only the strategy it has just used but all the strategies in its long-term memory that could have actually come up with the right prediction. Consider an agent i, and an outcome of the current round t given as $A(t)$ in Eq. (1). For each strategy s_{ij}, where $1 \le j \le S$, the agent assigns virtual points as follows. If the strategy s_{ij} suggested staying home ($a_{ij} = -1$), and the bar turned out to be crowded ($A(t) > 0$), then this strategy is rewarded by adding one virtual point. Conversely, if this strategy suggested staying home and the bar turned out to be enjoyable ($A(t) < 0$), then this strategy's merit score should remain unchanged (alternatively, the score could be lowered). Similar rewarding is done for strategies for which the associated actions are $a_{ij} = +1$ (go to the bar). This can be written as

$$\sigma_{ij}(t) = \begin{cases} \sigma_{ij}(t-1) & , \quad \text{if} \quad \left(a_{ij} \cdot A(t)\right) < 0 \\ \sigma_{ij}(t-1) + 1 & , \quad \text{if} \quad \left(a_{ij} \cdot A(t)\right) > 0 \end{cases} \tag{3}$$

Initially, the success scores for all strategies are reset to zero, i.e., $\sigma_{ij} = 0$ for all i and j. For the first round, each agent selects a strategy randomly from its long-term memory. For example, in Figure 3 the agent would select any of the five strategies randomly. For subsequent rounds, agents pick the strategy with the highest success score and make decision based on it (with ties being broken at random). Since agents keep track of how their strategies are performing, update their points, and pick the strategy that is performing best, the agents are constantly adapting.

Each agent also accumulates "capital" reflecting its overall score $c_i(t)$, so that the agent gets a real point only if the strategy used happens to win in the next play. Each agent tends to maximize its capital (accumulated points) and its performance is judged only on its time averaged capital gain.

In summary, the minority-game simulator works as follows:

Step 1 (Initialization): Each agent initially selects randomly S strategies from the strategy space of 2^{2^M} possible strategies and stores them in its long-term memory. The success scores for all strategies are reset to zero, i.e., $\sigma_{ij} = 0$ for all i and j. The short-term memories are initialized, either to a fixed value, e.g., all "loss," or randomly initializing each slot with "win" or "loss."

Step 2: Each agent retrieves the history of recent outcomes from its short-term memory and uses this sequence to look up the entries in its strategies. The agent selects the strategy with the highest success score σ_{ij} (with ties being broken at random). Agent i decides to pursue the action $a_i(t)$ for the current round t recommended by its most successful strategy, i.e., $a_i(t) = a_{ij}$, where $j = \max_j \arg(\sigma_{ij})$.

Step 3: Each agent receives the outcome of the current round t as $A(t)$ computed using Eq. (1). Based on this, the agent computes its reward function $g_i(t)$ using Eq. (2) and stores it in its short-term memory. The previous content of the short-term memory is shifted so that the oldest value $g_i(t - (M - 1))$ is discarded from the memory.

Step 4: Each agent performs strategy learning by updating its strategy scores using Eq. (3). The agent also updates its accumulated capital $c_i(t) = c_i(t - 1) + g_i(t)$.

Step 5: Each agent goes to Step 2 and does new iteration, unless the maximum number of iterations is reached.

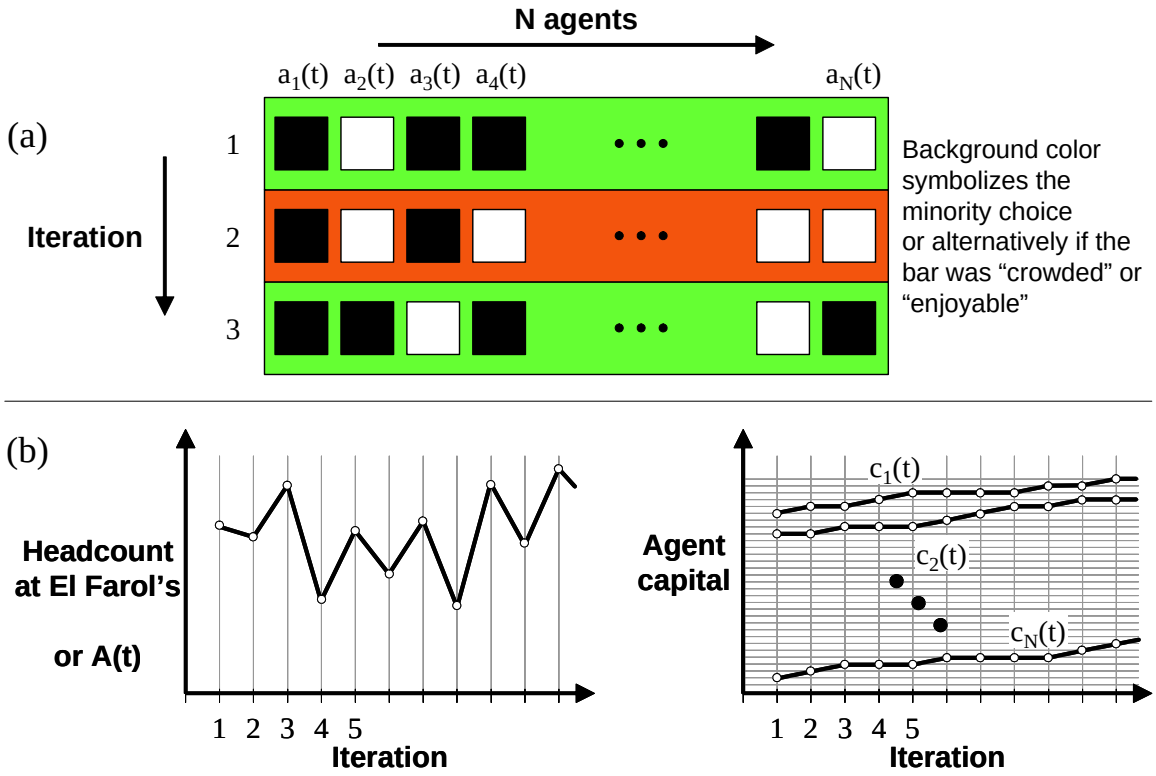


Figure 4: Ideas for output visualization of the outcomes of game turns.

1.2 Outcome Visualization and Interaction

Some ideas for visualizing the game outcomes are shown in Figure 4. The student should consider these only as a suggestion and come up with his or her ideas.

It would be of interest to somehow visualize the contents of agents’ long-term memories (strategies and associated merit scores), either individually for each agent or combined together for all agents. It would also be interesting to display which strategy each agent picked for the current round.

Interaction with the game may include the option to interrupt the game and examine the content’s of each agent’s “brain.” The student should decide what information to expose and how to visualize it.

Another option is to allow the user to substitute one of the agents. That is, the simulator runs the minority game with $N-1$ electronic agents and the user. The user interface should include to buttons for the user to inform the simulator about his or her action (“stay” or “go”) for each iteration. The system should also visualize the bar state (“crowded” or “enjoyable”) and perhaps the visualization in Figure 4 may be sufficient. However, the user’s accumulated capital should be highlighted so the user can distinguish their own score from other agents.

2. Extensions

More ambitious students should consider some or all of potential extensions to the basic minority game that are described next.

This original minority game model functions as infinite time horizon market where agents keep collecting the virtual points throughout the duration of the game. One virtual point scored early in the game for a particular strategy counts as much as a point scored during the most recent iteration. However, we know that in real life more recent experiences count for more than the old ones. Therefore, we may consider modifying the strategy-scoring equation (3). One idea is to assign more weight to the recent outcomes. Hence, we may use the following equation

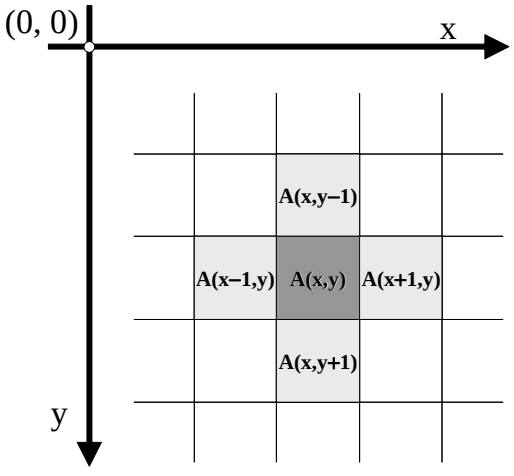
$$\sigma_{ij}(t) = \begin{cases} \alpha \cdot \sigma_{ij}(t-1) & , \quad \text{if} \quad \left(a_{ij} \cdot A(t)\right) < 0 \\ \alpha \cdot \sigma_{ij}(t-1) + 1 & , \quad \text{if} \quad \left(a_{ij} \cdot A(t)\right) > 0 \end{cases} \tag{4}$$

where the parameter α, $0 < \alpha \leq 1$, plays a role of a memory-decay factor. With each new iteration, the old virtual-point scores are diminishing and the most recent ones contribute more significantly to the overall strategy-success score. The parameter α could be an input parameter for the simulator and you should run the game with different values to discover the one which yields the best results.

Another problem with strategies is that the agent sticks with the same strategies for the duration of the game. It can happen, although very unlikely, that all of the agent's strategies for a given historic sequence suggest the same action. For example, assuming $M = 3$ and $S = 5$ as in Figure 3, given a historic sequence of "win," "loss," "win," all strategies may recommend the next action as "stay." This is unlikely to be always the best action. One option is to periodically allow the agent to discard poorly performing strategies and substitute them with new strategies. An issue here is that the remaining old strategies have an unfair advantage since they will start with non-zero merit scores that they gained earlier. To rectify this problem, an option is to reset the virtual-point scores for all strategies. An alternative is to initialize the merit scores for the new strategies with an average score of the old strategies. The student should experiment with different solutions and find which works best.

Another extension is to make a networked version of the game, so several geographically distributed users could substitute the corresponding number of electronic agents and play the game.

Yet another extension is to allow agents to form cliques of friends. In the basic minority game, each agent acts independently of other agents. But in real life we know that people are influenced with their friends' opinions. To simulate this feature, we connect agents into cliques. Each clique has five agents that are selected as follows. Because there are N agents, we create a rectangular grid with dimensions $N_X \times N_Y = N - 1$. In the simplest case $N_X = N_Y = (N/2)$. We position each agent on one cell of the grid. Each agent forms a clique its four



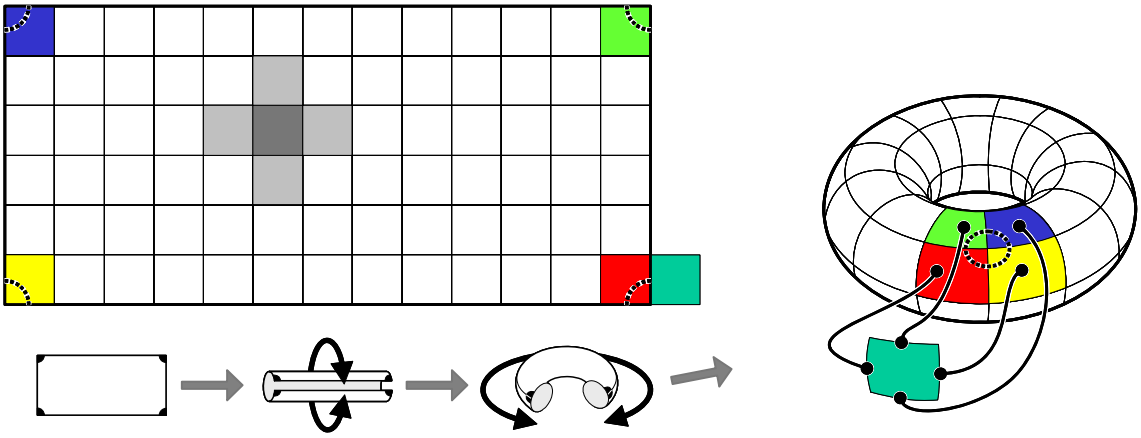


Figure 5: Creating 4-member cliques for boundary agents. Given $N = 79$ agents, we fold a 13×6 rectangular grid to a torus, so all boundary agents have four neighbors. For the remaining 79th agent, one option is to connect it to the four corner agents.

closest neighbors on the grid, as illustrated in the figure on the right.

When deciding about the action for the next round, each agent first finds what its best strategy recommends (as in Step 2 of the algorithm outlined above). Then each agent gathers the actions from the four neighbors in its clique and decides that its action will be the majority vote of the clique. This always yields an unambiguous action because of the odd number (five) of agents in the clique. Regardless of the way each agent in a clique votes, there will never be a tie.

There are two problems to resolve. First, we need to determine how the boundary agents for a clique. Second, because N is odd, there will be one agent which does not fit to the rectangular grid. A solution is shown in Figure 5.

3. Additional Information

Additional information about this project can be found at the project website, http://www.ece.rutgers.edu/~marsic/books/SE/projects/.